On The Pareto-Following Variation Operator For Fast Converging Multiobjective Evolutionary Algorithms

A.K.M. Khaled Ahsan Talukder

Submitted in total fulfilment of the requirements of the degree of

Master of Engineering Science

Department of Computer Science and Software Engineering
THE UNIVERSITY OF MELBOURNE

November 2008

Abstract

The focus of this research is to provide an efficient approach to deal with computationally expensive Multiobjective Optimization Problems (MOP's). Typically, approximation or surrogate based techniques are adopted to reduce the computational cost. In such cases, the original expensive objective function is replaced by a cheaper mathematical model, where this model mimics the behavior/input-output (i.e. design variable – objective value) relationship. However, it is difficult to model an exact substitute of the targeted objective function. Furthermore, if this kind of approach is used in an evolutionary search, literally, the number of function evaluations does not reduce (i.e. The number of original function evaluation is replaced by the number of surrogate/approximate function evaluation). However, if a large number of individuals are considered, the surrogate model fails to offer smaller computational cost.

To tackle this problem, we have reformulated the concept of surrogate modeling in a different way, which is more suitable for the Multiobjective Evolutionary Algorithm (MOEA) paradigm. In our approach, we do not approximate the objective function; rather we model the input-output behavior of the underlying MOEA itself. The model attempts to identify the search path (in both design-variable and objective spaces) and from this trajectory the model is guaranteed to generate non-dominated solutions (especially, during the initial iterations of the underlying MOEA – with respect to the current solutions) for the next iterations of the MOEA. Therefore, the MOEA can avoid re-evaluating the dominated solutions and thus can save large amount of computational cost due to expensive function evaluations. We have designed our approximation model as a variation operator – that follows the trajectory of the fronts and can be "plugged-in" to any kind of MOEA where non-domination based selection is used. Hence it is termed

– the **“Pareto-Following Variation Operator (PFVO)”**. This approach also provides some added advantage that we can still use the original objective function and thus the search procedure becomes robust and suitable, especially for dynamic problems.

We have integrated the model into three base-line MOEA’s: “Non-dominated Sorting Genetic Algorithm - II (NSGA-II)”, “Strength Pareto Evolutionary Algorithm - II (SPEA-II)” and the recently proposed “Regularity Model Based Estimation of Distribution Algorithm (RM-MEDA)”. We have also conducted an exhaustive simulation study using several benchmark MOP’s. Detailed performance and statistical analysis reveals promising results. As an extension, we have implemented our idea for dynamic MOP’s. We have also integrated PFVO into diffusion based/cellular MOEA in a distributed/Grid environment. Most experimental results and analysis reveal that PFVO can be used as a performance enhancement tool for any kind of MOEA.

Publications

Part of the work which is described in this thesis has been published as conference proceedings. Following is the list of the papers which have been published during the course of the candidature.

1. *A Pareto Following Variation Operator for Fast-Converging Multiobjective Evolutionary Algorithms* – A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya. In proceedings of the ACM Annual Genetic and Evolutionary Computation Conference 2008 (GECCO '08). Published by ACM Press, pages 721 - 728, 2008
2. *Towards High Speed Multiobjective Evolutionary Optimizers* – A.K.M. Khaled Ahsan Talukder. In proceedings of the GECCO 2008 Graduate Student Workshop. Published by ACM Press, pages 1791 - 1794, 2008.
3. *A Pareto Following Variation Operator for Evolutionary Dynamic Multi-objective Optimization* – A.K.M. Khaled Ahsan Talukder and Michael Kirley. In proceedings of the IEEE Congress on Evolutionary Computation 2008 (CEC 2008) within IEEE World Congress on Computational Intelligence. Published by IEEE Press, pages 2270 - 2277, 2008.
4. *Multiobjective Differential Evolution for Workflow Execution on Grids* – A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya. In proceedings of the 5th international workshop on Middleware for Grid Computing 2007 (MGC '07) within ACM/IFIP/Usenix Middleware Conference 2007 (Middleware 2007). Published by ACM Press, 2007.

5. *The Pareto-Following Variation Operator as An Alternative Approximation Model and Analysis on Its Applicability Issues* – A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya. Submitted to IEEE Congress on Evolutionary Computation 2009. Under review.
6. *Multiobjective Differential Evolution for Scheduling Workflow Applications on Global Grids* – A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya. Submitted to the Concurrency and Computation: Practice and Experience (Special Issue: Middleware for Grid Computing 2007). Published by John Wiley & Sons, Ltd., Under review (Invited Paper).
7. *The Pareto-Following Variation Operator: An Alternative Approximation Model for Evolutionary Multiobjective Optimization* – A.K.M. Khaled Ahsan Talukder, Michael Kirley and Rajkumar Buyya. Submitted to the IEEE Transactions on Evolutionary Computation. Published by IEEE Press, Under review.

Declaration

This is to certify that

1. The thesis comprises only my original work towards Master of Engineering Science,
2. Due acknowledgement has been made in the text to all other material used,
3. The thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

A.K.M. Khaled Ahsan Talukder, November 2008

Acknowledgements

I would like to express my profound gratitude and deep appreciation to my Masters by Research advisor, Dr. Michael G. Kirley, who supervised my overall research and guided me throughout the course of my Masters of Engineering Science degree. Without his help and guidance, this research would not be possible. I would also like to thank my co-advisor Associate Professor Dr. Rajkumar Buyya for offering his extraordinary insight on diverse aspects of Grid computing. I would also like to thank my supervisors for financially supporting me during my last term at The University of Melbourne, arranging visits to conferences and providing all resources needed for my research.

I would also like to thank the group members in GRIDS laboratory for the interesting weekly meetings, especially to Dr. Christian Vecchiola – without his help the parallel implementations on the Grid would not be possible. I would also like to thank Dr. Xiaodong Li and Associate Professor Vic Ciesielski of ECML group in RMIT for their interesting discussions on different aspects of nature inspired computing during my initial years at The University of Melbourne.

Thanks to my colleagues at CSSE department, Kapil Kumar Gupta, Allauddin Bhuiyan and Dr. Robert Stewart for lively discussions, insightful comments and honest opinions on my research. My heartfelt thanks to my parents, especially to my mother, who supervised my home schooling during the first six academic years of my life, which served as the foundation for my education since then. Finally, my special thanks to my wife, Taibun Nessa, who always stayed by my side, offered her support during the hard time with research, and did everything to uphold my spirit whenever I was down.

To my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past two decades, Multi-objective Evolutionary Algorithms (MOEA's) have become one of the major tools in Multidisciplinary Design Optimization (MDO) and Operations Research (OR). MOEA's are now a well established technique, both in terms of methodologies and algorithm development [1], [2], [3]. However, one of the major difficulties when applying evolutionary algorithms (EA's) to real-world problems is the computational costs associated with the large number of function evaluations necessary to obtain a range of acceptable solutions. A typical problem can be considered as high-fidelity engineering design [4], [5], [6]. Often in these problems, the function evaluations are time-consuming and are obtained by solving a large number of numerical calculations [7].

The use of distributed systems, where each fitness evaluation is performed on a separate processor, does offer one approach to reduce the computational time. Such models, however, typically require a large number of networked computers (scaling in size from local clusters to full Grid deployment) and an adequate parallelization of the numerical code [8], [9], [10]. However, a parallel approach *per se* does not necessarily reduce the number of function evaluations.

Recently the development of techniques enabling a reduction in the number of function evaluations, without reducing the solution quality, has sparked the interest in the field of both Multi-objective and Single-objective Evolutionary Optimization [11], [12], [13], [6], [14]. An on-going challenge, therefore, is to develop good approximate methods or **"acceleration"** methods that can be used to solve multi-objective problems while considering the number of objectives and the possible interaction between them. Typically, these acceleration models known as **"Surrogate"** or **"Meta-models"** [15], employ a computationally cheaper mathematical model which, replaces the actual expensive functions.

1.1 Motivation

When tackling real world problem, an important performance criterion is the convergence speed. Although, EA's are good at finding (near) optimal solutions, the quality of the solutions is dependent on the execution cycle of the EA, generally termed as **"generation"**. As the number of generations increases for a given EA, we are likely to find better result with respect to the previous generations. However, if each of these generational cycles take a considerably long time to execute, the effectiveness of EA's when tackling real world problems is degraded. Hence, there is a need for **"Surrogate/Meta-models"** or **"Intelligent Operators"**. The **"Surrogate Model"** replaces the original fitness function with a cheaper mathematical model that has smaller computational cost than the original one. On the other hand, **"Intelligent Operators"** guide the conventional EA in the right direction with increased speed [16], [17]. The goal of employing such technique is to speed up the normal convergence rate of an EA.

The motivation behind this research was to increase the normal convergence speed of any Pareto based MOEA's. In this thesis, we propose a novel **"Variation Operator"** that utilizes the objective values in the existing Pareto-front[1] to approximate the possible solutions for the next Pareto-front. Our model helps existing MOEA to follow the next Pareto-front without wasting computational cost on redundant crossover/mutation hence we call the model as **"Pareto-Following Variation Operator"**.

1.2 Aims and Objectives

1.2.1 Aim

The general aim of this research project is to design and develop a new approximation based algorithm to speed up the normal convergence rate of existing Multi-objective Evo-

[1]The concept of Pareto-front and Pareto-optimality will be covered in chapter 2

lutionary Algorithms (MOEA).

1.2.2 Objectives

The specific objectives of this project are outlined below -

- Design a novel approximation model, the **"Pareto-Following Variation Operator (PFVO)"**, which can be plugged into any population and Pareto-based Evolutionary Algorithm.
- Integrate the **PFVO** with three well-known MOEA's such as NSGA-II, SPEA-II and RM-MEDA; compare the performance of the **PFVO** enhanced algorithm with respect to different performance metrics, such as, Hypervolume and Epsilon indicator. Here, we need to consider the speed up of the hosting optimizer, therefore we have to compare the result with respect to the total number of function evaluations.
- Deploy the model on a Grid-based[2] parallel system to test the efficacy of **PFVO** with parallel MOEA's to solve computationally expensive multiobjective optimization problems.

1.3 Organization of The Thesis

The rest of the thesis is organized as follows. In chapter 2, we introduce the **"Multi-objective Optimization Problem" (MOP)** and discuss how to solve such problems using EA's. In addition, we also discuss baseline algorithms found in the literature to solve conventional MOP's. In chapter 3, we review alternative acceleration models, typically used with evolutionary algorithms. In the case of a typical **"Surrogate Modeling"**, the original objective functions (generally computationally very expensive) are replaced by cheaper mathematical models. However, in our case, we do not replace the original fitness function, rather we use an **"Intelligent Operator"** based model that can increase the convergence speed of the EA. In chapter 4, we introduce the **"Pareto-Following Variation Operator" (PFVO)** – the main contribution of this thesis. Implementation details are provided as well as a description of the plugging architecture with alternative

[2]The concept of Grid computing will be covered in chapter 7

"Multi-objective Evolutionary Algorithms (MOEA's)". Experimental results using several benchmark problems are also presented in chapter 5. In chapter 6, the focus shifts to dynamic MOP's, where the fitness functions change with time. A modified version of the **PFVO** is also implemented. Experimental results are presented as a proof of concept behind **PFVO**. Chapter 7 discusses the implementation of **PFVO** for a **parallel MOEA** algorithm, executed in a **"Global Grid"** environment. In chapter 8, we conclude the dissertation providing a summary of key contributions, addressing some of its existing limitations and identifying future research directions.

1.4 Contributions

The specific contribution of this project can be summarized as follows –

- Chapter 4: Development of the **PFVO** – the core contribution of this thesis. This work appeared in [18][3]. An extended version of this paper (integration mechanism with SPEA-II and RM-MEDA) entitled *"The Pareto-Following Variation Operator as An Alternative Approximation Model and Analysis on Its Applicability Issues"* is also submitted in *IEEE Congress on Evolutionary Computation 2009 (CEC-2009)*.
- Chapter 5: A modified implementation of the **PFVO** for dynamic multi-objective problems. This part was published in [19].
- Chapter 7: To conduct an exhaustive performance test, we have also deployed PFVO on parallel MOEA which will be executed on Global Grid environment. This work will be a part of another paper which will be submitted in *IEEE International Conference on E-Science and Grid Computing 2009 (e-Science 2009)*.
- An abridged version of this thesis entitled *"The Pareto-Following Variation Operator: An Alternative Approximation Model for Evolutionary Multiobjective Optimization"* is also submitted in *IEEE Transactions on Evolutionary Computation*.

[3]This paper only describes the integration mechanism with NSGA-II.

Chapter 2
Evolutionary Multi-objective Optimization

2.1 Basic Definitions

2.1.1 Global Optimization

Global optimization is the process of finding the global optimum (or minimum, since $min\{\vec{f}(x)\} = max\{-\vec{f}(x)\}$) within search space $\mathcal{S}$. The single-objective global optimization problem can be formally defined as follows [20] -

Given a function $f : \Omega \subseteq \mathcal{S} = \Re^n \rightarrow \Re, \Omega \neq \emptyset$, for $\vec{x} \in \Omega$ the value $f^* \doteq f(\vec{x}^*) > -\infty$ is called a global minimum if and only if

$$\forall \vec{x} \in \Omega : f(\vec{x}^*) \leq f(\vec{x}) \tag{2.1}$$

Then, $\vec{x}^*$ is the global minimum solution(s), f is the objective function, and the set Ω is the feasible region ($\Omega \subset \mathcal{S}$).

2.1.2 Multi-objective Optimization

Multi-objective Optimization (also called multi-criteria optimization, multi-performance or vector optimization problem) can then be defined as the problem of finding –

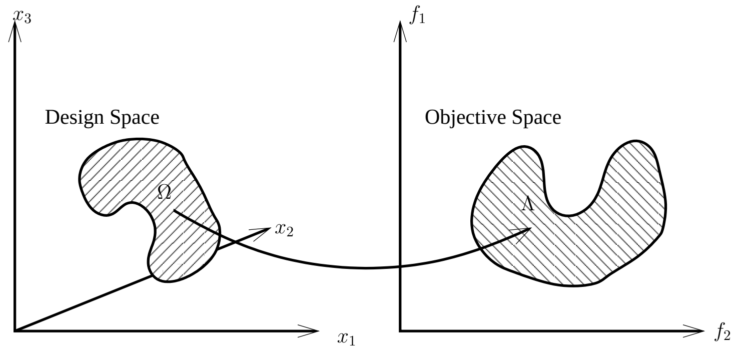


Figure 2.1: Representation of the decision variable space and the corresponding objective space

> A vector of design variables (decision variables), which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker [21].

Design Variables

The **Design Variables or Decision Variables** are the numerical quantities for which values are chosen in an optimization problem. These quantities are denoted as x_j, $j = 1, 2, \ldots, n$.

The vector of n design variables $\vec{x}$ is represented by -

$$\vec{x} = [x_1, x_2, \ldots, x_n]^T \tag{2.2}$$

Constraints

In most numerical optimization problems, there are restrictions imposed by the particular characteristics of the environment or resource available (e.g., physical limitations, time restrictions, etc.). These restrictions must be satisfied in order to consider that a certain solution is acceptable. Restrictions in general are called **constraints**, and they de-

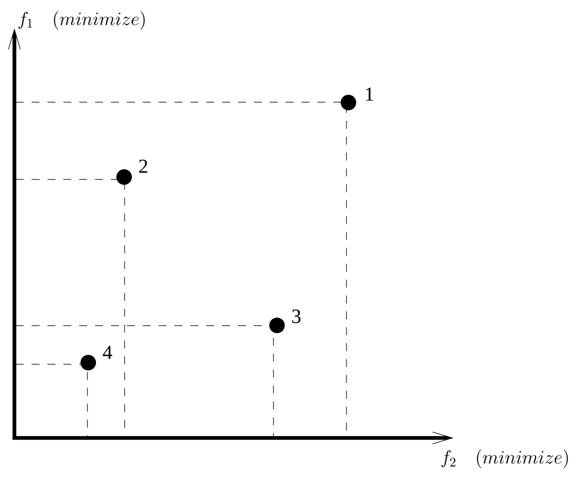


Figure 2.2: The concept of domination

scribe dependencies among design variables and constants (or parameters) involved in the problem, These constraints are expressed in the form of mathematical inequalities -

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{2.3}$$

or equalities -

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{2.4}$$

Note that p, the number of **equality constraints**, must be less than n, the number of design variables, because if $p \geq n$ the problem is said to be **over-constrained**, since there are no degrees of freedom left for optimizing (i.e., in other words, there would be more unknowns than equations). The number of degrees of freedom is given by $n - p$. Also, constraints can be **explicit** (i.e., given in algebraic form) or **implicit**, in which case the formulation to compute $g_i(\vec{x})$ (for any given vector $\vec{x}$) must be known.

The Concept of domination

Most multi-objective optimization algorithms use the concept of domination [22] [23] [24] [25]. In these algorithms, two solutions are compared on the basis of whether one dominates the other or not. Any solution $\vec{x}^{(1)}$ is said to dominate $\vec{x}^{(2)}$ or $\vec{x}^{(1)}$ is said to be non-dominated by $\vec{x}^{(2)}$ if the following conditions are true -

1. $\vec{x}^{(1)}$ is no worse than $\vec{x}^{(2)}$ in all objectives.
2. $\vec{x}^{(1)}$ is strictly better than $\vec{x}^{(2)}$ in at least one objective.

Say, we have two objective functions f_1 and f_2, both to be minimized. In Figure 2.2, we have 4 solutions and -

- Solution 4 dominates solutions 1, 2 and 3.
- Solutions 2 and 3 dominates solution 1.

If any of the two conditions mentioned above is violated, the solution $\vec{x}^{(1)}$ does not dominate $\vec{x}^{(2)}$. Hence in Figure 2.2, neither of solutions 2 or 3 dominate each other; they are non-dominated.

Multi-objective Optimization Problem

The general **Multi-objective Optimization Problem (MOP)** can now be defined as follows -

Find the vector, $\vec{x}^* = [x_1{}^*, x_2{}^* \ldots x_n{}^*]^T$ which satisfies m inequality and p equality constraints:

$$g_i(\vec{x}) \geq 0 \qquad i = 1, 2 \ldots m \tag{2.5}$$

$$h_i(\vec{x}) = 0 \qquad i = 1, 2 \ldots p \tag{2.6}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}) \ldots f_k(\vec{x})]^T \tag{2.7}$$

In other words, the aim is to determine from among the set of all values which satisfy (2.5) and (2.6) the particular set $x^*{}_1, x^*{}_2 \ldots x^*{}_n$ which yields the optimum values of all the objective functions. In MOP's there is no single solution rather we have to find all compromising (**Pareto-optimal**) solutions. A solution $\vec{x}^* \in \Omega$ is **Pareto-optimal** if for every $\vec{x} \in \Omega$ and $I = \{1, 2 \ldots k\}$ either,

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{2.8}$$

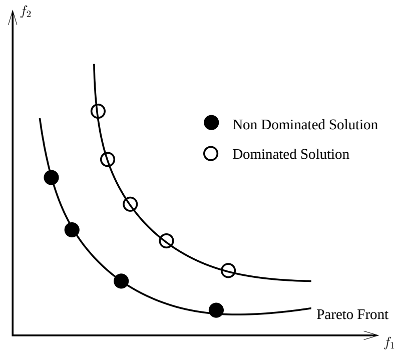


Figure 2.3: Dominated/Non-dominated solutions and the Pareto-front

or, there is at least one $i \in I$ such that

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{2.9}$$

The constraints given by (2.5) and (2.6) define the **feasible region** Ω and any point $\vec{x}$ in Ω defines a **feasible solution**. The vector function $\vec{f}(\vec{x})$ is a function which maps the set Ω into the set Λ which represents all possible values of the objective functions. Please refer to Figure 2.1 for the concept of objective space and design variable space. For a given MOP $\vec{f}(x)$, the **Pareto-optimal Set** ($\mathcal{PS}^*$) is defined as

$$\mathcal{PS}^* := \{x \in \Omega | \neg\exists x' \in \Omega : \vec{f}(x') \preceq \vec{f}(x)\} \tag{2.10}$$

Here the sign $\preceq$ refers to **Pareto-dominance**. A vector $\vec{u} = (u_1, u_2 \dots u_k)$ is said to dominate $\vec{v} = (v_1, v_2 \dots v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, i.e. $\forall i \in \{1, 2, \dots k\} : u_i \leq v_i \wedge \exists i \in \{1, 2, \dots k\} : u_i < v_i$. The concept of Dominated/Non-dominated solution and Pareto-front is illustrated in Figure 2.3.

Our goal is to find the set of all **Pareto-optimal** solutions and the corresponding objective values of this set is defined as **Pareto-front**. The **Pareto-front** ($\mathcal{PF}^*$) can be mathematically defined as,

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x) \dots f_k(x)) | x \in \mathcal{PS}^*\} \tag{2.11}$$

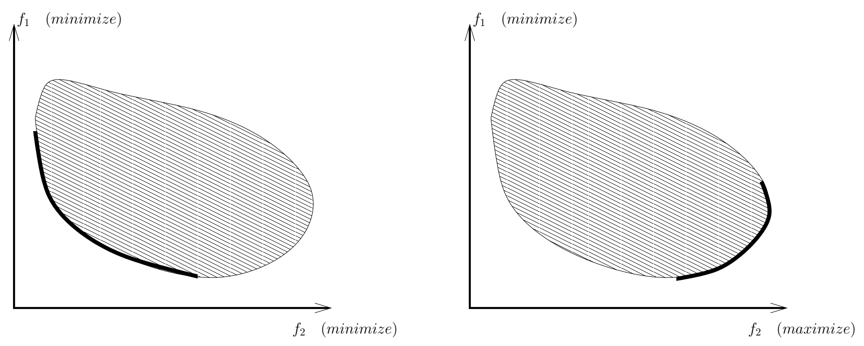


Figure 2.4: Different Pareto-optimal solution sets, for the same objective space, depending on maximization or minimization problem

The Pareto-optimal solutions are those solutions within the search space whose corresponding objective vector components can not be improved simultaneously. These solutions are also known as *non-inferior*, *admissible* or *efficient* solutions, with the entire set represented by $\mathcal{PS}^*$ or PS_{true}. Their corresponding vectors are known as *non-dominated*; selecting a vector(s) from this vector set (the Pareto-front set $\mathcal{PF}^*$ or $\mathcal{PF}_{true}$) implicitly indicates acceptable Pareto-optimal solutions. These are the set of all solutions whose vectors are non-dominated; these solutions are classified based on their *objective value* expression. Their expression (the nondominated vectors), when plotted in criterion space, is known as the *Pareto-front* [2], [1], [26]. In Figure 2.4, the left portion depicts a two objective Pareto-front where both objectives needs to be minimized and in the right portion, the Pareto-front is illustrated where f_1 needs to be maximize and f_2 needs to be minimized.

Just as there are "global" and "local" optimal solutions in single-objective optimization, there could be "global" and "local" Pareto-optimal fronts in MOP's. The Globally Pareto-optimal set is the non-dominated set of entire search space $\mathcal{S}$. Since the solutions of this set are not dominated by any feasible member of the search space, they are the ***optimal*** solutions of MOP. On the other hand, a locally Pareto-optimal set can be defined as -

If $\forall \vec{x} \in \mathcal{PS}'$, $\exists \vec{y} \in \mathcal{PS}$ and in the neighborhood of $\vec{x}$, such that,

$$||\vec{x} - \vec{y}||_\infty \leq \epsilon \tag{2.12}$$

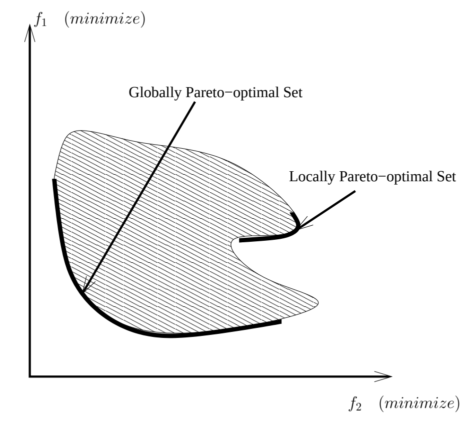


Figure 2.5: Global and local Pareto-optimal fronts

where ϵ is a small positive number dominating any member of the set $\mathcal{PS}'$, then solutions belonging to the set $\mathcal{PS}'$ constitute a locally Pareto-optimal set. In Figure 2.5, the shorter bold curve represents a local Pareto-optimal front. None of its points have any neighbor which dominate any member of the set. With these basic MOP definitions, we are now ready to delve into the structure of MOP's and the specifics of various MOEA's.

2.2 Basic Techniques to Solve MOP's

From the preceding discussions, we have seen that there is no single solution for a given MOP. So, classical methods like Linear Programming (LP), Non-Linear Programming (NLP), Quadratic Programming (QP) fail to provide a set of trade-off solutions. Although, there are numerical techniques that have been proposed to solve MOP's such as Normal Boundary Intersection (NBI) [27] and Normal Constraint Method (NC) [28], [29] etc., However, they are not capable of solving all types of MOP's.

General search and optimization techniques are typically classified into three categories: enumerative, deterministic and stochastic (random) [2]. Although, an enumerative search is deterministic; a distinction is made here as it employs no heuristics [30], [31]. Figure 2.6 shows common examples of each type.

As in the case of single-objective optimization, MOP has also been studied extensively. There exists many algorithms and application case studies involving multiple

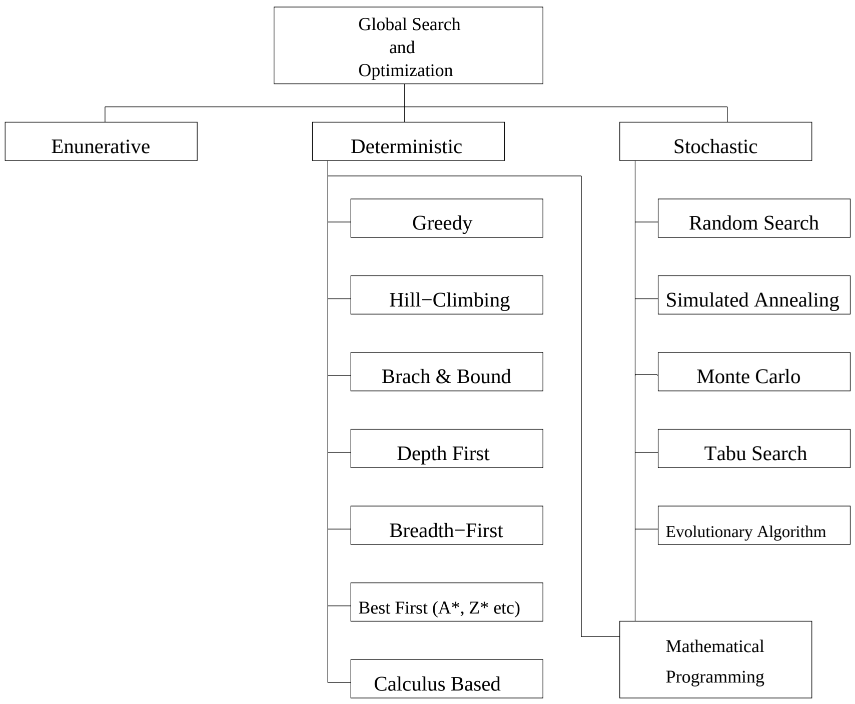


Figure 2.6: Global Optimization Approaches

objectives [32], [33]. The majority of these methods avoid the complexities involved in a true multi-objective optimization problem and transform multiple objectives into a single objective function by introducing user-defined parameters (see Figure 2.7). Thus, most studies in classical multi-objective optimization do not treat MOP's any differently to single-objective optimization problem. In fact, MOP is considered as an application of single objective optimization for handling multiple objectives. The studies seem to concentrate on various means of converting multiple objectives into a single objective. Many studies involve comparing different conversion schemes and provide reasons in favor of one conversion over another [1]. This is contrary to our intuitive realization that single-objective optimization is a degenerate case of MOP and MOP is not a simple extension of single-objective optimization.

It is true that most theories and algorithms for single-objective optimization are applicable to the optimization of multi-objective function. However, there is a fundamental difference between single and multiple objective optimization, which is ignored when we use transformation methods.

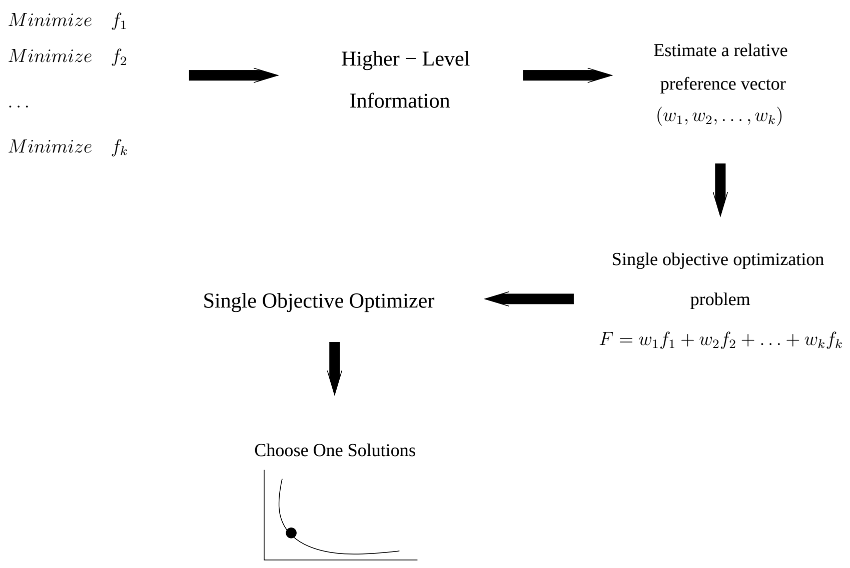

$Minimize \quad f_1$
$Minimize \quad f_2$
$\ldots$
$Minimize \quad f_k$
Higher − Level Information
Estimate a relative preference vector $(w_1, w_2, \ldots, w_k)$
Single objective optimization problem $F = w_1f_1 + w_2f_2 + \ldots + w_kf_k$
Single Objective Optimizer
Choose One Solutions

Figure 2.7: Schematic of preference based multi-objective optimization procedure

To address this issue more clearly, we first describe what is an ideal multi-objective optimization algorithm. According to [1], we found two basic requirements for an ideal MOP algorithm (the basic idea is illustrated in figure 2.8) -

- Find multiple trade-off solutions with a wide range of values for objectives.
- Choose one of the obtained solutions using higher level information.

It is important to realize that the trade-off solution obtained by using this preference based strategy is largely sensitive to the relative preference vector used in forming the composite function. A change in this preference vector will result (hopefully) in a different trade-off solution. However, an arbitrary preference vector does not result in a trade-off optimal solution to all problems. Besides this difficulty, it it trivial to realize that finding an appropriate relative preference vector is highly subjective and not straight forward. Classical MOP methods works according to this preference based strategy. So unless a reliable and accurate preference vector is available, the optimal solution found by such algorithms is highly subjective to the particular user. To solve this problem, we actually need an optimizer that can handle multiple solutions with multiple-objectives, simultaneously, so that it is possible to find all trade-off solutions.

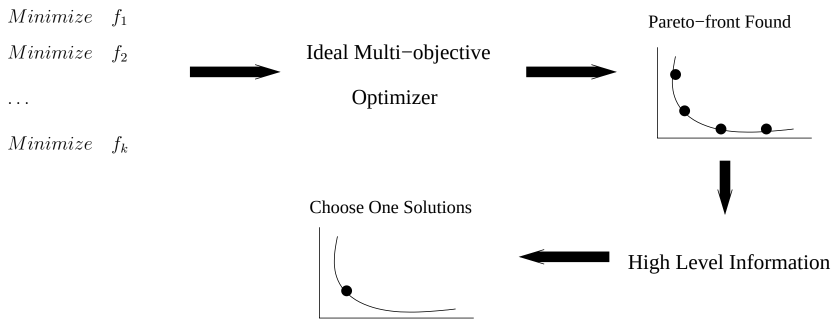


Figure 2.8: Schematic of an ideal multi-objective optimization procedure

2.3 Evolutionary Algorithms for MOP's

In the previous section, we have seen that the classic methods to solve multi-objective optimization is to follow the preference based approach, where a relative preference vector is used to scalarize multiple objectives. Since classical search and optimization methods use a point-by-point approach, where one solution in each iteration is modified to a different (hopefully better) solution, the outcome of using a classical optimization method is a single solution.

However, the field of search and optimization has changed over the last few years by introduction of a number of non-classical, unorthodox and stochastic search and optimization algorithms. Of these, EA's mimic nature's evolutionary principles to drive its search towards an optimal solution. One of the most striking differences to classical approach is that EA's use a population of solutions in each iteration, instead of single solution. Since a population of solutions are processed in each iteration, the outcome of an EA is also a population of solutions. If an optimization problem has a single optimum, all EA population members can be expected to converge to that optimum solution. However, if an optimization problem has multiple optimum solutions, an EA can be used to capture all Pareto-optimal solutions in its final population. Moreover, EA's are also less susceptible to the shape of Pareto-front (concave or convex).

This ability of an EA to find multiple optimal solutions in one single simulation run makes EA's unique in solving MOP's. As we mentioned the basic schematics of an ideal multi-objective optimizer, an EA's population based approach can be suitably utilized to find a number of solutions in a single run. We assume that the readers have background knowledge on basic EA mechanisms and entities (i.e. individual, population, genetic operator, selection mechanism and “basic building block hypothesis/schema theorem”), we are not going to discuss the details of EA. For the interested readers, the basic theories on EA can be found in [34].

2.4 Rise of Multi-objective Evolutionary Algorithms

Early applications of EA’s to multi-objective optimization problems were mainly preference based approaches, although the need for EA’s to find multiple trade-off solutions was clearly stated. There are three methods typically used to solve MOP’s with EA -

- Preference based approach
- Population based approach
- Pareto based approach

We will briefly discuss each of them in following subsections -

2.4.1 Preference Based Approach

As discussed in section 2.2, the preference based approach is the most straight forward techniques to solve MOP’s. In this case, an aggregating function from the k objectives $(f_1, f_2, \ldots, f_k)$ are designed with a set of user defined weights/preferences and EA’s are deployed to solve the following problem -

$$min \quad \sum_{i=1}^{k} w_i f_i \tag{2.13}$$

Where, $w_i \geq 0$ are the weighting coefficients representing the relative importance of the objective function f_i of the problem. It is usually assumed that -

$$\sum_{i=1}^{k} w_i = 1 \tag{2.14}$$

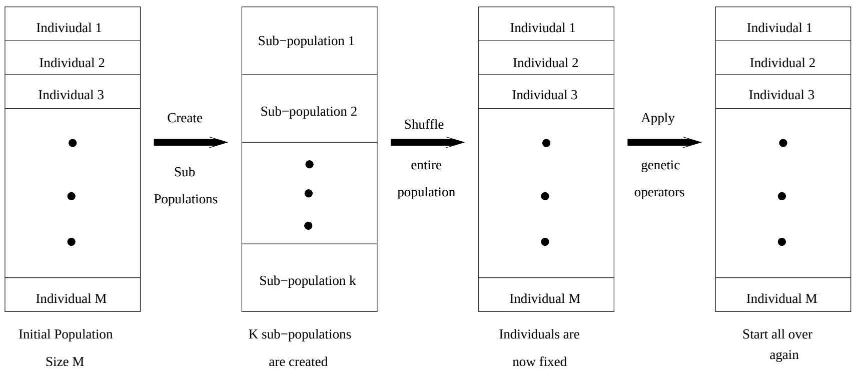


Figure 2.9: Schematic of VEGA's selection mechanism

Aggregating functions may be linear or non-linear [35] [36]. Aggregating functions have been largely underestimated by MOEA researchers mainly because of the limitation of linear aggregating functions (i.e. they can not generate non-convex portions of the Pareto-front regardless of the weight combination used [37]). Note, however, that non-linear aggregating functions do not necessarily present such limitation [2], and they have been quite successful in multi-objective combinatorial optimization [38].

2.4.2 Population Based Approach

In this type of approach, the population of an EA is used to diversify the search, but the concept of Pareto-dominance is not directly incorporated into the selection process. The classical example of this sort of approach is the *Vector Evaluated Genetic Algorithm (VEGA)* [39], [40]. VEGA basically consists of a simple genetic algorithm with a modified selection mechanism. At each generation, a number of sub-populations are generated by performing proportional selection according to each objective in-turn. Thus for a problem with k objectives, k sub-populations of size M/k each are generated (assuming a total population size of M). These sub-population are then shuffled together to obtain a new population of size M, on which the genetic algorithm applies the crossover and mutation operators. However VEGA has several problems, from which the most serious is that its selection scheme is opposed to the concept of Pareto-dominance. The basic schematic of a population based MOEA is depicted in Figure 2.9.

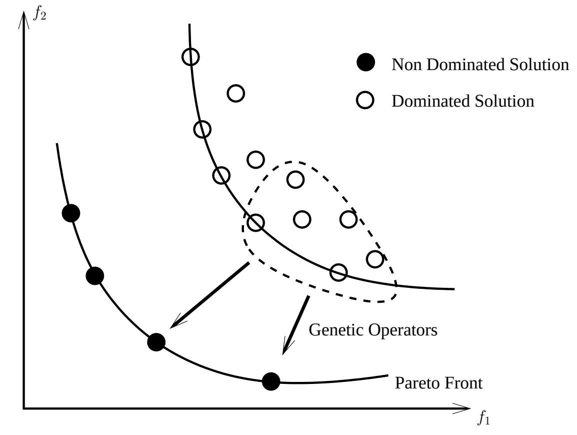


Figure 2.10: A typical non-domination based MOEA

2.4.3 Pareto Based Approach

Under this category, we consider MOEA's that incorporate the concept of Pareto-optimality in their selection mechanism. A wide variety of Pareto-based MOEA's have been proposed in the last few years and it is not the intent of this section to provide a comprehensive survey of them since a review is available elsewhere [2], [1]. In contrast, this section provides a brief discussion of a relatively small set of Pareto-based MOEA's that are representative of the research being conducted in this area.

A typical non-dominated sorting MOEA employing an elitist model has the following functionality: Firstly, the algorithm starts with a randomly generated population P_t, and then after evaluation, the individuals are sorted according to the non-domination criteria and divided into ϕ fronts.

$$P_t := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1\} \tag{2.15}$$

Then, from the best front $\mathcal{F}_\phi$, mutation, crossover and other genetic operators are applied to expand the population to the next best front $\phi + 1$ to create the next population P_{t+1}. Different algorithms uses different techniques to expand this population to the next front.

Generally, most of the Pareto-based algorithms follow the similar architecture as stated above. Now, we briefly describe the popular algorithms that use this approach.

Goldberg's Pareto Ranking

Goldberg suggested moving the population toward PF_{true} by using a selection mechanism that favors solutions that are non-dominated with respect to the current population [34]. He also suggested the use of fitness sharing and niching as a diversity maintenance mechanism [41].

Multi-Objective Genetic Algorithm (MOGA)

Fonseca and Fleming [22] proposed a ranking approach different from Goldberg's scheme. In this case, each individual in the population is ranked based on how many other points dominate them. All the non-dominated individuals in the population are assigned the same rank and obtain the same fitness, so that they all have the same probability of being selected. MOGA uses a niche-formation method in order to diversify the population, and a relatively simple methodology is proposed to compute the similarity threshold (called σ_{share}) required to determine the radius of each niche.

Non-dominated Sorting Genetic Algorithm (NSGA)

NSGA [24] is based on several layers of classifications of the individuals as suggested by Goldberg [34]. Before selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category with a dummy fitness value, which is proportional to the population size, to provide an equally productive potential for this individuals. To maintain the population diversity, the classified individuals are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of non-dominated individuals is considered. The process continues until all individuals in the population are classified. Stochastic remainder proportionate selection is adopted for this technique. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. An enhanced extended version of this algorithm, NSGA-II [42], uses elitism and crowded comparison operator that ranks the population based on both Pareto-dominance and region density. This crowded comparison operator makes the NSGA-II considerably faster than its predecessor while producing very good results. Due its immense popularity, we are going to compare our algorithm with NSGA-II.

Niched Pareto Genetic Algorithm (NPGA)

This method employs an interesting form of tournament selection called Pareto-domination tournaments. Two members of the population are chosen at random and they are each compared to a subset of the population. If one is non-dominated and the other is not, then the non-dominated one is selected. If there is a tie (both are either dominated or non-dominated), then fitness sharing decides the tourney results [23].

Strength Pareto Evolutionary Algorithm (SPEA)

This method attempts to integrate different MOEAs [25]. The algorithm uses a "strength" value that is computed in a similar way to the MOGA ranking system. Each member of population is assigned a fitness value according to the strengths of all non-dominated solutions that dominate it. Diversity is maintained through the use of a clustering technique called the "Average Linkage Method".

A revision of this method called SPEA-II [43], adjusts the fitness strategy slightly and uses nearest neighbor techniques for clustering. In addition, archiving mechanism enhancements allow for the preservation of the boundary solutions that are missed with SPEA.

Pareto Archived Evolution Strategy (PAES)

PAES [44], [45] uses a $(1 + 1)$ evolution strategy, where each parent generates one offspring through mutation. The method uses an archive of non-dominated solutions to compare with individuals in the current population. For diversity, the algorithm generates a grid overlaid on the search space and counts the number of solutions in each grid. A disadvantage of this method is its performance on disconnected Pareto-fronts.

2.5 Conclusion

In this chapter we discussed the basic theory behind MOP's. We also identified that an aggregate function approach does not meet the criteria for ideal multi-objective optimization algorithm, whereas, Pareto-based approach meets the exact requirements. For this reason, MOEA's are most widely used in the field. Even though, MOEA's are good at finding a set of Pareto-optimal solutions, they still suffer from slow convergence and their performance degrades as the complexity of the problems increases (i.e. increasing number of design variables, objective functions and complex shape of Pareto-front). So, the concept of **"Approximation/Surrogate/Meta-model/Intelligent Operator"** based techniques come to the scene to help speeding up the convergence rate. In the next chapter we illustrate different kind of "speeding up" techniques generally used in the MOEA paradigm.

Chapter 3

Acceleration Models for Evolutionary Algorithms

3.1 Introduction

Most numerical optimization problems require experiments and/or simulations to evaluate design objectives and constraints. For many real world problems, however, a single simulation can take a long time. As a result, routine tasks such as design optimization, design space exploration, sensitivity analysis and what-if analysis become impossible since they require thousands or even millions of simulations [5].

One way of alleviating this burden is to construct approximation models, known as **Surrogate Models**, **Response Surface Models**, **Meta Models** or **Emulators**, that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate. Surrogate models are typically constructed using a data-driven, bottom-up approach [5]. The Response Surface Methodology (RSM) [15], on the other hand, explores the relationships between several explanatory variables and one or more response variables. In this chapter we review the aspects of these kind of approximation models. Generally, in the MOEA domain, approximation can be adopted in different ways, not only as Surrogate or Response Surface models, approximation can also be achieved by **"Intelligent Operator"**[4] based models. Here, the **"Intelligent Operator"** utilizes the search path in such a way that the original evolutionary algorithm can reach the global optimum in smaller computational effort. In this chapter, we discuss these

[4]Most instances of *"Intelligent Operator"* uses the directional information from the previous search path or identify the relationship between the objective values and design variables, so that the original EA can avoid the redundant creation of infeasible solutions

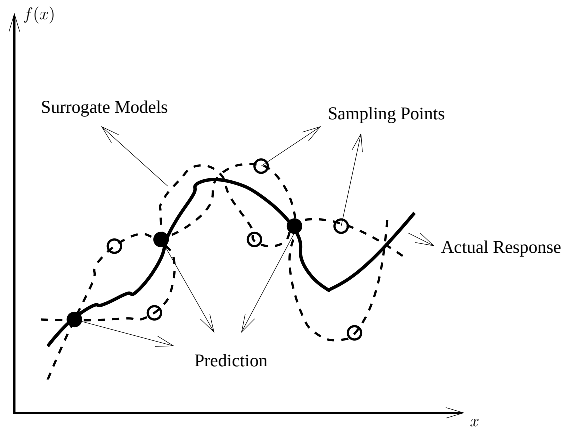


Figure 3.1: A polynomial surrogate to model the behavior of a Design of Experiments Simulation

kind of models in detail along with exhaustive literature review.

The main idea of Surrogate/RSM is to use a set of designed experiments to obtain an optimal response. In this case, the exact, inner working of the simulation code is not assumed to be known (or even understood), solely the input-output behavior is important. A model is constructed based on modeling the response of the simulator to a limited number of intelligently chosen data points. This approach is also known as **Behavioral Modeling** or **Black-box Modeling** [46], though the terminology is not always consistent [47]. When only a single design variable is involved, the process is known as curve fitting as illustrated in Figure 3.1.

An important distinction can be made between two different applications of surrogate models. The first involves building small and simple surrogates for use in optimization. Simple surrogates are used to guide the search towards a global optimum. Once the optimum is found the surrogates are discarded [48]. In the second case, one is not interested in finding the optimal parameter vector, but the interest is in determining the global behavior of the system. In such case, the surrogate is tuned to mimic the underlying model over the complete design space. Such surrogates are a useful and a cheap way to gain insight into the global behavior of the system. Optimization can still can be performed as a post processing step [49].

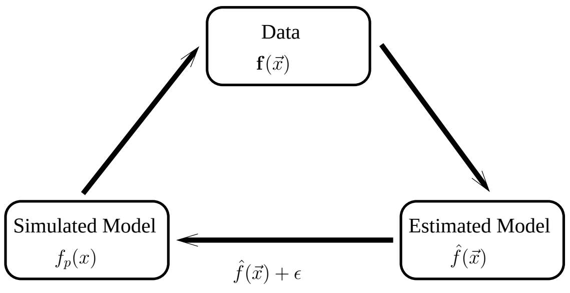


Figure 3.2: A basic flow chart for a surrogate based optimization

The challenge of surrogate modeling is the generation of an approximator that is as accurate as possible, using as little simulation evaluations as possible. The process comprises three major steps, which may be interleaved iteratively:

1. Sample selection (also known as sequential design, optimal experimental design (OED) or active learning)
2. Construction of the surrogate model and optimizing the model parameters (Bias-Variance trade-off)
3. Validation of the accuracy of the surrogate

Surrogate modeling can be thus seen as a non-linear inverse problem with an aim to determine a continuous function $f(\vec{x})$ of a set of design variables from a limited amount of available data $\mathbf{f}(\vec{x})$. The available data '$\mathbf{f}$' while deterministic in nature can represent exact evaluations of the function $f(\vec{x})$ or noisy observations and in general cannot carry sufficient information to uniquely identify $f(\vec{x})$ (multiple surrogates may be consistent with the available data as illustrated in Figure 3.1). Thus, surrogate modeling deals with the twin problems of: (a) constructing a model $\hat{f}$ from the available data $\mathbf{f}(\vec{x})$ (model estimation), and (b) assessing the errors 'ϵ' attached to it (model validation). This idea will be apparent from the flowchart presented in Figure 3.2. A general description of the anatomy of inverse problems can be found in [50].

When using the surrogate modeling approach, the prediction of the simulation-based model output is formulated as $f_p(\vec{x}) = \hat{f}(\vec{x}) + \epsilon(\vec{x})$: The predicted expected value and its variance $V(f_p)$ are illustrated in Figure 3.3, with θ being a probability density function. Note that in Figure 3.1 it is assumed that the expected value of $\epsilon(\vec{x})$ is zero.

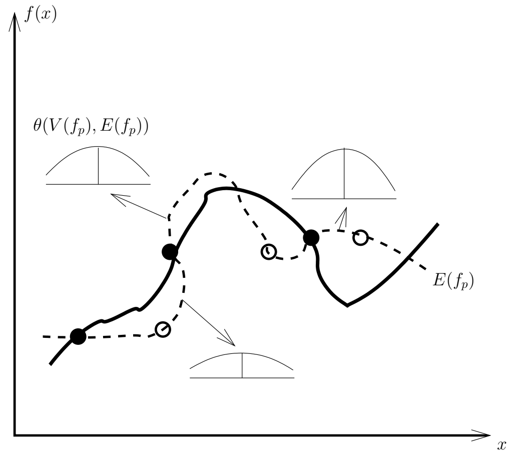


Figure 3.3: The prediction expected value $E(f_p)$ and its variance $V(f_p)$

Different model estimation and model appraisal components of the prediction have been shown to be effective in the context of surrogate based analysis. In Table 3.1, we summarize the different approximation techniques which are described in the following sections.

Type	Implementation	Reference
Numerical Models	Quadratic Approximation	[51]
	Polynomial Regression	[52], [53]
	Stepwise Regression	[54]
	Gradient	[55], [56], [57]
	Directional Information	[58], [59]
Radial Basis Function Models	RBF Surrogate	[60], [61]
	Local & Global RBF	[11]
	Max-min RBF	[12]
	RBF & Rough Set	[62]
Artificial Neural Network Models	Informed Convergence Accelerator	[63]
	Inverse Mapping	[64]
	General ANN	[65], [66], [7], [67]
Kriging/Response Surface Models	General Kriging	[68]
	Co-Kriging	[69]
	Gaussian Random Field(GRF)	[70], [52], [71], [72]
	Efficient Global Optimization (EGO)	[73]
	Response Surface	[74], [14]
Intelligent Operator Models	Directed Variation	[75]
	Guided Mutation	[76]
	Extrapolation Directed Crossover	[77]
Other Models	Principal Component Analysis	[78], [79]
	Weighting	[80] [81]

Table 3.1: Summary of different approaches for high speed convergence techniques

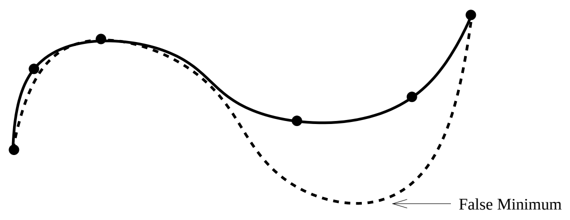


Figure 3.4: An example of a false minimum in the approximate model. Solid line denotes the original fitness function, dashes line the approximate model and the dots the available samples

3.2 Different Surrogate Models Applied to EA Domain

The rising trend of using time-consuming simulation in scientific and engineering works has restricted the use of EA's as a global optimization tool. To address this problem, it has been a standard practice to use a computationally cheap approximation or surrogate models in lieu of an exact model. In early research [82], [47], a function approximation based surrogate model was introduced to Nonlinear Programming (NLP) methods. As its applicability has been proved in general numerical optimization domain, naturally it has become one of the central interests in the evolutionary computation community [5], [83]. We also find some rigorous survey on meta-model assisted evolutionary search techniques in [84], [85], [86], [4], [87].

The application of approximation models to evolutionary computation is not straightforward. There are two major concerns when using approximate models for the fitness evaluation. First, it is necessary that the EA converges to the global optimum or a near-optimum of the original fitness function. Second, the computational cost should be reduced as much as possible. One essential point is that it is very difficult to construct an approximate model that is globally correct due to the high dimensionality, ill distribution and limited number of training samples. It is found that if an approximate model is used for fitness evaluation, it is very likely that the EA will converge to a false optimum. A false optimum is an optimum of the approximate model, which is not one of the original fitness function as shown in Figure 3.4 [84].

Therefore, in most cases, it is essential that the approximate model should be used together with the original fitness function. This can be regarded as the issue of model management or evolution control. By evolution control, it is meant that in evolutionary computation using approximate models, the original fitness function is used to evaluate some of the individuals or all individuals in some generations [88]. An individual that is evaluated using the original fitness function is called as the *Controlled Individual*. Similarly, a generation in which all its individual are evaluated using the original fitness function is called as the *Controlled Generation* [84].

Generally, model management in evolutionary computation can be divided into three main approaches from the viewpoint of evolution control -

- **No Evolution Control**. Very often, the approximate model is assumed to be of high-fidelity and therefore, the original fitness function is not used in evolutionary computation.
- **Fixed Evolution Control**. The importance of using both the approximate model and the original function for fitness evaluation has been recognized [89], [88], [83]. There are generally two approaches to evolution control, one is individual-based and the other is generation-based.
- **Adaptive Evolution Control**. It is straightforward to imagine that the frequency of evolution control should depend on the fidelity of the approximate model. A method to adjust the frequency of evolution control based on the trust region framework [90], [91] has been suggested in [60], in which the generation-based approach is used. A framework for approximate model management has also been suggested in [92].

The above mentioned approach to control the evolutionary process is illustrated in Figures 3.5, 3.6 and 3.7

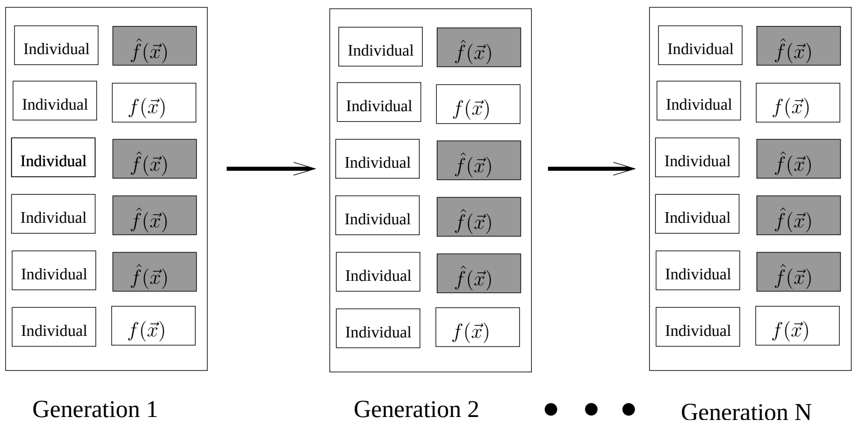


Figure 3.5: The best individual is controlled in each generation. $\hat{f}(\vec{x})$: approximate model; $f(\vec{x})$: original function

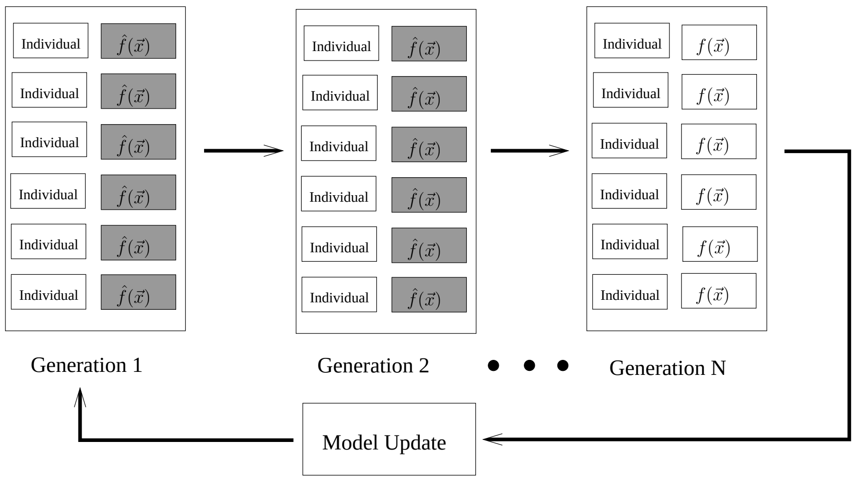


Figure 3.6: Generation-based evolution control. $\hat{f}(\vec{x})$: approximate model; $f(\vec{x})$: original function

With the advent of surrogate models in single-objective optimization problems, its application to multi-objective problems has also became an interesting research topic in MOEA. However in most of the cases, implementation strategies were very similar to that of single-objective approaches.

Since in the case of MOP's, the goal is to find a set of compromising solutions that satisfy two or more objectives, the implementation of surrogate models are not as straight forward as in single-objective EA's. In the literature, we find different approaches to handle this problem. The most trivial approach to handle this problem, is to design the

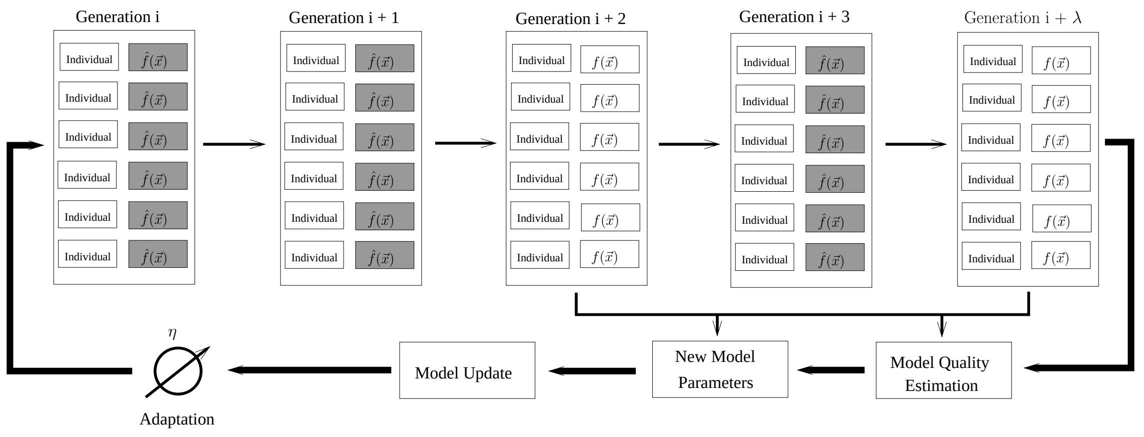


Figure 3.7: Adaptive generation-based evolution control. In the evolution control cycle, there are λ generations, η ($\eta \leq \lambda$) generations will be controlled. $\hat{f}(\vec{x})$: approximate model; $f(\vec{x})$: original function

approximation functions using direct implementation of surrogate models as in single-objective EA's. In this case, we must construct a number of surrogate models, one for each of the objective functions [16]. Another approach is to decompose the multiple objectives into several scalar optimization problems [72] and thus construct the surrogate model. In recent work [93], a multi-level surrogate model was presented, where a multiple approximation scheme was adopted in a two levels of optimization procedure. The first level deals with RBF network for global optimization and the second level uses a gradient method to perform local optimization. A detailed survey on surrogate models in MOEA is presented in [17].

Based on a thorough survey, we found a number of limitations of conventional surrogate models for MOEA's -

- Since MOP's deal with more than one objective, we have to design different approximation models for different objective functions. If we are going to use a computationally expensive method like Kriging/RBF network/Artificial Neural Network[5], the learning cost increases proportionally with the number of objective functions.
- The main goal of an MOEA is to find all possible solutions in the Pareto-front. However, the shape of Pareto-front may effect on the performance of the surrogate.

[5]These techniques will be described in coming sections

Surrogate models are designed for continuous functions, so when a broken (not connected) Pareto-front is encountered, the model may not work properly.

- Surrogate models are also error-prone to deceptive phenomenon and trapping into the false optima in single objective optimization [85], [84]. The same problem exists for MOP's.

In the following sections, we will discuss the implementation of surrogate/approximation models in EA's and MOEA's briefly.

3.3 Polynomial Regression Models

The most widely used polynomial approximation model is the second-order model which has the following form:

$$\hat{f}(\vec{x}) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j \leq i}^{n} \beta_{ij} x_i x_j \tag{3.1}$$

where β_0 and β_i are the coefficients to be estimated, and the total number of terms in the quadratic model is $n_t = (n+1)(n+2)/2$, where n is the number of input variables. To estimate the unknown coefficients of the polynomial model, both the **Least Square Method (LSM)** and **Gradient Method** can be used:

3.3.1 Least Square Method

To generate a unique estimation of the coefficients using LSM, the number of samples (N) drawn from the original function should be equal to or larger than the number of coefficients n_t. Let

$$\vec{f}(\vec{x}) = \left[\hat{f}^1(\vec{x}), \hat{f}^2(\vec{x}) \ldots \hat{f}^N(\vec{x})\right]^T \tag{3.2}$$

and

$$\mathbf{X} = \begin{bmatrix} 1 & {x_1}^1 & {x_2}^1 & \ldots & ({x_n}^1)^2 \\ 1 & {x_1}^2 & {x_2}^2 & \ldots & ({x_n}^2)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & {x_1}^N & {x_2}^N & \ldots & ({x_n}^N)^2 \end{bmatrix} \tag{3.3}$$

then this equation holds:

$$\mathbf{f}(\mathbf{x}) = \mathbf{X}\boldsymbol{\Theta} \tag{3.4}$$

where $\boldsymbol{\Theta}$ is the parameters of equation 3.1 and approximated $\hat{\boldsymbol{\Theta}}$ can be calculated using LMS method -

$$\hat{\boldsymbol{\Theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{f}(\mathbf{x}) \tag{3.5}$$

here, equation 3.5 assumes that the rows of $\mathbf{X}$ are linearly independent. Details can be found in [48].

3.3.2 Gradient Method

The main drawback of the least square method is that the computational expense becomes unacceptable as the dimensionality increases. To address this problem, the gradient method can be used. This approach defines the following square error function for the sample k.

$$E^k = 0.5(f(\vec{x}) - \hat{f}^k(\vec{x}))^2 \tag{3.6}$$

where $\hat{f}(\cdot)$ is defined in equation 3.1 and it is straightforward to get the update rule for the unknown coefficients β_0, β_i and β_{ij}.

$$\Delta\beta_0 = -\xi \cdot (f(\vec{x}) - \hat{f}^k(\vec{x})) \tag{3.7}$$

$$\Delta\beta_i = -\xi \cdot (f(\vec{x}) - \hat{f}^k(\vec{x}))x_i^k \tag{3.8}$$

$$\Delta\beta_{ij} = -\xi \cdot (f(\vec{x}) - \hat{f}^k(\vec{x}))x_i^k x_j^k, \quad 1 \le i \le j \le n \tag{3.9}$$

The detailed implementation of this approach can be found in [94].

3.3.3 Polynomial Regression Models for Single-Objective Evolutionary Algorithms

Generally, there are numerous ways to implement a surrogate model for a specific numerical optimization problem, such as numerical function approximation, correlation analysis, gradient based computation. For example, in [13], the model was applied using correlation analysis, in [94] the proposed model uses gradient based response surface approximation and in [95], the authors propose a coupled model that uses both gradient based surrogate and EA to optimize Learning Classifier System (LCS). In [52], the surrogate was designed using both polynomial regression model and gaussian random field based approximation methods. In [53], the proposed model also uses linear estimation methods for reducing the number of expensive evaluations.

In the case of numerical approximation methods, the model is generally built from the previous search path of the algorithm. When using an EA, the search path is constructed using the previously visited points in the search space (best individuals found so far). Once the model is built, then the next possible best solutions are approximated. Methods, such as gradient or polynomial or least square models works reasonably well for problems with smaller numbers of design variables. When the number of design variable increases, the performance of such methods tends to be degraded.

3.3.4 Polynomial Regression Models for Multi-Objective Evolutionary Algorithms

Similar to single-objective EA's, in [6] a fitness estimation based surrogate model was used. Here, a locally weighted regression (using QR factorization) was used to model the surrogate. The proposed algorithm was applied to an internal-combustion engine controller optimization problem. In other work [51], the surrogate model was built from the Quadratic Approximation (QA) function. The study reports on the use of a number of independent distributed EA's run concurrently where each of them was used to optimize one objective function. Along with this scheme, they have also used their own **"Informed Operator"** [96] to generate new offsprings.

In other work [54], the response surface method (similar to Kriging, that will be discussed in section 3.6) with linear and quadratic basis functions was employed to formulate the objectives, in which optimal Latin hypercube sampling and stepwise regression techniques were implemented.

Gradient based methods are also popular. In [55], the authors proposed the concept of **"Pareto-Descent Method (PDM)"**. Where **"Pareto-Descent Direction"** is explained as

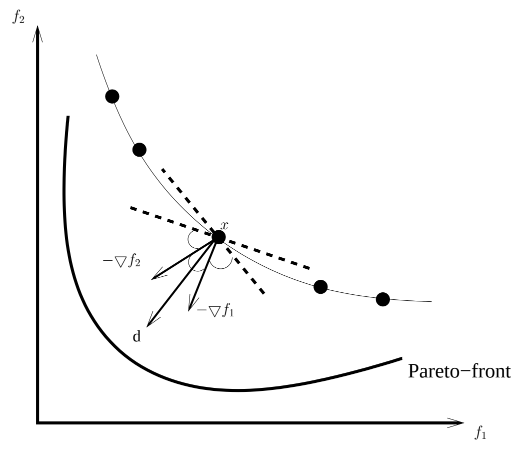


Figure 3.8: Pareto-descent directions d: at a solution x of a 2-variable-2-objective problem

the descent directions to which no other descent directions are superior in improving all objective functions. The work proposed a new Local Search (LS) method which finds Pareto descent directions and moves solutions in those directions thereby improving all objective functions simultaneously. The concept is illustrated in Figure 3.8. If a part or all of them are infeasible, it finds feasible Pareto descent directions or descent directions as appropriate. PDM finds these directions by simple linear programming. Experiments have shown PDMs superiority over other existing methods. Similar approaches are also described in [56], [57].

In [58], the authors developed a local framework for MOP by geometrically analyzing the multi-objective concepts of descent, diversity and convergence/optimality. The study also showed that locally optimal, multi-objective descent direction can be calculated in such a way that it maximally reduces all the objectives (ensuring both diversity and convergence). Their concept was to extend the idea of "Half-space" from single-objective problems to MOP. More details on the theoretical analysis on "Half-space" and its extension for MOP's can be found in [59]. In [97], directional information is also utilized to speed up the convergence rate of "Non-dominated Sorting Genetic Algorithm (NSGA-II)" [42]. NSGA-II is considered as one of the most popular algorithms in MOEA. The authors have tested their models on several benchmark problems and it is also reported that their model showed promising improvement with respect to NSGA-II.

3.4 Radial Basis Function (RBF) Models

Radial basis functions have been developed for the interpolation of scattered multivariate data. The method uses linear combinations of n radially symmetric functions, $h_i(\vec{x})$; based on Euclidean distance or other such metric, to approximate the response functions as

$$f_p(\vec{x}) = \sum_{i=1}^{n} w_i h_i(\vec{x}) + \epsilon_i \tag{3.10}$$

where $\vec{w}$ represents the coefficients of the linear combinations, $h_i(\vec{x})$ the radial basis functions and ϵ_i independent errors with variance σ^2.

Radial basis functions are a special class of functions with their main feature being that their response decreases (or increases) monotonically with distance from a central point. The center, the distance scale, and the precise shape of the radial function are the parameters of the model.

A typical radial function is the Gaussian which, in the case of a scalar input, is expressed as

$$h_i(\vec{x}) = \exp\left(\frac{(\vec{x} - \vec{c})^2}{\delta^2}\right) \tag{3.11}$$

The parameters are its center $\vec{c}$ and its radius δ (See Figure 3.9). Note that the response of the Gaussian radial basis function decreases monotonically with the distance from the center, giving a significant response only in the center neighborhood. Given a set of N input/output pairs (sample data) a radial basis function model can be expressed in matrix form as,

$$\mathbf{f} = \mathbf{Hw} \tag{3.12}$$

where $\mathbf{H}$ is a matrix given by,

$$\mathbf{H} = \begin{bmatrix} h_1(x^1) & h_2(x^1) & \dots & h_n(x^1) \\ h_1(x^2) & h_2(x^2) & \dots & h_n(x^2) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x^N) & h_2(x^N) & \dots & h_n(x^N) \end{bmatrix} \tag{3.13}$$

Similar to the polynomial regression method, by solving equation 3.12 the optimal weights

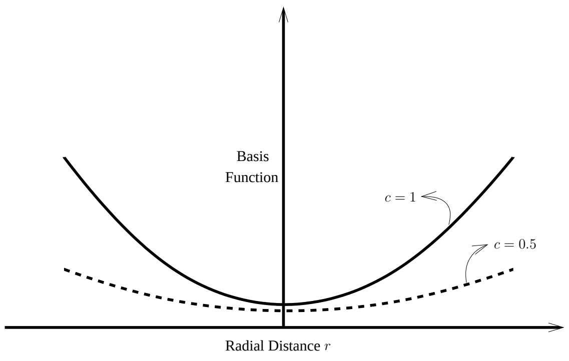


Figure 3.9: A multiquadric radial basis function

(in the least squares sense) can be found to be

$$\hat{\mathbf{w}} = \mathbf{A}^{-1}\mathbf{H}^T\mathbf{f} \tag{3.14}$$

where A^{-1} is a matrix given by,

$$\mathbf{A}^{-1} = (\mathbf{H}^T\mathbf{H})^{-1} \tag{3.15}$$

The RBF model estimate for a new set input values is given by,

$$\hat{f}(\mathbf{x}) = \mathbf{h}^T\hat{\mathbf{w}} \tag{3.16}$$

where, **h** is a column vector with the radial basis functions evaluations,

$$\mathbf{h} = [h_1(\mathbf{x}), h_2(\mathbf{x}) \ldots, h_n(\mathbf{x})]^T \tag{3.17}$$

The function represented in equation 3.11 is known as Radial Basis Kernel. Typical choices for the kernel include linear splines, cubic splines, multi-quadrics, thin-plate splines, and Gaussian functions [98]. The structure of some commonly used radial basis kernels and their parameterizations are shown in Table 3.2. Given a suitable kernel, the weight vector can be computed by solving the linear algebraic system of equations as described in equation 3.14. [99] provides the detailed approximation of this approach.

Type	Definition
Linear Splines	$\|\|r\|\|$
Thin Plate Splines	$\|\|r\|\|^{2m+1}\ln\|\|r\|\|$
Cubic Splines	$\|\|r\|\|^3$
Multiquadric Splines	$\sqrt{1+\epsilon\|\|r\|\|^2}$
Gaussian Splines	$e^{-(\epsilon\|\|r\|\|)^2}$

Table 3.2: Radial Basis Functions, Where $r = (\vec{x} - \vec{c})^2/\delta^2$

3.4.1 RBF Models for Single-Objective Evolutionary Algorithms

To construct the RBF based meta-model for single-objective EA's, we have to consider the previously visited design points as its learning parameters [60], [100]. This is similar to the numerical approach (polynomial regression, gradient based approach) in the sense that it also uses previous experience of the hosting search algorithms as the building block. As an example, let us consider the problem of optimizing an expensive function $f(\vec{x})$, which is defined as

$$\text{Minimize} \quad f(\vec{x}) \tag{3.18}$$

$$\text{Subject to} \quad x_l \leq x_i \leq x_m \tag{3.19}$$

A conventional EA first starts by creating a set of random $\vec{x}$ and uses genetic operators (crossover, mutation) and selection to generate new design variables. Once a good design variable is found, the algorithm stores it in an external population for modeling the surrogate. Once there are enough visited points in the external population, surrogate models will be created from those points. To illustrate this more clearly, the points stored in the external population are considered as $\vec{x}$ in equation 3.11. This equation has a special property, that is, its response $h_i(\vec{x})$ is monotonically increasing (or decreasing) with respect to the distance of $\vec{x}$ from the central point c (Figure 3.9). This property will help to approximate the next best design variable $\vec{x}$ from the existing points without re-evaluating the newly generated population using the expensive objective function $f(\vec{x})$. RBF based surrogate models usually exhibits better approximation of the search path when combined with the hosting optimization algorithm than other general numerical

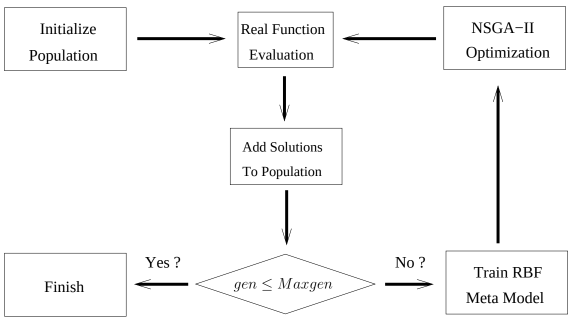


Figure 3.10: RBF algorithm, which uses the NSGA-II to optimize the meta-model

approach (i.e. polynomial, quadratic or gradient based surrogate models). The improvement is typically greater for optimizing larger number of design variables [11], [12], [100].

3.4.2 RBF Models for Multi-Objective Evolutionary Algorithms

In the case of MOEA's, the approach is similar to single-objective approaches. One can design RBF networks for individual objective functions for MOEA population or different models for different sub-populations. We have found several methods describing implementation of RBF networks for MOEA's. In [101], the population was divided into several clusters and each of the clusters is used to design different RBF network models. The authors compared their model with NSGA-II on five benchmark MOP's and significant improvement was reported.

In [62], the model was also integrated with NSGA-II to enhance its convergence speed. In that paper, it was reported that the spread of solutions in the Pareto-front was not always satisfactory, if a RBF network was only used. Consequently, the authors integrated a "Rough Set" based approach for enhancement. Moreover, in the MOP domain, convergence is not the only criteria to be achieved. To find all satisfactory solutions, we must generate the Pareto-front with a "good spread" [1], [2]. Therefore, the authors used the notion of "Rough-Set Theory" to ensure this criteria. They tested their model on several popular benchmark MOP's and found promising results. The model used in [62] is presented in Figure 3.10. More details on the use of RBF networks for MOEA's are provided in [61].

3.5 Artificial Neural Network (ANN) Models

Neural networks have been shown to be effective tools for function approximation. Given their good approximation and adaptive ability, they have been widely used in the field of dynamic system identification and automatic control [102], [103]. However, when they are applied in numerical optimization problems as a surrogate model, the ANN is used in the sense that the hosting optimization algorithm is also considered as a "dynamic system" that takes some real values (design variables) as input signal and generates some real values (objective functions) as output response. In the case of ANN, both Feed-Forward Multi-Layer Perceptrons (MLP) and Radial Basis Function (RBF) networks have widely been used. A MLP with one input layer, two hidden layers and one output neuron can be described by the following equation (see Figure 3.11):

$$f(\vec{x}) = \sum_{k=1}^{K} v_k \varphi \left(\sum_{j=1}^{M} w_{jk} \varphi \left(\sum_{i=1}^{N} w_{ij} x_i \right) \right) \tag{3.20}$$

where, n is the input number, K and L are the number of hidden nodes, and $\varphi(\cdot)$ is called activation function, which usually is the logistic function

$$\varphi(z) = \frac{1}{1 + e^{-az}} \tag{3.21}$$

Figure 3.11 shows a typical ANN architecture. More details on ANN as a surrogate model, can be referred from [104], [105], [106], [66].

3.5.1 ANN Models for Single-Objective Evolutionary Algorithms

ANN based models are one of the most popular approximation techniques implemented in EA paradigm [104], [66], [107]. In this case, first, an ANN with random weights replaces the expensive original functions. Then, the EA starts its optimization with a set of randomly generated individuals. After the evaluation of the individuals by the original functions. The ANN is trained with each of the individuals and their objective values. It is trivial that the design variables of the individuals are considered as input to the ANN and the weight of the ANN are adjusted according to their respective objective functions.

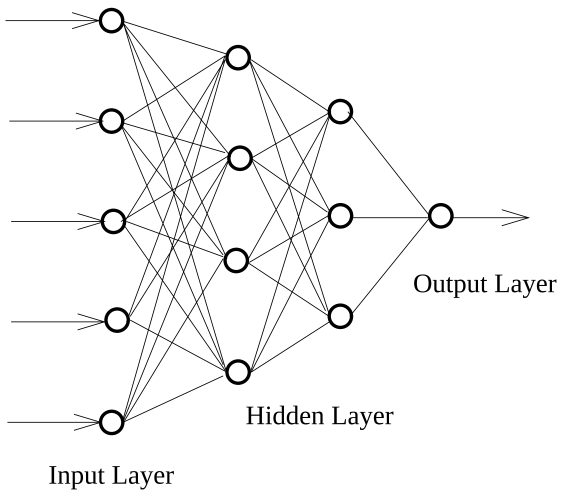


Figure 3.11: A simple artificial neural network

Different algorithms uses different schemes to replace the original expensive function with the trained ANN.

Although ANN are popular models for speeding up the convergence rate of single objective optimizers [63], [64], [65], they have also some shortcomings as stated below -

- Building and training an ANN for a specific input/output pattern of a dynamic system itself is a very difficult problem. Subsequently, determining a good ANN model (as a surrogate) may not lead to an efficient approximation model for most problems. For more insight to this issue is presented in [108], [109].
- The prediction accuracy and adaptability of a typical ANN inherently depends on its architecture, i.e. number of hidden layers along with their complex interconnections. So it is very hard to construct an accurate ANN that replaces the original expensive function. To handle this situation, in [110], we find a Genetic Algorithm/Genetic Programming based approach to optimize the structure and weight of an ANN for particular surrogate model.
- Training an ANN is also a computationally expensive task, so if the training cost becomes larger than the function evaluation, the motivation of ANN as surrogate is demised.

3.5.2 ANN Models for Multi-Objective Evolutionary Algorithms

The early work on ANN based approximation models for MOEA can be found in [7]. For local optimization, an iterative Gaussian Random Field based model was also used. The paper reports insights into general guidelines on how to use ANN based surrogate models for MOEA's.

In [67], [111], the author presented a generic procedure to combine a evolutionary optimization technique with the approximation technique. The focus of that study was to use a "successive approximation" – "Coarse-to-Fine Grain" optimization. Figure 3.12 (Left Side) depicts this procedure. The figure shows a hypothetical one-dimensional objective function for minimization in a solid line. Since the problem may have a number of local minimum solutions, it would be a difficult problem for any optimization technique. Figure 3.12 also shows a coarsely approximated function in the entire range of the function with a dashed line.

The objective functions can be evaluated exactly by a few pre-specified solutions in the entire range of the decision variables. Thereafter, an approximating function was fitted through these function values using an ANN. After the optimization, it is likely that an evolutionary optimizer will proceed in the right direction. Since the population diversity will be reduced while approximating the first approximated function, the second approximating function need not be defined over the whole range of the decision variables, as shown in Figure 3.12. Since the approximating function will be defined over a smaller search region, more local details can appear in successive approximations. This process was continued until no further approximation results in an improvement in the function value. Figure 3.12 (Right Side) outlines a schematic of a plausible plan for the generic procedure. The study also reports that the total number of function evaluation were reduced by 32%. Another similar example can be found in [112].

There is another class of ANN based models termed **"Inverse Neural Network"**[6] [113], [64], [63], [114]. The term **"Inverse Mapping"** actually defines the technique where we reverse map the design variables from the objective functions to the design space. This

[6]The most interesting approach regarding our research methodology. Since our model also performs **"Inverse Mapping"**, however in a slightly different way.

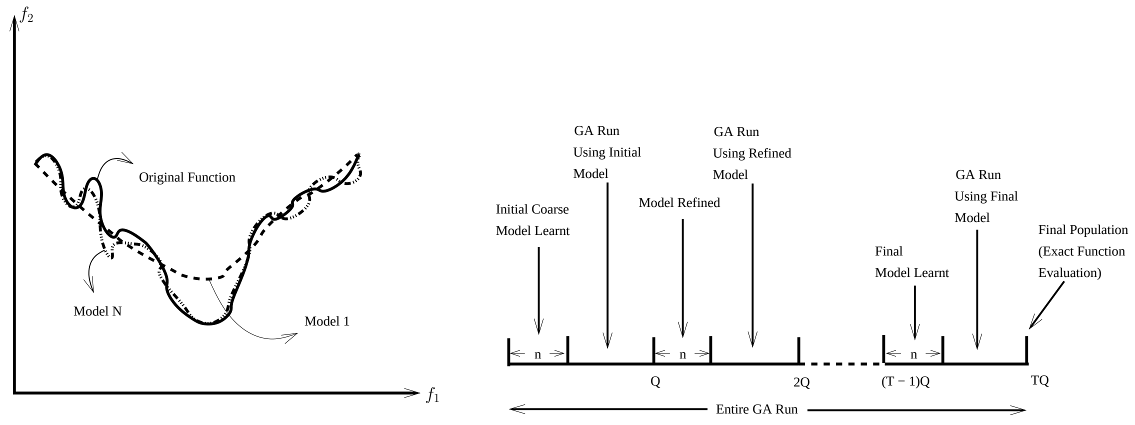

f_2
Original Function
Model N
Model 1
f_1
GA Run Using Initial Model
GA Run Using Refined Model
Initial Coarse Model Learnt
Model Refined
GA Run Using Final Model
Final Model Learnt
Final Population (Exact Function Evaluation)
n
Q
2Q
(T − 1)Q
TQ
Entire GA Run

Figure 3.12: Approximate modeling (Left) and A line diagram of the generic procedure

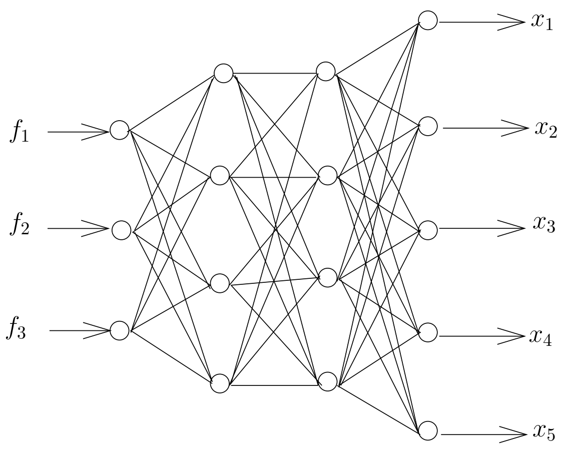


Figure 3.13: An inverse ANN for 3 objectives and 5 design variables problem

idea has a special influence in the case of MOP's since if we could some how extrapolate the next (future) Pareto-front from the existing (present) ones, the **"Reverse Mapping"** technique will help us to approximate those design variables that would make up the future Pareto-front. To implement this idea of **"Reverse Mapping"**, we consider the optimizer as a **"Dynamic System"** that takes **"Design Variables"** as **"Inputs"** and **"Objective Functions"** as **"Outputs"**. Under this assumption, the ANN mimics this **"Dynamic System"** or **"Optimizer"**. The Figure 3.13 illustrates the **"Reverse Mapping"** processing using an ANN. Although, ANN based models have been used successfully in some circumstances, we found some implementation limitations -

- It is very difficult to design the most appropriate ANN structure (both in terms of accurate weights and neuron connections) that will exactly mimic the behavior of original expensive fitness function/evolutionary optimizer (in the case of "Inverse Mapping").

- ANN have a corresponding huge computational cost due to learning. A good ANN may help us to design an efficient approximator, however the improvement has typically a large computational cost.
- Specially for the "Reverse Mapping" procedure, if the total number of objective is smaller than its design variables (the usual situation), it is very difficult to design an efficient ANN model. Given the fact that, the number of output neurons will have to be smaller than the number of input neurons.

3.6 Kriging Models

The "Kriging" model is named after the pioneering work of D.G. Krige and its formal development was first reported in [115]. The Kriging (also known as "Response Surface Methodology" [116], [117]) method in its basic formulation estimates the value of a function (response) at some un-sampled location as the sum of two components: the linear model (e.g., polynomial trend) and a systematic departure representing low (large scale) and high frequency (small scale) variation components, respectively. The systematic departure component represents the fluctuations around the trend, with the basic assumption being that these are correlated and the correlation depends only on the distance between the locations under consideration. More precisely, it is represented by a zero mean, second-order, stationary process (mean and variance constant with a correlation depending on a distance) as described by a correlation model. Hence, these models (Ordinary Kriging) suggest estimating deterministic functions as

$$\hat{f}(\vec{x}) = \mu + \epsilon(\vec{x}), \quad E(\epsilon) = 0, \tag{3.22}$$

$$cov(\epsilon(x^i), \epsilon(x^j)) \neq 0, \quad \forall i, j \tag{3.23}$$

where μ is the mean of the response at sampled design points, and ϵ is the error with zero expected value, and with a correlation structure that is a function of a generalized distance between the sample data points. A possible correlation structure [115] is given by

$$cov(\epsilon(x^i), \epsilon(x^j)) = \sigma^2 \exp\left(\sum_{k=1}^{n} \phi_k (x_k^i - x_k^j)^2\right) \tag{3.24}$$

where n denotes the number of dimensions in the set of design variables x; σ; identifies the standard deviation of the response at sampled design points, and, ϕ_k is a parameter which is a measure of the degree of correlation among the data along the direction k. Specifically, given a set of N input/output pairs $(\mathbf{x}, \mathbf{f})$, the parameters, μ, σ; and ϕ are estimated such that a likelihood function is maximized [115]. Given a probability distribution and the corresponding parameters, the likelihood function is a measure of the probability of the sample data being drawn from it. The model estimated at un-sampled points is -

$$E(\hat{f}(\mathbf{x})) = \bar{\mu} + \mathbf{r}^T R^{-1}(\mathbf{f} - \mathbf{1}\bar{\mu}) \tag{3.25}$$

where the bar above the letters denotes estimates, $\mathbf{r}$ identifies the correlation vector between the set of prediction points and the points used to construct the model, $\mathbf{R}$ is the correlation matrix among the N sample points, and $\mathbf{1}$ denotes an N-vector of ones. On the other hand, the estimation variance at un-sampled design points is given by

$$V(\hat{f}(\mathbf{x})) = \sigma^2 \left[1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r} + \frac{1 - \mathbf{1}^T\mathbf{R}^{-1}\mathbf{r}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}\right] \tag{3.26}$$

The stationary covariance process (also known as "Kriging Kernel") expressed in equation 3.24, is known as "Gaussian Correlation Function" [118] since the construction of the model starts with assumption that the probability distribution of random noise '$\epsilon(\vec{x})$' is also Gaussian. There are several ways to select the "Kriging Kernel"; the most popular kernels are illustrated in Table 3.3. For more insight into theories regarding Kriging models, the readers are referred to [116], [117], [119], [46].

Most systems can be approximated with models of varying degrees of accuracy or "fidelity". Everything else being equal, it would be desirable to work with the most accurate model. But highly accurate models may take days to compute, making optimization dif-

Type	Definition
Linear Correlation	$max(1 - \phi r, 0)$
Exponential Correlation	$e^{-\phi r}$
Gaussian Correlation	$e^{-\phi r^2}$

Table 3.3: Kriging Kernel Functions, Where $r = x^i - x^j$

ficult for almost any algorithm. On the other hand, switching to a very fast, low-fidelity model may entail an intolerable loss of accuracy. Rather than searching for the optimal intermediate degree of accuracy, the best approach is to use both low and high fidelity models. So a more appealing approach, from the conceptual point of view, would be to treat the outputs of the low and high fidelity models as correlated dependent variables in a multivariate surrogate model. In the mathematical geology literature, this approach is known as “Co-Kriging”. Other “Kriging” and “Co-Kriging” based optimization approaches can be found in [120], [121], [49].

On the other hand, Gaussian Random Field (GRF) [122], [123], another well-known approach to surrogate modeling, can be shown to provide identical expressions for the prediction and prediction variance to those provided by Kriging, under the strong assumption that the available data (model responses) is a sample of a multivariate normal distribution [124].

3.6.1 Kriging Models for Single-Objective Evolutionary Algorithms

Generally, the incorporation of a a Kriging based meta-model into a conventional EA is similar to the previous approach as discussed in RBF and ANN based models. The model is built using the initial random population of the EA. After the model construction, the hybrid algorithm replaces the expensive original functions with the Kriging model and the evaluation mechanism can be implemented in various ways.

However, the main computational cost involved in constructing Kriging models occurs in the maximum likelihood estimation phase. Here, a nonlinear optimization technique must be employed to estimate the parameters (μ, σ; and ϕ in equation 3.24) by maximizing a likelihood function. Evaluation of the likelihood function requires factorization of the correlation matrix $\mathbf{R}$ which scales as $\mathbf{O}(n^3)$. For this reason, constructing a Kriging model for cases with more than two thousand points can be computationally very expensive. As an aside, in [125], we find a data parallel approach which is possible to apply Kriging to dataset with tens of thousands of points. "Kriging" based surrogate models and their diverse implementations can be found in [68], [69], [89]. Moreover, in [70], [52] we find some experiments on Gaussian Random Field Metamodel to speed up a conventional EA.

The Kriging based "Efficient Global Optimization (EGO)" model is introduced in [46]. The prediction performance of EGO was reported better than other surrogate models[7]. Moreover, in [73], we find rigorous analysis on the performance of EGO coupled with conventional EA. The algorithm was tested on several benchmark single objective functions and it was proved that the incorporation of EGO in conventional EA increased the performance by 50 - 250 times.

3.6.2 Kriging Models for Multi-Objective Evolutionary Algorithms

When applying Kriging models in MOEA's, the design of an accurate approximator is crucial. However, to achieve this goal, we have to spend a large amount of computational effort [74]. In [14], we see that the model was utilized to generate a good approximation from a roughly estimated samples, the authors termed the approximator a "Pseudo-Response Surface". Under the new framework, the "Pseudo Response Surface" is constructed for each design objective. An important distinguishing feature of the new framework was that the response surfaces for all the design objectives were constructed simultaneously in a mutually dependent fashion, in a way that identifies Pareto regions for the multi-objective problem. Figure 3.14 illustrates this basic idea. This pseudo response surface has the unique property of being highly accurate in Pareto-optimal regions, while it is intentionally allowed to be inaccurate in other regions. In short, the response surface for each design objective is accurate only where it matters. It was reported that the computational cost of construction is dramatically reduced.

[7]Specifically, EGO uses the concept of "Design and Analysis of Computer Experiments (DACE)" model [116], [117].

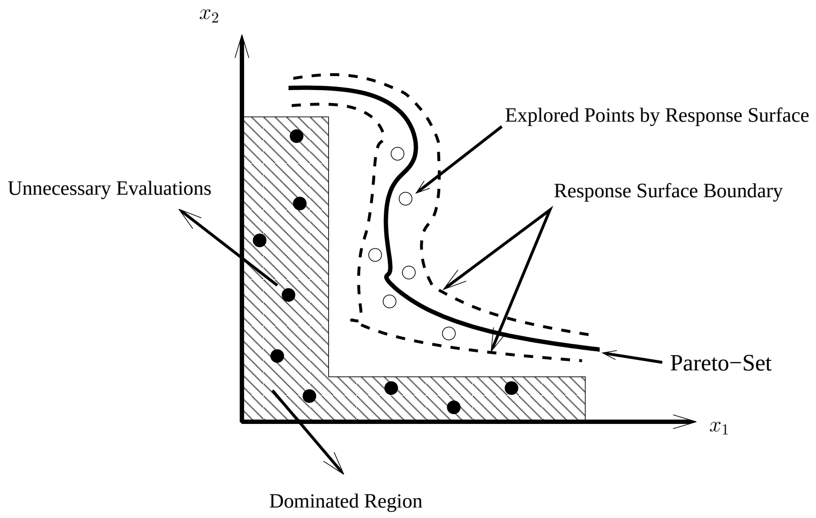

x_2
Explored Points by Response Surface
Unnecessary Evaluations
Response Surface Boundary
Pareto−Set
x_1
Dominated Region

Figure 3.14: A response surface based approximation schematic in design variable space

In [71], the authors proposes a **Gaussian Random Field (GRF)** based approximation model that could predict objective function values for new candidate solutions by exploiting information recorded during previous evaluations. Moreover, GRF are able to provide estimates of the confidence interval of their (candidate solutions') predictions. The model selects the promising members in each generation and carries out exact and costly evaluations only for them. The extensive use of the uncertainty information of predictions for screening the candidate solutions made it possible to significantly reduce the computational cost of single and multi-objective EA's.

We also find a similar approach in [73], where the author proposes an Efficient Global Optimization (EGO)[46] based model to reduce the total number of expensive function evaluations. It was also reported that the proposed model generally outperformed NSGA-II [42] on some hard benchmark functions, at both 100 and 250 function evaluations [126], especially when the worst-case performance was measured. It was also suggested that the model may offer a more effective search on problems where only one function evaluation can be performed at a time. Although its performance was satisfactory on the tested functions, the paper reports some potential drawbacks of the current version.

- Normalization of the cost space [46] relies on knowledge of the cost limits.
- The use of uniformly random scalarizing vectors [46] does not necessarily result in

the best distribution of non-dominated points. Some parts of the Pareto front may be far easier to find than others, which may lead to a poor approximation.

A similar approach is presented in [72], where the performance was analyzed and compared on several test problems, which shows a promising perspective on Kriging. For more insight into this method, the readers are encouraged to refer to [127].

3.7 "Intelligent Operator" Based Models

There are also alternative approaches to speed up the normal convergence rate of an EA. Typically, these kind of models are implemented using some sort of **"Intelligent"** variation operator, that uses the information from the local search space to generate more promising solutions for the next generation of an EA, so that the conventional EA can bypass the extra function evaluations to reach the global optimum with more speed. So, in that sense, these kind of techniques are not generally regarded as "Surrogate" models, rather they are known as **"Informed/Intelligent Genetic Operators"**[8].

Moreover, there is also another class of algorithms that only uses the **"Intelligent Variation Operator"** in place of conventional crossover and mutation, which is known as the **"Estimation of Distribution Algorithm (EDA)"** [128], [73]. Generally, in the case of EDA, it explicitly extracts globally statistical information from the selected solutions and build a posterior probability distribution model of promising solutions, based on the extracted information. A recent algorithm in which we integrate the proposed model in this thesis, RM-MEDA [79], can be considered as a candidate of EDA's. More details on EDA's can be referred from [129], [130].

In [96], the author presents a new concept of **"Informed Operator"**. During mutation, instead of generating one individual from just one random flip; they generate several individuals, rank them using a surrogate model, then take the best to be the result of the mutation. The proposed method is particularly suitable for search spaces with expensive evaluation functions. The authors used the technique to optimize the aerodynamic structure of supersonic missile and found a significant degree of speed up.

[8]This idea has a special relevance to this thesis since our model also exhibits some similarity with this notion of **"Intelligent Operator"**.

The concept of **“Directed Evolution”** is presented in [75], which is taken from the self-organization evolutionary theory. The model was tested with a (μ, λ) Evolutionary Strategy using several benchmark single-objective optimization problems and reported to be achieved good results in terms of total number of function evaluation, i.e. achieving good results with comparative small computational cost. In other work [76], the concept of **“Guided Mutation”** was presented. This new mutation operator was specifically designed for the EDA that uses the probability model from the local search space and creates new offspring such that a good portion of them will always lie in the promising region of the search space. The new scheme was tested on the “Maximum Clique Problem (MCP)” and it was also reported that it shows improvement with respect to conventional EDA’s. In [131], the **Angular Distance Dependant Alteration (ADDA)** model was introduced, where all offspring, generated by crossover operations, will be clustered by corresponding parents based on the angular distance metric and then, the offsprings were transposed from the parent. This operator was used with the multi-parental Unimodal Normal Distribution Crossover (UNDX-m) [132], [133, 134]. The ADDA model shows good performance on some typical benchmark problems.

In [77], the model uses extrapolation directed crossover that uses numerical extrapolation technique to relax the bias created by conventional real-value crossover operators. Sometimes, the conventional crossover operator [132], [135], [136], [137] generates new offspring with the Probability Density Function towards the parents’ distribution in the search space, which is not desirable. It was also reported that extrapolation directed crossover shows significant improvement with respect to the total number of function evaluation in entire EA run.

In [138], the author proposed a new crossover operator that uses linear interpolation on the search path and from this approximated search path, new promising offspring are created. Moreover, Kalman filter based approximation model [139], Rough Set based model [140] and Meme based model [141] can also be found in the literature.

3.8 Other Models

In the case of MOEA's, we also see other related techniques for guiding the solutions towards the Pareto-front such as Scatter Search, Weighting [80], [81] and Bayesian Inference based guiding scheme in MOEA's.

A Principal Component Analysis (PCA) based model was proposed in [78]. The work proposed a model-based evolutionary algorithm (M-MOEA) for bi-objective optimization problems. Inspired by the ideas from EDA's [128], M-MOEA uses a probability model to capture the regularity of the distribution of the Pareto-optimal solutions. The Local Principal Component Analysis (Local PCA) and the least-squares method were employed for building the model. New solutions are sampled from the model. At alternate generations, M-MOEA uses crossover and mutation to produce new solutions. The selection in M-MOEA was the same as in NSGA-II [42]. Therefore, MOEA can be regarded as a combination of EDA and NSGA-II. The preliminary experimental results showed that M-MOEA performs better than NSGA-II.

In [142], the authors proposed a multi-objective evolutionary algorithm based on decomposition (MOEA/D). It decomposes a multi-objective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously. Each subproblem was optimized by using information from its several neighboring subproblems. It was claimed that at each generation MOEA/D has lower computational complexity than NSGA-II. Experimental results indicates that MOEA/D with simple decomposition methods (i.e. Weighted-Sum [143], Tchebycheff [143] and Boundary Intersection approach [27]) outperforms or performs similarly to NSGA-II on multi-objective 01 knapsack problems and continuous multi-objective optimization problems.

In [144], we also find another model that uses Bayesian inference technique to speed up the convergence rate of general EDA's. Other local search and scatter search methods to fulfil the same objective can be found in [145], [146], [147], [148].

3.9 Conclusion

In this chapter, we briefly reviewed different approximation techniques used to speed up the convergence rate of a typical MOEA. However, no clear conclusions related to the advantages and disadvantages of the different approximation models have been drawn[9]. This is reasonable not only because the performance may depend on the problem to be addressed, but also because more than one criterion needs to be considered. The most important factors are accuracy, both on training data and test data, computational complexity and transparency. It has been found in [85] that an approximate model may introduce false optima, although it has very good performance on the training data. This is more harmful than a lower approximation accuracy if the model is used in global optimization procedures such as evolutionary optimization. Methods to prevent an ANN model from generating false minima have been suggested in [85], which are very effective for lower dimensional problems.

In the next chapter we introduce our acceleration model – **Pareto-following Variation Operator** – that utilizes some of the ideas introduced in this chapter.

[9]For more insight on the comparative analysis on different approximation/surrogate/intelligent operator based models, readers are referred to [5], [47], [49], [120].

Chapter 4

The Pareto-Following Variation Operator

4.1 Introduction

In this chapter we describe the core contribution of this thesis – the **Pareto-following Variation Operator (PFVO)**. The most interesting and novel aspect of our model is that it does not model the objective functions directly, rather it works on the basis of the input/output relationship of the underlying MOEA. Here the term **"input/output"** refers to the **"design-variables/objective values"** relationship.

As we have seen before, in the case of a typical MOEA, an initial perturbation of the available design variables (of the current front) generates new (hopefully promising) solutions[10], the algorithm then selects the best solutions with respect to their corresponding objective values and thus expand the search direction towards true Pareto-front.

If we reformulate this idea in a slightly different way, we can assume that the MOEA is a **"dynamic system"** that takes **"design variables"** as **"input"** and generates **"objective values"** as **"output"**. Given that the above assumption is valid, it is possible to identify the parameters of the dynamic system. So, if we could identify the input-output behavior of a dynamic system (the parameters of the dynamic system)[11], we can then predict the design variables (i.e. input) of the next front, given that the guessed objective values (i.e. output) of the next front are provided to the (inverse) dynamic system as input. This notion is the key concept of our algorithm.

[10]The perturbation (crossover/mutation) does not always guarantee the creation of good solutions.

[11]In another sense, the parameters of the stochastic search procedure – More details of this concept will be discussed in coming sections

Typically, ANN based systems are widely used in the field of **Dynamic System Identification (DSI)** [102], [103] to achieve similar goals. However, a significant disadvantage of the ANN approach is the computational cost. In our approach, we extend this idea of modeling the input-output behavior of an stochastic search procedure. We have used **QR Factorization** or **Singular Value Decomposition (SVD)** [149], [150] to implement this idea. However, in the case of MOP's, the applicability of DSI theory is not straight forward since DSI is mainly concerned with the modeling of a physical system [149], [150]. Moreover, the concept of "high level of accuracy" is not crucial in the case of MOEA. Since the approximation of a good Pareto-front from the existing ones needs to be just "fair enough" to bypass the extra computation for evolutionary optimizer [63]. Moreover, to maintain the accuracy of the approximation[12], we are not going to substitute the original fitness function, rather we will use the same objective function and PFVO will try to help the hosting optimizer[13] to reduce the computational cost by skipping function evaluations for non-promising individuals. In the next section, we introduce some useful notations that will be used to describe this model throughout the chapter. After that, we will describe our algorithms in detail along with integration mechanism with different base-line popular MOEA's.

4.2 Useful Notations

In any typical non-dominated sorting MOEA, a number of individuals are generated randomly, evaluated and then sorted according to non-domination criteria. Figure 4.1 provides a schematic view of the individuals in the objective space and design variable space after sorting.

Here, we consider M individuals with n design variables and k objectives. Suppose, $x_i^p(\phi)$ and $f_j^p(\phi)$ denote the design variable i and the objective value j of an individual p in front $\mathcal{F}_\phi$. If we sort the individuals in every front with respect to one objective, we can also assume that a specific individual p of each front is the same individual moving

[12]Please note that we are not interested in accuracy of the approximation of objective function, rather we put stress only on the accurate behavior modeling of the stochastic search procedure.

[13]Here "Hosting Optimizer" refers to any kind of MOEA that uses non-domination based sorting on the population before selection process. It is considered as the "Base MOEA" for our experiment.

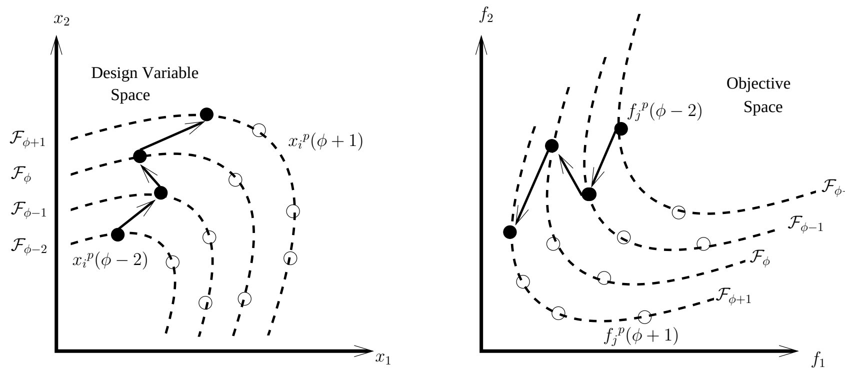


Figure 4.1: After non-dominated sorting, individuals in each front are sorted with respect to one objective. For modeling purposes, (●) is considered as the same individual moving from front $\mathcal{F}_{\phi-2}$ to front $\mathcal{F}_{\phi}$

towards front $\mathcal{F}_{\phi}$ from front $\mathcal{F}_{\phi-2}$ (Refer to Figure 4.1). Here, decreasing values of ϕ represent a worse front[14]. Now if we could somehow extrapolate the trajectory of p, we can infer that this individual will eventually reach the next front $\mathcal{F}_{\phi+1}$ (Refer to Figure 4.1). Moreover, if the distance between two consecutive fronts is small, then we can also assume that this trajectory is piece-wise linear.

Considering only one individual p, we can also say that this search algorithm takes $x_i^p(\phi-1)$ as input and generates $x_i^p(\phi)$ as output. Instead of considering the search algorithm as a "procedure" *per se*, let us consider it as a "Dynamic System" [149], [150] (with transfer function H), which takes the series

$$\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2)\ldots\} \tag{4.1}$$

as input and generates

$$\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2)\ldots\} \tag{4.2}$$

as output (Refer to Figure 4.2). Again, if we consider an inverse system (with transfer function H^{-1}), then it will generate

$$\{x_i^p(\phi), x_i^p(\phi-1), x_i^p(\phi-2)\ldots\} \tag{4.3}$$

[14]Here, the fronts $\mathcal{F}_{\phi}, \mathcal{F}_{\phi-1}, \ldots, \mathcal{F}_1$ are generated in a single generation, i.e. $\mathcal{F}_{\phi}$ and $\mathcal{F}_{\phi-i}$ are not from two consecutive generations.

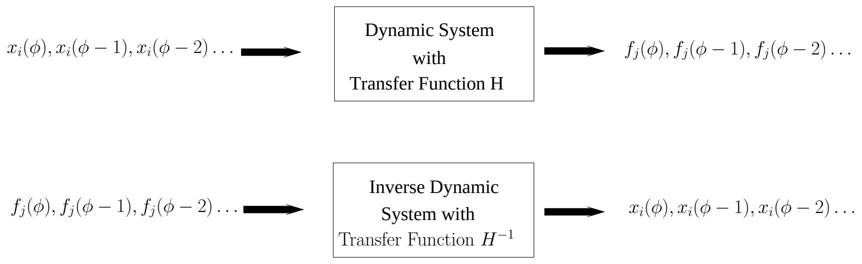

$x_i(\phi), x_i(\phi-1), x_i(\phi-2)\dots$
Dynamic System with Transfer Function H
$f_j(\phi), f_j(\phi-1), f_j(\phi-2)\dots$
$f_j(\phi), f_j(\phi-1), f_j(\phi-2)\dots$
Inverse Dynamic System with Transfer Function H^{-1}
$x_i(\phi), x_i(\phi-1), x_i(\phi-2)\dots$

Figure 4.2: The forward and inverse dynamic system

as output when

$$\{f_j^p(\phi), f_j^p(\phi-1), f_j^p(\phi-2)\dots\} \tag{4.4}$$

is input. Therefore, if we could somehow approximate the system's parameters, then we can easily approximate the design variable of the next front $x_i^p(\phi+1)$ (when $f_j^p(\phi+1)$ is given as input). The concept of "Forward" and "Inverse" system is depicted in Figure 4.2.

Our next task is to generate the so-called "Mirage Solutions"[15] $f_j^p(\phi+1)$. Approximation of $f_j^p(\phi+1)$ is quite straight-forward since in any type of MOP, we have to maximize (or minimize) multiple objectives. In the case of a minimization problem, the objective value in the next front $\mathcal{F}_{\phi+1}$ will be smaller than that of the current front $\mathcal{F}_\phi$ by some amount Δf. So for a minimization problem,

$$f_j^p(\phi+1) = f_j^p(\phi) - \Delta f \tag{4.5}$$

The preceding discussions dictate that PFVO depends on the following formulations:

- approximating the parameters of the dynamic system
- approximating $f_j^p(\phi+1)$ from $f_j^p(\phi)$ by choosing a suitable Δf value
- from $f_j^p(\phi+1)$, approximate the design variable $x_i^p(\phi+1)$ of the next front $\mathcal{F}_{\phi+1}$

[15] "Mirage Solutions" or "Projected Solutions" means the objective values of the next (future) front that are approximated from current design variables, for more details please see [63], [64]

4.3 Model Formulation

Before describing the model formulation in detail, first we construct a simple linear system for the design variable *i* and objective *j*:

$$x_i(\phi) + a_0 x_i(\phi - 1) = b_0 f_j(\phi) + b_1 f_j(\phi - 1) + \epsilon(\phi) \tag{4.6}$$

Here, f_j are input and x_i are considered as output. Therefore,

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \begin{bmatrix} a_0 \\ b_0 \\ b_1 \end{bmatrix} + \epsilon(\phi) \tag{4.7}$$

Now, we can adopt a matrix formation:

$$\underbrace{\begin{bmatrix} x_i(\phi) \\ x_i(\phi-1) \\ x_i(\phi-2) \\ \vdots \\ x_i(2) \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \\ x_i(\phi-2) & f_j(\phi-1) & f_j(\phi-2) \\ x_i(\phi-3) & f_j(\phi-2) & f_j(\phi-3) \\ \vdots & \vdots & \vdots \\ x_i(1) & f_j(2) & f_j(1) \end{bmatrix}}_{\mathbf{\Phi}} \cdot \underbrace{\begin{bmatrix} a_0 \\ b_0 \\ b_1 \end{bmatrix}}_{\boldsymbol{\beta}_{ij}} + \underbrace{\begin{bmatrix} \epsilon(\phi) \\ \epsilon(\phi-1) \\ \epsilon(\phi-2) \\ \vdots \\ \epsilon(2) \end{bmatrix}}_{\boldsymbol{\epsilon}}$$

Or we can rewrite

$$\mathbf{y} = \mathbf{\Phi} \cdot \boldsymbol{\beta}_{ij} + \boldsymbol{\epsilon} \tag{4.8}$$

Here, matrix $\boldsymbol{\beta}_{ij}$ denotes parameter of the dynamic system for the design variable *i* and objective *j*. $\boldsymbol{\beta}_{ij}$ can be approximated using Least Squares. Let us denote $\hat{\boldsymbol{\beta}}_{ij}$ as approximated $\boldsymbol{\beta}_{ij}$:

$$\hat{\boldsymbol{\beta}}_{ij} = \underbrace{(\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T}_{\text{pseudo-inverse}} \; \mathbf{y} \tag{4.9}$$

So,

$$x_i(\phi) = \begin{bmatrix} -x_i(\phi-1) & f_j(\phi) & f_j(\phi-1) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{ij} \tag{4.10}$$

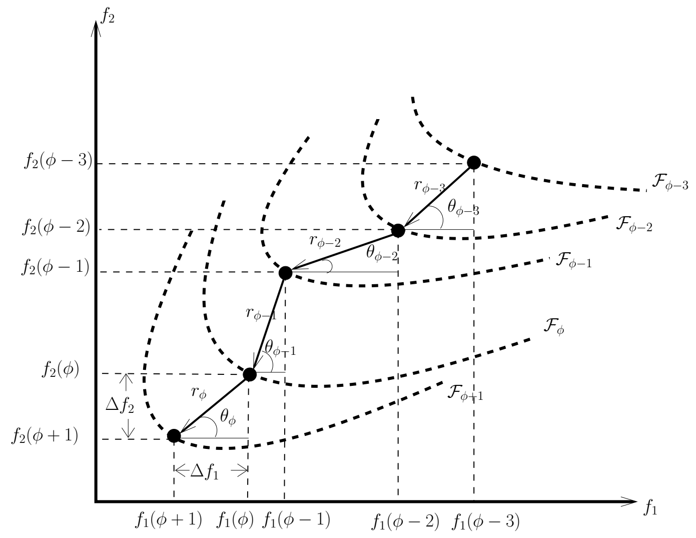


Figure 4.3: Calculation of Δf from existing fronts

Now we have $\hat{\boldsymbol{\beta}}_{ij}$. From equations 4.5 and 4.10, now we can easily approximate the i^{th} design variable of the next front $\mathcal{F}_{\phi+1}$ -

$$x_i(\phi+1) = \begin{bmatrix} -x_i(\phi) & f_j(\phi+1) & f_j(\phi) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{ij} \tag{4.11}$$

$$= \begin{bmatrix} -x_i(\phi) & f_j(\phi)-\Delta f & f_j(\phi) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{ij} \tag{4.12}$$

4.3.1 Calculation of Δf

The critical portion of the algorithm is to calculate the approximate objective values of the next front. In the initial implementation of PFVO we have assigned the value of Δf from empirical experimentations. In the previous implementations, the value of Δf was calculated based on the difference in nadir and utopia objective values [1]. However, the performance was not identical for all problems/host algorithms. So, in the later implementation, we have adopted an adaptive calculation of Δf based on the average Δf of the previous points. Since, PFVO does not necessarily need exact predicted values of the next front, we have found that a "informed guess" is feasible enough when approximating the design variables for the next front[16]. Moreover, it is also necessary to limit the computational complexity of the whole procedure. The calculation is conducted as follows for two objectives problem (see Figure 4.3) -

$$\text{For first objective} \quad f_1, \quad \Delta f_1 = r_\phi \cos\theta_\phi \tag{4.13}$$

$$\text{For second objective} \quad f_2, \quad \Delta f_2 = r_\phi \sin\theta_\phi \tag{4.14}$$

$$where, \quad r_\phi = \sum_{i=\phi-1}^{1} \frac{r_i}{N} \quad and \quad \theta_\phi = \sum_{i=\phi-1}^{1} \frac{\theta_i}{N} \tag{4.15}$$

[16]Please note that PFVO is not for exact prediction, it will be used to generate an approximate objective values of the next front so that the host optimizer can explore the search space in the right way. This concept is further explained in section 5.6 of chapter 5

4.4 The Algorithm

The algorithm for PFVO is listed in Algorithm 1. Readers can review the schematics illustrated in Figure 4.4 and 4.5 for better understanding of the functionality.

Algorithm 1 Pareto Following Variation Operator(P_t,Δf)

```
Require: Parent population $P_t$ is sorted with respect to non domination.

    $$P_t := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \dots, \mathcal{F}_1\}$$

    and individuals in $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \dots, \mathcal{F}_1$ are sorted again with respect to one objective.
    Each front $\mathcal{F}_i$ has size $m_i$ and $\mathcal{F}_\phi$ is the best front. $\Delta f$ is the objective distance from
    current best front to next approximating front.
Ensure: Creates $km_\phi$ number of approximated individuals of the front $\mathcal{F}_{\phi+1}$
 1: for all objectives $j$ such that $1 \le j \le k$ do
 2:   for all individuals $p$ such that $1 \le p \le m_\phi$ do
 3:     for all design variables $i$ such that $1 \le i \le n$ do
 4:       Construct matrix $\mathbf{y}$ from the individual $p$ from every front $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \dots, \mathcal{F}_1$
 5:       Construct matrix $\mathbf{\Phi}$ from the individual $p$ from every front $\mathcal{F}_\phi, \mathcal{F}_{\phi-1}, \dots, \mathcal{F}_1$
 6:       Calculate $\hat{\boldsymbol{\beta}}_{ij} := (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}$
 7:       Approximate $f_j^p(\phi+1) := f_j^p(\phi) - \Delta f$
 8:       $x_i(\phi+1) := \begin{bmatrix} -x_i(\phi) & f_j(\phi) - \Delta f & f_j(\phi) \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_{ij}$
 9:       $x_i(\phi+1)$ is the design variable $i$ of the new approximated individual $I^z$, where
          $z = p + (j-1)m_\phi$.
10:       So, $x_i^{p+(j-1)m_\phi}(\phi+1) := x_i(\phi+1)$
11:     end for
12:   end for
13: end for
14: Returns new approximated population $\mathcal{F}_{\phi+1}$ with population size $km_\phi$
```

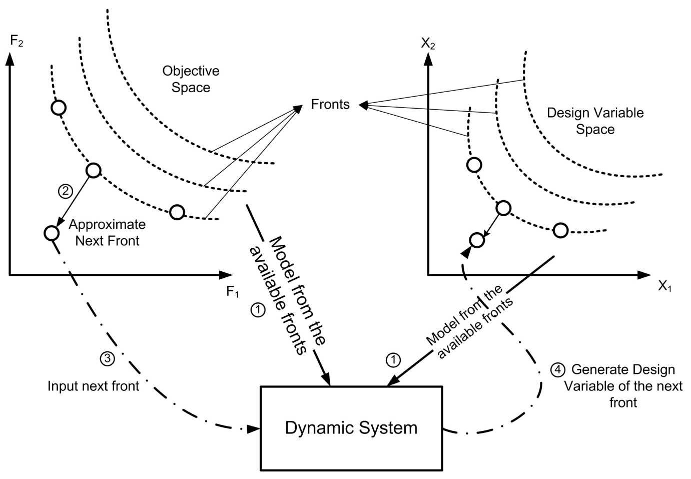

F₂
Objective Space
Fronts
X₂
Design Variable Space
②
Approximate Next Front
F₁
Model from the available fronts
①
X₁
Model from the available fronts
①
③
Input next front
④ Generate Design Variable of the next front
Dynamic System

Figure 4.4: Working steps of the Pareto-following Variation Operator

A basic flowchart for the mathematical operations are illustrated in Figure 4.5. This figure represents the schematic of PFVO for a scenario where the problem has three design variables and two objectives. After non-dominated sorting, three fronts are created. The approximated population has km_ϕ individuals.

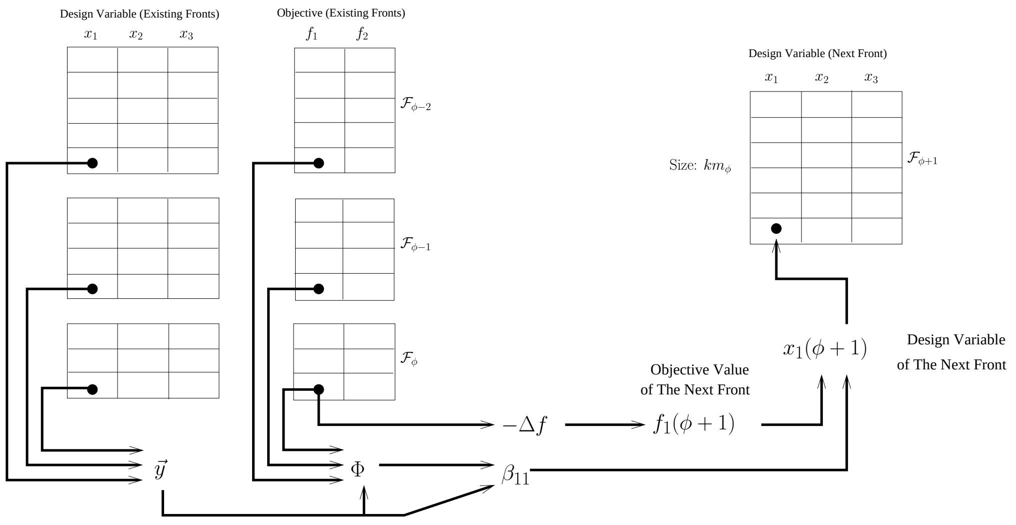


Figure 4.5: Mathematical steps in the Pareto-following Variation Operator. Here the operation is illustrated for objective function f_1 and design variable x_1

4.5 Complexity Analysis

We now provide the complexity analysis of our algorithm. To find the pseudo inverse in equation 13, we have used QR factorization with the aid of the Householder transformation [151]. The complexity of the algorithm largely depends on the size of the matrix $\mathbf{\Phi}$ in equation 4.8. Here we denote this time (or front) steps as t and the initial "guess" of the dynamic model starts with 2 time (or front) steps in equation 4.7. We are applying QR factorization on matrix $\mathbf{\Phi}$ whose dimension is $(|\phi| - 1) \times t$. Here $|\phi|$ is the size of the best front $\mathcal{F}_\phi$[17]. This operation requires $2(|\phi| - 1)t^2 - 2/3t^3$ computations. So it has a computational complexity of $\mathcal{O}(2(|\phi| - 1)t^2 - 2/3t^3)$.

After applying QR factorization, the upper triangular matrix $\mathbf{R}$ has a dimension of $t \times t$ and the orthogonal matrix $\mathbf{Q}$ has a dimension of $(|\phi| - 1) \times t$. In the next step, we are solving the systems parameter $\boldsymbol{\beta}_{ij}$ in equation 4.9. This requires one inversion on the upper triangular matrix $\mathbf{R}$, one multiplication on $\mathbf{Q}^T\mathbf{y}$ and another multiplication on $\mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y}$. So equation 4.9 and 4.12 have an overall complexity of:

$$\begin{aligned}
\mathcal{O}(2(|\phi| - 1)t^2 - 2/3t^3) + \mathcal{O}((|\phi| - 1)t) \\
+\mathcal{O}(t^3) + \mathcal{O}((|\phi| - 1)^2 t) + \mathcal{O}(t^2) \\
\approx \mathcal{O}(2|\phi|^2 t^2) \\
\text{Here, } |\phi| \gg t
\end{aligned}$$

From line 1 to 12 in Algorithm 1, $\boldsymbol{\beta}_{ij}$ is evaluated for $nk|\phi|$ times. So the overall complexity of the variation operator will be $\mathcal{O}(2nk|\phi|^3 t^2)$. Where n is the number of design variables, k is the number of objectives.

If we consider the complexity with N individuals in the worst case, there will be only two fronts, where the best front has $N - 1$ individuals and worst one has only 1. In that case, $|\phi| = N - 1$. So the overall worst case complexity will be $\mathcal{O}(2nk(N-1)^3 t^2) \approx \mathcal{O}(nkN^3 t^2)$. Moreover, in our experiment $t = 3$, and obviously in the worst case $|\phi| \gg t$, so the variation operator has the complexity of $\mathcal{O}(nkN^3)$, which largely depends on the population size. Actually, this added complexity will not increase the overall running

[17]Both $|\phi|$ and m_ϕ (in Algorithm 1) denotes the size of the best front $\mathcal{F}_\phi$

time of the host optimizer, since this operator can save extra objective evaluation of the hosting optimizer by approximate mapping of the future Pareto front to future design variables. In this experiment we have used the Linear Algebra package **Meschach 1.2b**, [152] for matrix operations.

4.6 Integration Mechanism

The PFVO can be integrated with any kind of evolutionary optimizer where Pareto-based selection is applied. In our experiments, we have tested the performance of PFVO with the **"Non-dominated Sorting Genetic Algorithm - II (NSGA-II)"** [42] and the **"Strength Pareto Evolutionary Algorithm - II (SPEA-II)"** [43]. Moreover, we have also tested with a recently proposed algorithm known as **"Regularity Model Based Multi-objective Estimation of Distribution Algorithm (RM-MEDA)"** [79]. In the following sections, we discuss the detailed implementations of the integration mechanism.

4.6.1 Integration with NSGA-II

In the case of NSGA-II, after applying the non-dominated sorting procedure on the mixed population R_t, we apply the Pareto following variation operator to create the individuals from the next approximated front $\mathcal{F}_{\phi+1}$. These new individuals are then combined with the newly created individuals in the parent population P_t (Refer to line 6 and 7 of the Algorithm 2). For the original pseudo code of NSGA-II, readers are referred to [42]. We have integrated PFVO with original implementation of NSGA-II available at `http://www.iitk.ac.in/kangal/index.shtml`. Detailed experimental results are provided in chapter 5.

Algorithm 2 NSGA-II with Pareto Following Variation Operator

```
Require: Randomly generated parent population $P_t$ at generation $t$ with population size $M$.
Ensure: After $t_{max}$ number of iteration, population $P_{t_{max}}$ will represent solution of the problem.
 1: while $t \leq t_{max}$ do
 2:   Start with child population, $Q_t := \emptyset$
 3:   Create mixed population, $R_t := P_t \cup Q_t$
 4:   $\mathcal{F}$ :=Apply Non-dominated Sort on $R_t$, create $\phi$ number of fronts.
      i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
 5:   if $\phi > 1$ then
 6:     $\mathcal{F}_{\phi+1}$ :=Pareto Following Variation($R_t$,$\Delta f$)
        $|\mathcal{F}_{\phi+1}| = km_\phi$
 7:     Insert newly approximated population to $R_t$, $R_t := R_t \cup \mathcal{F}_{\phi+1}$
 8:     $\mathcal{F}$ :=Apply Non-dominated Sort on $R_t$, create $\phi$ number of fronts.
        i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
 9:   end if
10:   $P_t := \emptyset$ and $i := 1$
11:   repeat
12:     Assign crowding distance on $\mathcal{F}_i$
13:     $P_{t+1} := P_{t+1} \cup \mathcal{F}_i$
14:     $i := i + 1$
15:   until $|P_{t+1}| + |\mathcal{F}_i| \leq M$
16:   Apply crowding distance sorting on $\mathcal{F}_i$
17:   Choose the first $(M - |P_{t+1}|)$ individuals of $\mathcal{F}_i$
18:   Use selection, crossover and mutation to create child population $Q_{t+1}$ from $P_{t+1}$
19: end while
```

4.6.2 Integration with SPEA-II

We have also integrated PFVO with SPEA-II. In this case we, have applied PFVO on the external archive before the original variation in SPEA-II occurs (see line 15 of Algorithm 3). The overall procedure of this integration scheme is illustrated in Algorithm 3. For the original pseudo code of SPEA-II, please refer to [43]. We have used the original SPEA-II code available at `http://www.tik.ee.ethz.ch/sop/pisa/`. Detailed experimental results are provided in chapter 5.

Algorithm 3 SPEA-II with Pareto Following Variation Operator

```
Require: Randomly generated parent population $P_t$ at generation $t$ with population size
    $N$ and an archive of external population $\overline{P}_t$ with size $\overline{N}$
Ensure: After $t_{max}$ number of iteration, population $\overline{P}_{t_{max}}$ will represent solution of the
    problem.
 1: Start with initial population, $P_0 := \emptyset$ and an empty archive (external set) $\overline{P}_0 := \emptyset$
 2: set $t = 0$
 3: while $t \leq t_{max}$ do
 4:     Calculate fitness values of $P_t$ and $\overline{P}_t$
 5:     $\mathcal{F}$ :=Apply Non-dominated Sort on $P_t$, create $\phi$ number of fronts.
        i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
 6:     $\overline{\mathcal{F}}$ :=Apply Non-dominated Sort on $\overline{P}_t$, create $\phi$ number of fronts.
        i.e. $\overline{\mathcal{F}} := \{\overline{\mathcal{F}}_\phi, \overline{\mathcal{F}}_{\phi-1} \ldots \overline{\mathcal{F}}_1\}$
 7:     Apply environmental selection. Copy all non-dominated individuals from $P_t$ and
        $\overline{P}_t$ to $\overline{P}_{t+1}$
 8:     if $|\overline{P}_{t+1}| > N$ then
 9:         Truncate individuals from $\overline{P}_{t+1}$ using clustering algorithm
10:     else
11:         Fill $\overline{P}_{t+1}$ with dominated individuals from $P_t$ and $\overline{P}_t$
12:     end if
13:     Again apply non-dominated sort on $\overline{P}_{t+1}$ and create $\phi$ number of fronts.
        i.e. $\overline{\mathcal{F}} := \{\overline{\mathcal{F}}_\phi, \overline{\mathcal{F}}_{\phi-1} \ldots \overline{\mathcal{F}}_1\}$
14:     if $\phi > 1$ then
15:         $\overline{\mathcal{F}}_{\phi+1}$ :=Pareto Following Variation($\overline{P}_{t+1}$,$\Delta f$)
            $|\overline{\mathcal{F}}_{\phi+1}| = km_\phi$
16:         Insert newly approximated population to $\overline{P}_{t+1}$, $\overline{P}_{t+1} := \overline{P}_{t+1} \cup \overline{\mathcal{F}}_{\phi+1}$
17:     end if
18:     Perform binary tournament selection with replacement on $\overline{P}_{t+1}$ to fill the mating
        pool.
19:     Apply crossover and mutation on individuals in $\overline{P}_{t+1}$ and copy them to $P_{t+1}$
20:     set $t := t + 1$
21: end while
```

4.6.3 Integration with RM-MEDA

The **"Regularity Model Based Multi-objective Estimation of Distribution Algorithm (RM-MEDA)"** is a recent development reported in [79]. This algorithm provides a special interest to our experiment since it does not fall into the general MOEA categories. This algorithm is an instance of so called **"Estimation of Distributed Algorithm (EDA)"**.

Algorithm 4 RM-MEDA with Pareto Following Variation Operator

```
Require: Randomly generated parent population $P_t$ at generation $t$ with population size
    $N$
Ensure: After $t_{max}$ number of iteration, population $P_{t_{max}}$ will represent solution of the
    problem.
 1: Start with initial population, $P_0 := \emptyset$
 2: set $t = 0$
 3: while $t \leq t_{max}$ do
 4:     Calculate fitness values of $P_t$
 5:     set $t := t + 1$
 6:     Build the probability model for the distribution of the solutions in $P_t$ using Local
        Principal Component Analysis (Local-PCA).
 7:     Generate new solution set $Q$ from the built probability model.
 8:     Evaluate fitness values of individuals in $Q$
 9:     Select $N$ individuals from $P_t$ and $Q$ and create new population $P_{t+1}$
        i.e. $P_{t+1} = P_t \cup Q$
10:     $\mathcal{F}$ :=Apply Non-dominated Sort on $P_{t+1}$, create $\phi$ number of fronts.
        i.e. $\mathcal{F} := \{\mathcal{F}_\phi, \mathcal{F}_{\phi-1} \ldots \mathcal{F}_1\}$
11:     if $\phi > 1$ then
12:         $\mathcal{F}_{\phi+1}$ :=Pareto Following Variation($P_{t+1}$,$\Delta f$)
            $|\mathcal{F}_{\phi+1}| = km_\phi$
13:         Insert newly approximated population to $P_{t+1}$, $P_{t+1} := P_{t+1} \cup \mathcal{F}_{\phi+1}$
14:         Retain the size of population in $P_{t+1}$ to $N$. If it exceeds this limit, just remove the
            individuals from the worst front.
15:     end if
16:     set $t := t + 1$
17: end while
```

In EDA's, normal genetic operators, such as mutation/crossover are not applied, rather probabilistic modeling or **"Intelligent Variation Operator"** based techniques are adopted.

The working principle of RM-MEDA is mainly derived from the regularity property defined by the **Karush-Kuhn-Tucker** condition [2]. This condition suggests that the Pareto set, in the decision space, of a continuous multi-objective optimization problem is a piecewise continuous $(m-1)D$ manifold [31], where m is the number of objectives. Based on this regularity property, RM-MEDA models a promising area in the decision space by a probability distribution whose centroid is a $(m-1)D$ piecewise continuous manifold [30], [153]. The local "Principal Component Analysis (PCA)" algorithm was used for building such a model. New trial solutions were sampled from the model thus built. A non-dominated sorting-based selection is used for choosing solutions for the next generation. The study suggests that RM-MEDA exhibits promising results with respect to NSGA-II. Detailed experimentation will be covered in chapter 5. Algorithm 4 illustrates the integration mechanism with RM-MEDA. We have used the original RM-MEDA source code available at `http://privatewww.essex.ac.uk/~azhou/publication.htm`.

4.7 Conclusion

In this chapter we have discussed the architecture of the proposed model in detail and its integration with different base algorithms. In the next chapter we present the experimental details along with performance measurement techniques and analysis.

Chapter 5

Experimental Results and Analysis

5.1 Introduction

In this chapter we introduce the benchmark problems typically used to test the performance of MOEA's and the definition of different metrics used to measure their performance. We will also illustrate the detailed experimental results using the benchmark problems and their corresponding statistical analysis.

There have been several attempts to define test suites or toolkits for building test suites. However, existing multiobjective test problems do not test a wide range of characteristics, and often have design flaws. Typical defects include not being scalable or being susceptible to simple search strategies. However we have chosen two types benchmark problem sets known as **"ZDT"** [33] and **"DTLZ"** [154] test suite. We now review those problems.

5.1.1 ZDT Test Problem Suite

The **ZDT** test suite was first proposed in [33]. The ZDT test suites are confined to only two objective problems and do not consider maximization or mixed minimization/maximization problems. The problems range from linear/nonlinear, connected/disconnected, concave/convex etc. Each of the test functions defined in Table 5.1 is structured in the same manner and consists of three functions f_1, g and h :

$$\begin{aligned} Minimize \quad \mathcal{T}(x) &= (f_1(x_1), f_2(\vec{x})) \\ \text{subject to} \quad f_2(x) &= g(x_2, \ldots, x_m)h(f_1(x_1), g(x_2, \ldots, x_m)) \\ where \quad \vec{x} &= (x_1, x_2, \ldots, x_m) \end{aligned}$$

The function f_1 is a function of the first decision variable only, g is a function of the remaining $m-1$ variables, and the parameters of h are the function values of f_1 and g. The test functions differ in these three functions as well as in the number of variables m and in the values the variables may take. There are total 6 problems defined in ZDT suite, among them ZDT5 is an instance of combinatorial optimization problem. Since, our model is well matched to numerical problems, we did not include ZDT5 in our experiments. For More detailed empirical analysis on ZDT test suite can be found in [155]. In Table 5.1 the definition of ZDT test suite problems are provided.

The ZDT Test Problem Suite

Problem	Definition	Parameter Domains
ZDT1	$f_1 = y_1$ $g = 1 + 9\sum_{i=1}^{k} z_i/k$ $h = 1 - \sqrt{f_1/g}$	[0,1]
ZDT2	as ZDT1, except $h = 1 - (f_1/g)^2$	$[0, 1]$
ZDT3	as ZDT1, except $h = 1 - \sqrt{(f_1/g)} - (f_1/g)\sin(10\pi f_1)$	$[0, 1]$
ZDT4	as ZDT1, except $g = 1 + 10k + \sum_{i=1}^{k}(z_i^2 - 10\cos(4\pi z_i))$	$y_1 \in [0, 1]$ $z_1, z_2, \ldots, z_k \in [-5, 5]$
ZDT6	$f_1 = 1 - \exp(-4y_1)\sin^6(6\pi y_1)$ $g = 1 + 9(\sum_{i=1}^{k} z_i/k)^{0.25}$ $h = 1 - (f_1/g)^2$	[0,1]

Table 5.1: The five real-valued ZDT two objective problems. The second objective is $f_2(\vec{y}, \vec{z}) = g(\vec{z})h(f_1(\vec{y}), g(\vec{z}))$, where both objectives are to be minimized

5.1.2 DTLZ Test Problem Suite

The **DTLZ** suite of benchmark problems, proposed in [154], is unlike the majority of multi-objective test problems in the sense that the problems are scalable to any number of objectives. This is an important characteristic that has facilitated several recent investigations into what are commonly called "many" objective problems[18].

Nine test problems are included in the DTLZ test suite, of which the first seven are shown in Table 5.2. DTLZ8 and DTLZ9 have side constraints, hence their omission from this research. More empirical analysis on DTLZ test suite can be found in [155].

The DTLZ Test Problem Suite

Problem	Definition	Parameter Domains
DTLZ1	$f_1 = (1+g)0.5\prod_{i=1}^{M-1} y_i$ $f_{m=2:M-1} = (1+g)0.5(\prod_{i=1}^{M-m} y_i)(1-y_{M-m+1})$ $f_M = (1+g)0.5(1-y_1)$ $g = 100[k + \sum_{i=1}^{k}((z_i-0.5)^2 - \cos(20\pi(z_i-0.5)))]$	[0,1]
DTLZ2	$f_1 = (1+g)0.5\prod_{i=1}^{M-1} \cos(y_i\pi/2)$ $f_{m=2:M-1} = (1+g)0.5(\prod_{i=1}^{M-m} \cos(y_i\pi/2))\sin(y_{M-m+1}\pi/2)$ $f_M = (1+g)\sin(y_1\pi/2)$ $g = \sum_{i=1}^{k}(z_i-0.5)^2$	[0,1]
DTLZ3	as DTLZ2, except the equation for g is replaced by the one from DTLZ1	$[0,1]$
DTLZ4	as DTLZ2, except all $y_i \in \vec{y}$ are replaced by y_i^{α}, where $\alpha > 0$	$[0,1]$
DTLZ5	as DTLZ2, except all $y_2, \ldots, y_{M-1} \in \vec{y}$ are replaced by $\frac{1+2gy_i}{2(1+g)}$	$[0,1]$
DTLZ6	as DTLZ5, except the equation for g is replaced by $g = \sum_{i=1}^{k} z_i^{0.1}$	$[0,1]$
DTLZ7	$f_{m=1:M-1} = y_m$ $f_M = (1+g)(M - \sum_{i=1}^{M-1}[\frac{f_i}{1+g}(1+\sin(3\pi f_i))])$ $g = 1 + 9\sum_{i=1}^{k} z_i/k$	$[0,1]$

Table 5.2: Seven of the nine DTLZ many objective problems. All objectives are to be minimized

[18]Please note that, we have tested our model on 2 objective instances of DTLZ test suite, this issue is further discussed in section 5.7

5.2 Parameter Settings

Parameter settings is the one of the most important aspect for any MOEA. The parameters include the population size, external archive size, crossover/mutation rates etc. Different algorithms use different approaches to solve the same problem and their respective parameters are dependant on the algorithms' basic working principle.

NSGA-II [42], [24] uses "Simulated Binary Crossover (SBX)" [135] and "Polynomial Mutation (PM)" [136] that depends on the probability distribution indices η_c and η_m respectively. We have also kept the original values of mutation and crossover probability (P_c and P_m) for all problems. For SPEA-II [43], we have also used the same parameters for "Individual Mutation (P^I_m) and Crossover (P^I_c) Probability", "Variable Mutation (P^v_m), Crossover (P^v_c) and Swap (P^v_s) Probability". Also the same values for η_m and η_c as original implementation of SPEA-II. On the other hand, RM-MEDA [79] normalizes the values of (any range) design variables to a specific amount of working boundary, namely, $[0, 1]$ and this specified by "Variable Tolerance" and "Objective Tolerance". It also uses "Parent Centric Recombination (PCX)" [156] [137] along with SBX [136] and PM. We have also kept the same values as original code.

The summary of parameter settings are illustrated in Table 5.3, 5.4 and 5.5 for integration with NSGA-II, SPEA-II and RM-MEDA respectively -

Problem Set	$\|P_t\|$	t_{max}	P_c	P_m	η_c	η_m	Δf
ZDT1, ZDT2, ZDT3	200	200	0.9	0.033	15	20	Adaptive
ZDT4	200	200	0.9	0.1	15	20	Adaptive
ZDT6	200	400	0.9	0.1	15	20	Adaptive
All DTLZ problems	200	400	0.9	0.033	15	20	Adaptive

Table 5.3: Parameter Settings for NSGA-II. Same values of random seeds were used for both NSGA-II and NSGA-II with PFVO (different random seeds in distinct runs). $|P_t|$ is the size of population P_t and t_{max} defines the total number of generations.

Problem Set	$\lvert P_t \rvert$	t_{max}	P_c^I	P_m^I	η_c	η_m	P_c^v	P_m^v	P_s^v	Δf
All ZDT problems	200	200	1.0	1.0	15	20	1.0	1.0	0.5	Adaptive
All DTLZ problems	200	200	1.0	1.0	15	20	1.0	1.0	0.5	Adaptive

Table 5.4: Parameter Settings for SPEA-II. Same values of random seeds were used for both SPEA-II and SPEA-II with PFVO (different random seeds in distinct runs). $|P_t|$ is the size of population P_t and t_{max} defines the total number of generations.

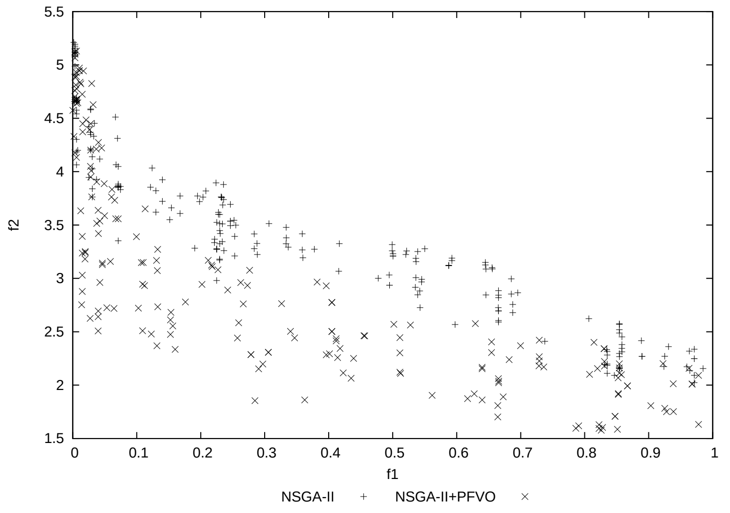
Problem Set	$\lvert P_t \rvert$	t_{max}	Number of Clusters	Training Steps for Local PCA	Extension Ratio	Objective Tolerance	Variable Tolerance
All ZDT Problems	200	200	5	50	0.25	$1.0e^{-5}$	$1.0e^{-5}$
All DTLZ Problems	200	200	5	50	0.25	$1.0e^{-5}$	$1.0e^{-5}$

Table 5.5: Parameter Settings for RM-MEDA. Same values of random seeds were used for both RM-MEDA and RM-MEDA with PFVO (different random seeds in distinct runs). $|P_t|$ is the size of population P_t and t_{max} defines the total number of generations. Here, Δf is adaptively calculated for all problems.

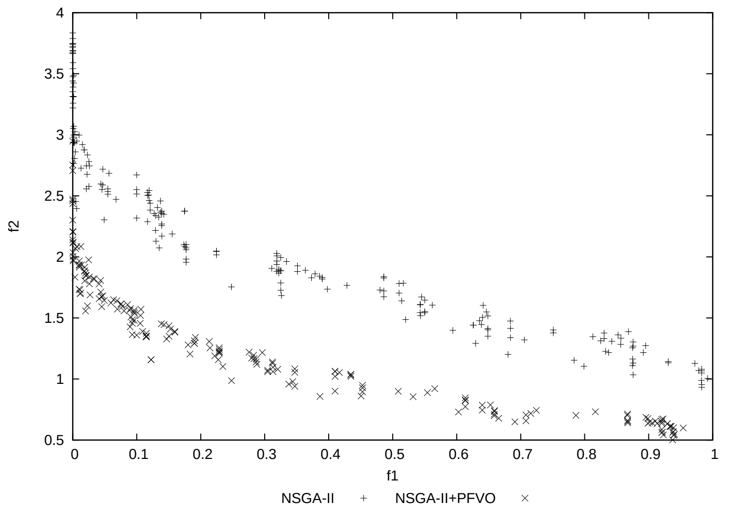
5.3 Simulation Results: Generational Snapshots

In this section we illustrate the simulation results in detail. First, we will present the generational snapshots[19] for every algorithms for the two test suites. For visual clarity of the plots, in the next page, we will present only 3 plots NSGA-II (along with PFVO enhanced NSGA-II) on problem ZDT1. We then provide the rest of the plots in a smaller size for all algorithms (on every problem) to accommodate the space restriction. Here "PFVO + Host Optimizer" indicates "Host Optimizer"(i.e. in our case, NSGA-II, SPEA-II or RM-MEDA) integrated with PFVO. The solutions found by the "Host Optimizer" are presented using (+) and solutions found by the "Host Optimizer + PFVO" are presented in ($\times$).

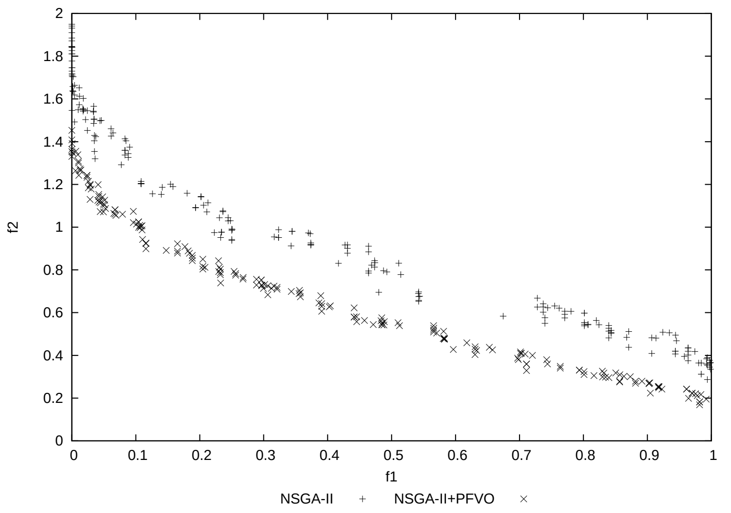
[19]These snapshots show which algorithm reaches to the true Pareto-front within how many function evaluations (FE). For example, in the Figure (a), at FE 800, the solutions generated by both algorithms are similar. However, after 2200 FE (at FE 2600, Figure (b)) solutions generated by PFVO enhanced NSGA-II converged better than the original algorithm. More details about these snapshots can be found online at `http://khaled.ahsan.talukder.googlepages.com`. Please note that FE values for all plots are not consistent, since the number of iterations for a specific MOEA is problem dependent and we have chosen those snapshots where the differences in convergence are prominent.



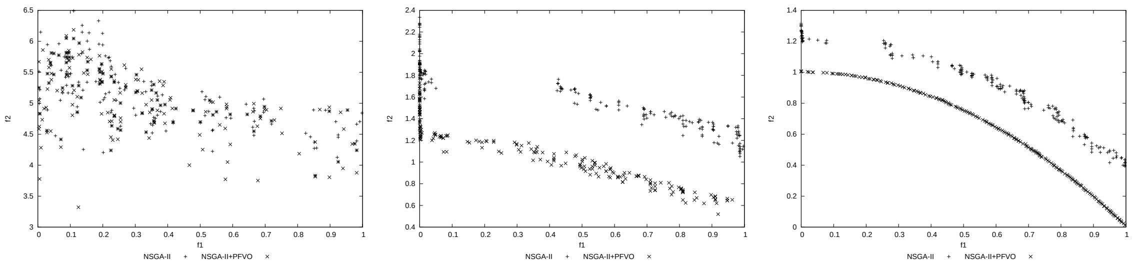
(a) NSGA-II and "NSGA-II + PFVO"on ZDT1 (at Function Evaluation 800)



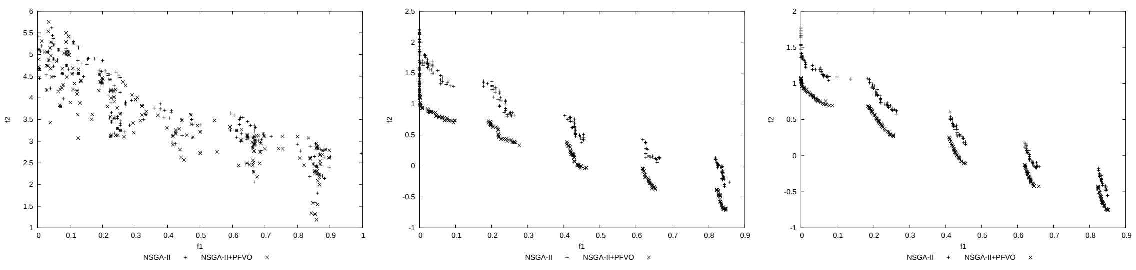
(b) NSGA-II and "NSGA-II + PFVO"on ZDT1 (at Function Evaluation 2600)



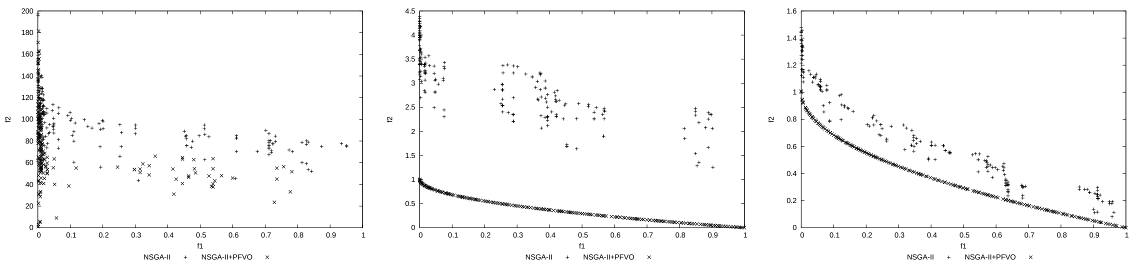
(c) NSGA-II and "NSGA-II + PFVO"on ZDT1 (at Function Evaluation 5200)



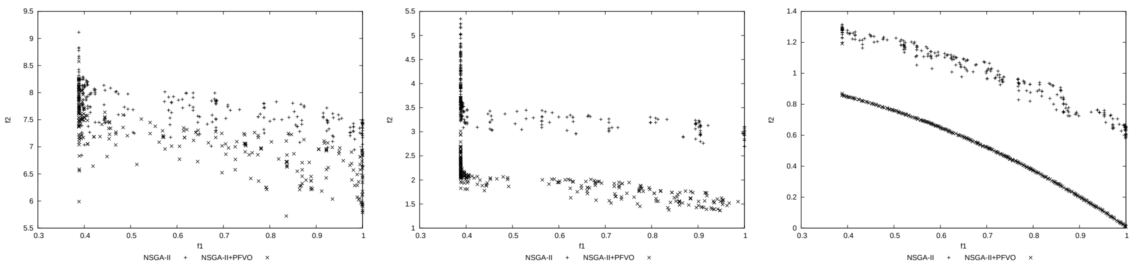
(d) NSGA-II and "NSGA-II + PFVO"on ZDT2 (at FE 400, 5200 and 9600)



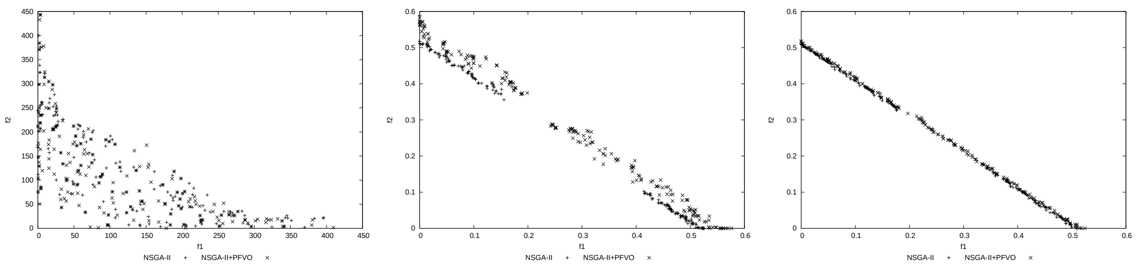
(e) NSGA-II and "NSGA-II + PFVO"on ZDT3 (at FE 600, 4800 and 6600)



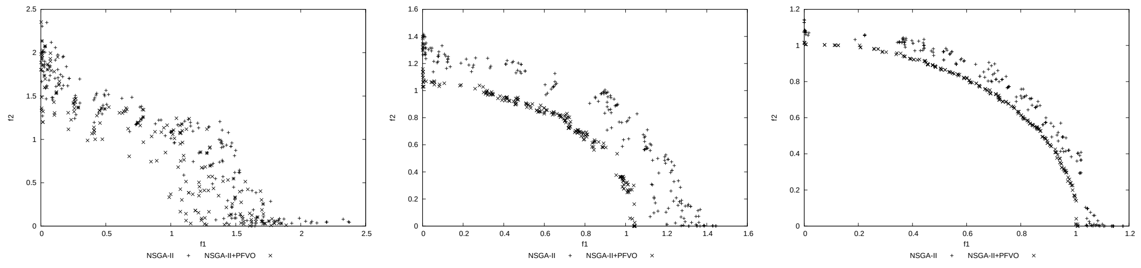
(f) NSGA-II and "NSGA-II + PFVO"on ZDT4 (at FE 800, 22600 and 24800)



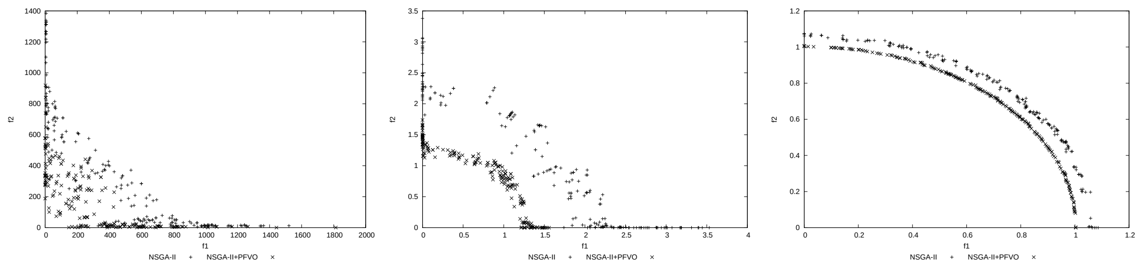
(g) NSGA-II and "NSGA-II + PFVO"on ZDT6 (at FE 1280, 8448 and 20992)



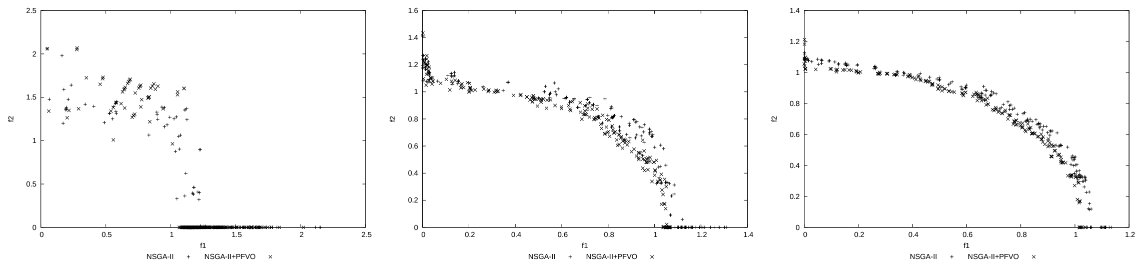
(h) NSGA-II and "NSGA-II + PFVO"on DTLZ1 (at FE 650, 9320 and 11514)



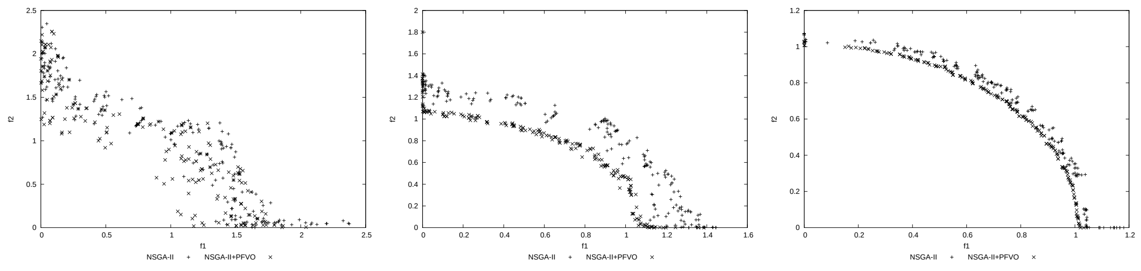
(i) NSGA-II and "NSGA-II + PFVO"on DTLZ2 (at FE 912, 2058 and 3996)



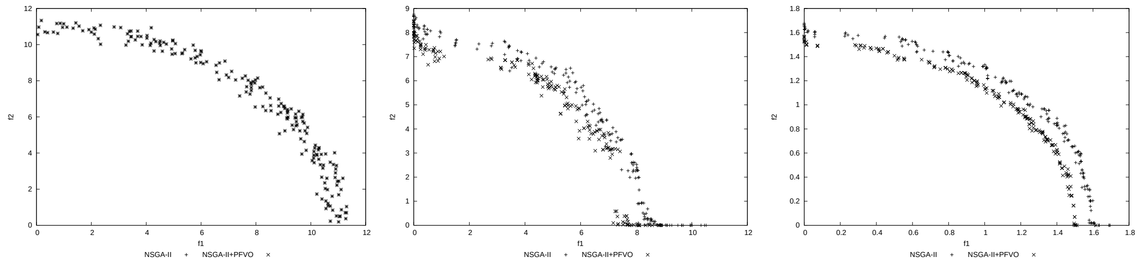
(j) NSGA-II and "NSGA-II + PFVO"on DTLZ3 (at FE 1304, 10188 and 18322)



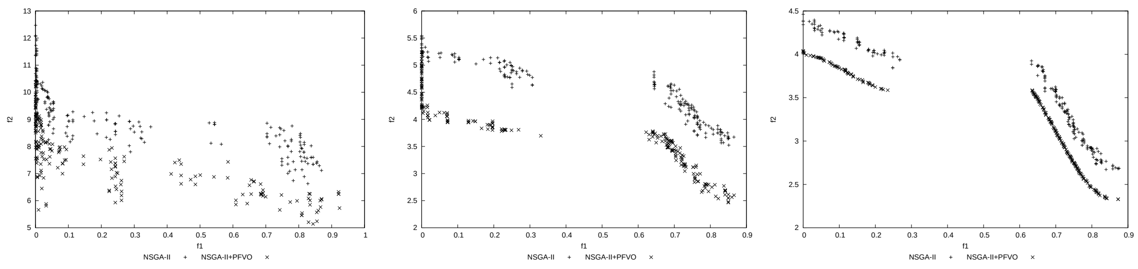
(k) NSGA-II and "NSGA-II + PFVO"on DTLZ4 (at FE 1574, 3114 and 4052)



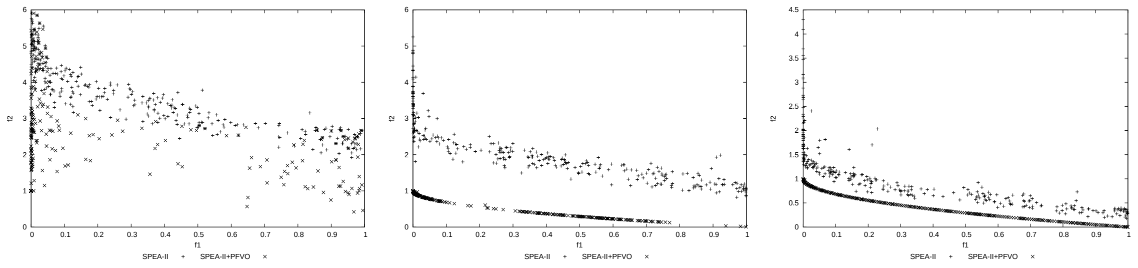
(l) NSGA-II and "NSGA-II + PFVO"on DTLZ5 (at FE 914, 1966 and 4122)



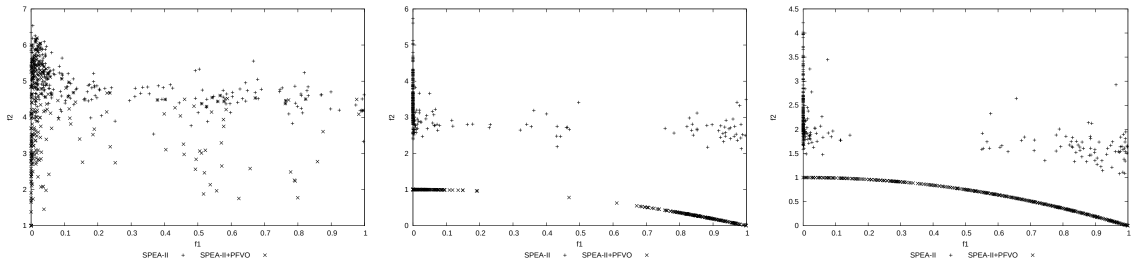
(m) NSGA-II and "NSGA-II + PFVO"on DTLZ6 (at FE 470, 4746 and 42446)



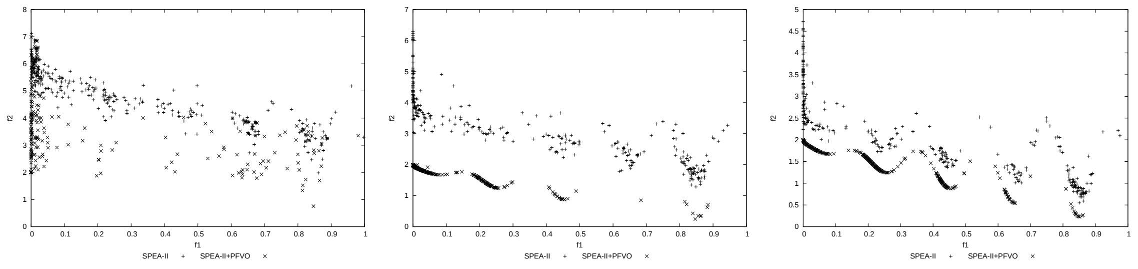
(n) NSGA-II and “NSGA-II + PFVO”on DTLZ7 (at FE 1728, 5534 and 11548)



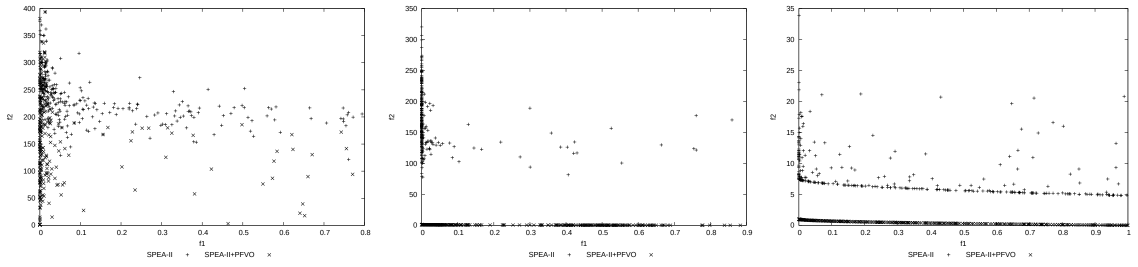
(a) SPEA-II and “SPEA-II + PFVO” on ZDT1 (at FE 544, 2752 and 6668)



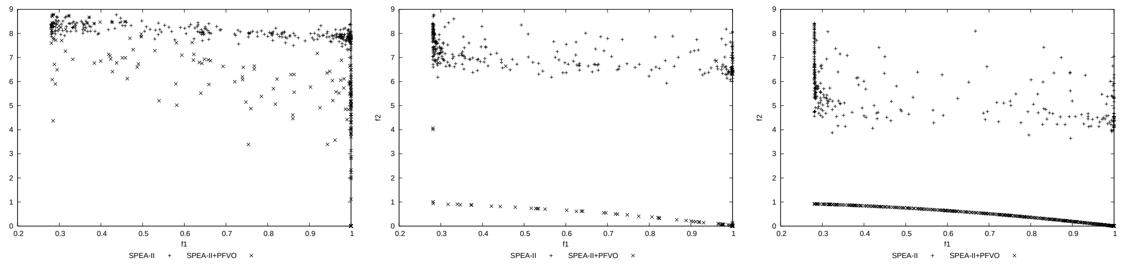
(b) SPEA-II and “SPEA-II + PFVO” on ZDT2 (at FE 534, 3146 and 6344)



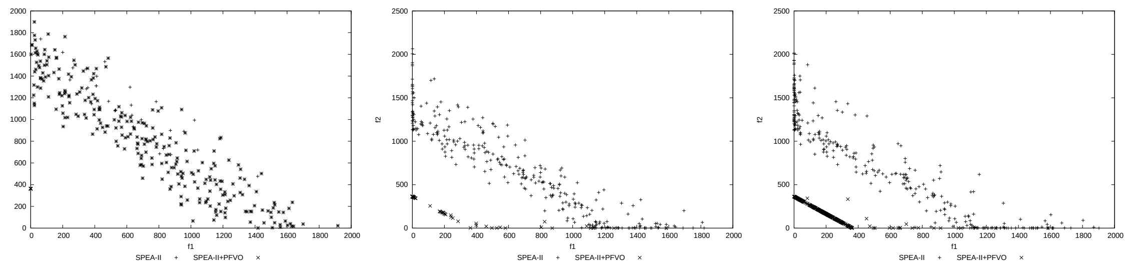
(c) SPEA-II and “SPEA-II + PFVO” on ZDT3 (at FE 532, 2684 and 6560)



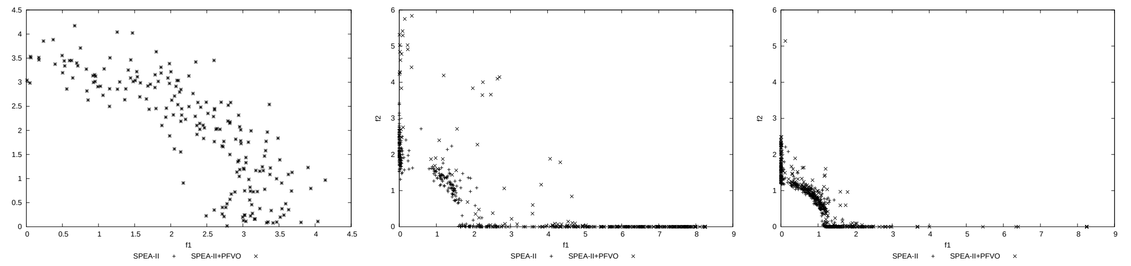
(d) SPEA-II and “SPEA-II + PFVO” on ZDT4 (at FE 636, 1774 and 39208)



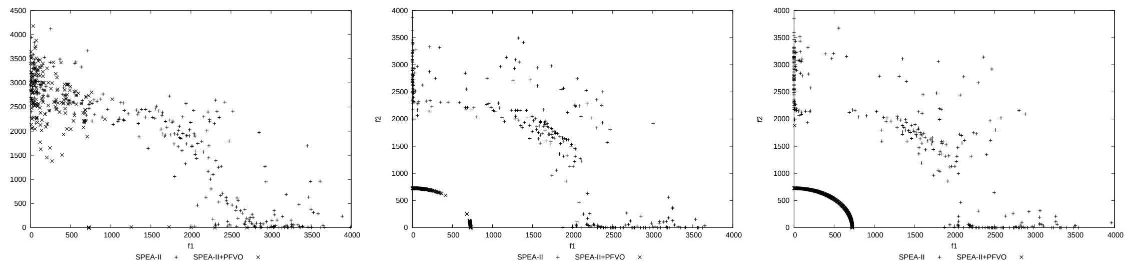
(e) SPEA-II and "SPEA-II + PFVO" on ZDT6 (at FE 910, 2676 and 7438)



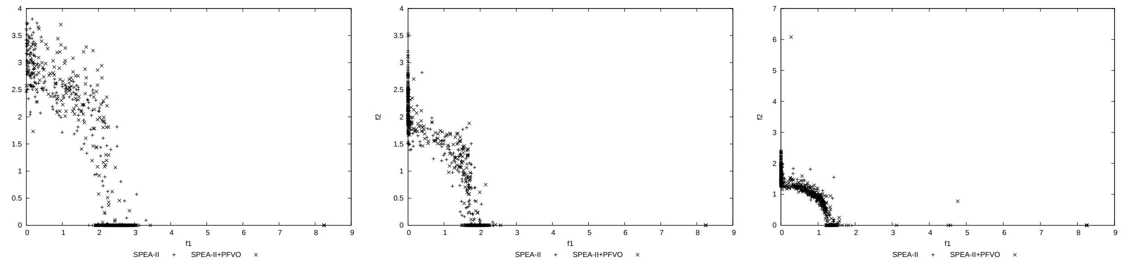
(f) SPEA-II and "SPEA-II + PFVO" on DTLZ1 (at FE 234, 1270 and 3080)



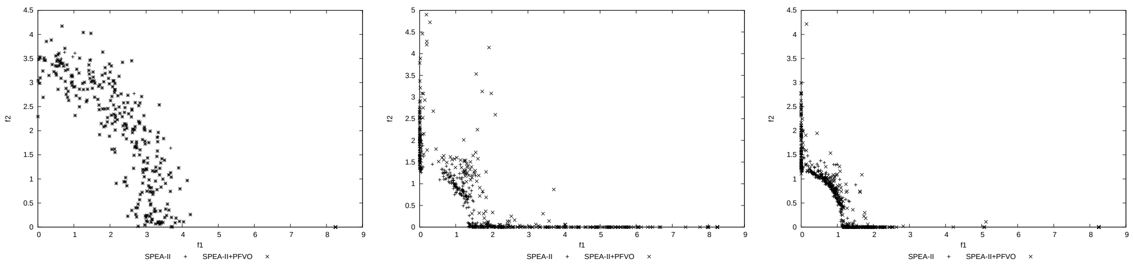
(g) SPEA-II and "SPEA-II + PFVO" on DTLZ2 (at FE 100, 2686 and 13676)



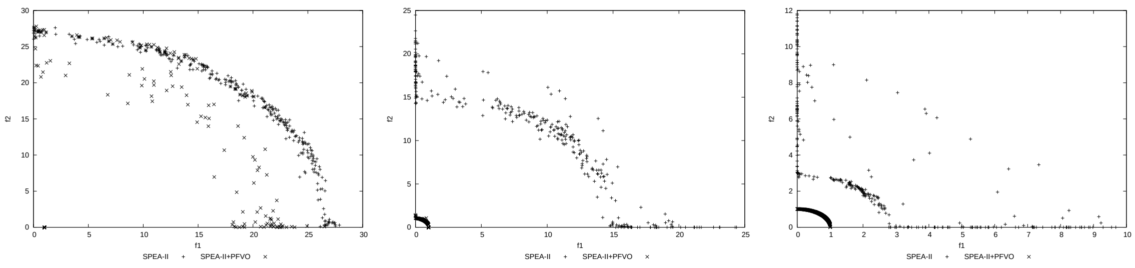
(h) SPEA-II and "SPEA-II + PFVO" on DTLZ3 (at FE 674, 2968 and 6104)



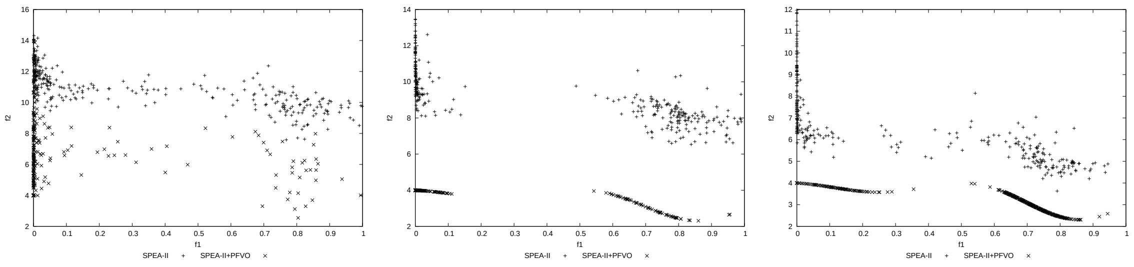
(i) SPEA-II and "SPEA-II + PFVO" on DTLZ4 (at FE 1380, 2832 and 6974)



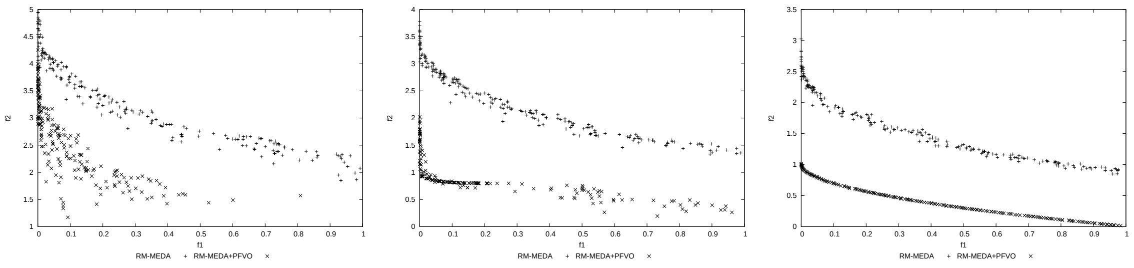
(j) SPEA-II and "SPEA-II + PFVO" on DTLZ5 (at FE 230, 5086 and 17358)



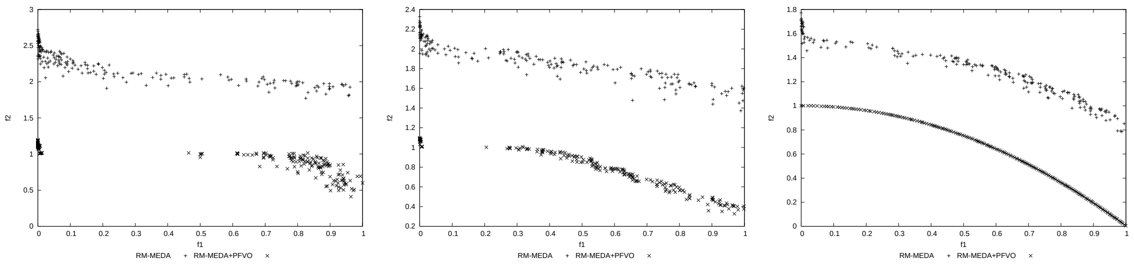
(k) SPEA-II and "SPEA-II + PFVO" on DTLZ6 (at FE 490, 5574 and 44548)



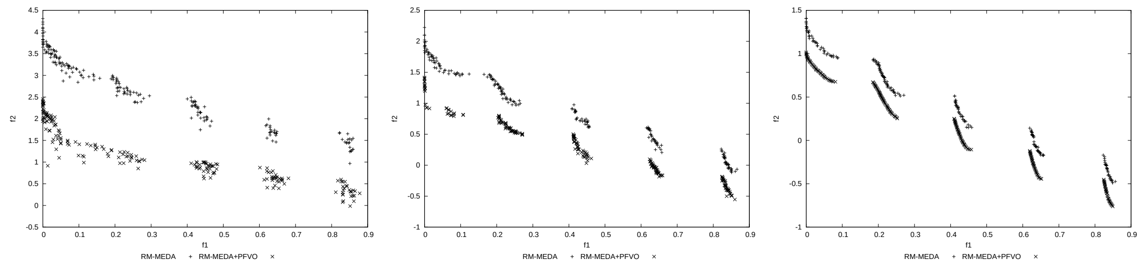
(l) SPEA-II and "SPEA-II + PFVO" on DTLZ7 (at FE 644, 1330 and 4196)



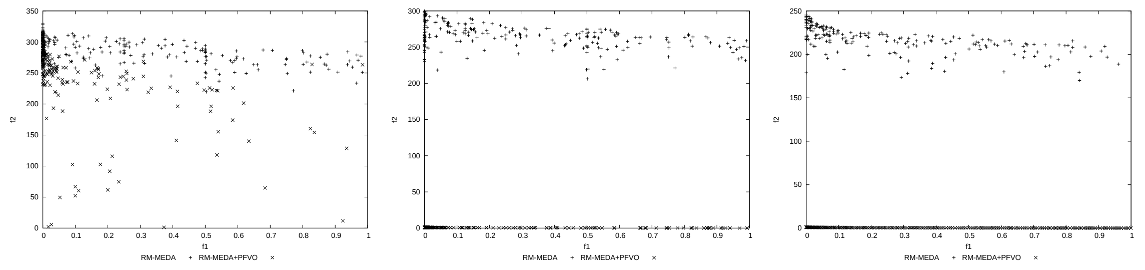
(a) RM-MEDA(+) and "RM-MEDA + PFVO"($\times$) on ZDT1 (at FE 2088, 6424 and 14836)



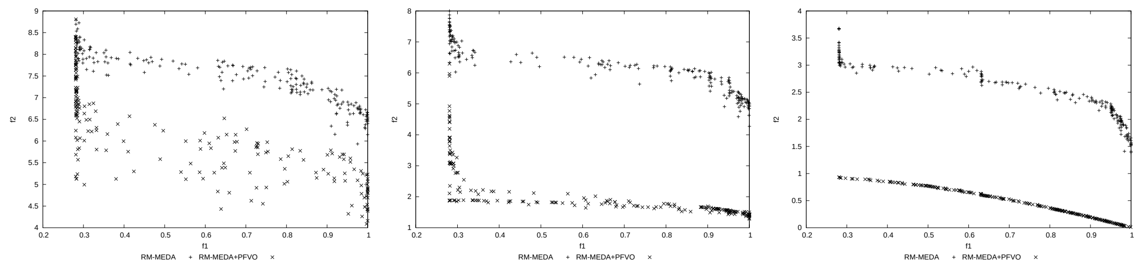
(b) RM-MEDA and "RM-MEDA + PFVO" on ZDT2 (at FE 11612, 14388 and 29496)



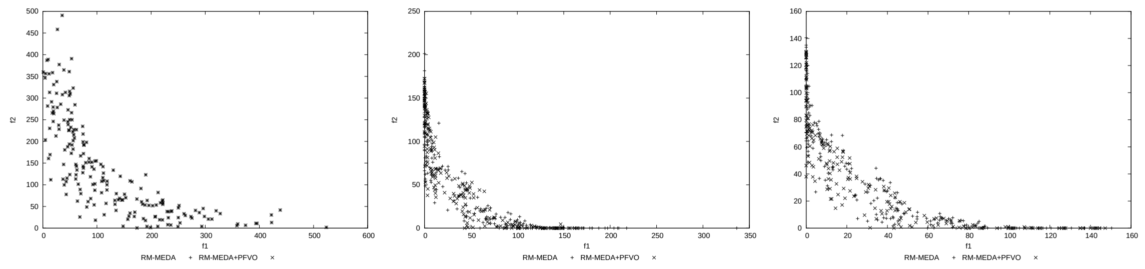
(c) RM-MEDA and “RM-MEDA + PFVO” on ZDT3 (at FE 5908, 30542 and 72144)



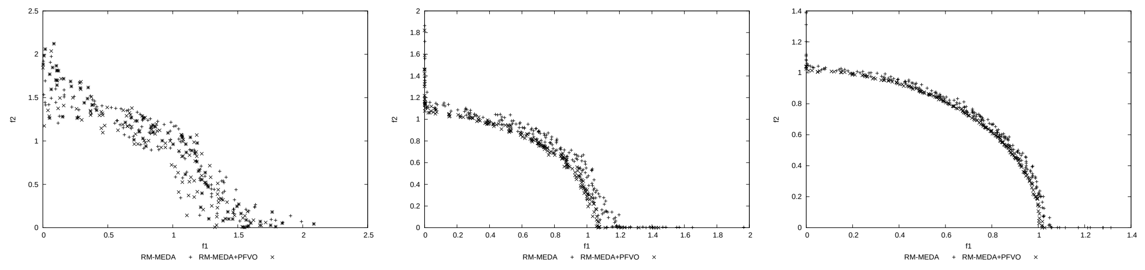
(d) RM-MEDA and “RM-MEDA + PFVO” on ZDT4 (at FE 4784, 6990 and 43668)



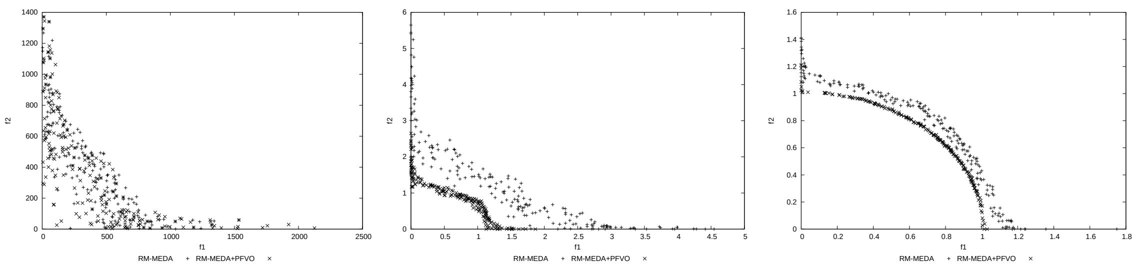
(e) RM-MEDA and “RM-MEDA + PFVO” on ZDT6 (at FE 3320, 11578 and 69758)



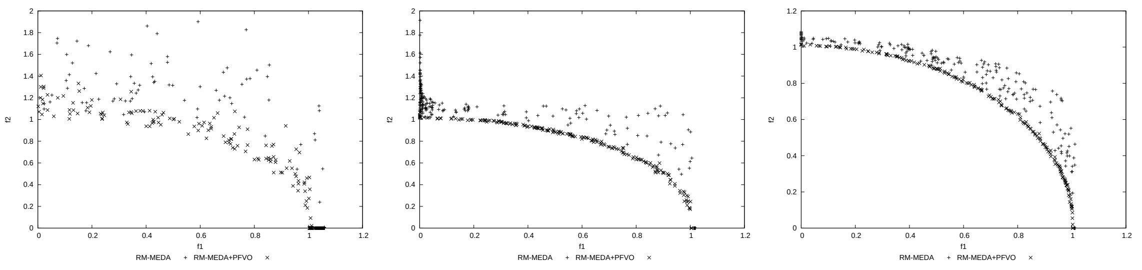
(f) RM-MEDA and “RM-MEDA + PFVO” on DTLZ1 (at FE 200, 20000 and 58600)



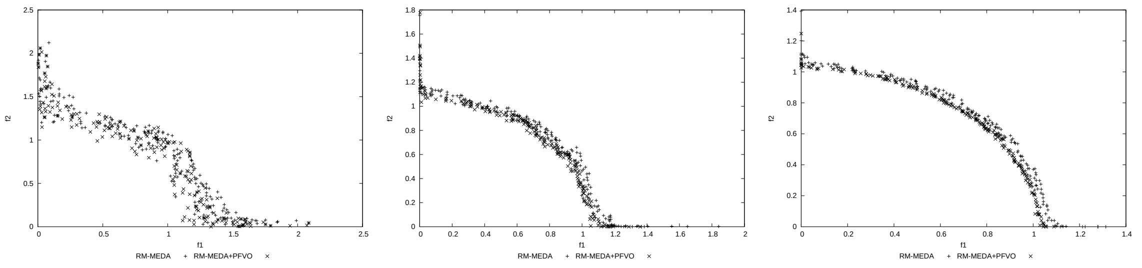
(g) RM-MEDA and “RM-MEDA + PFVO” on DTLZ2 (at FE 468, 3036 and 7342)



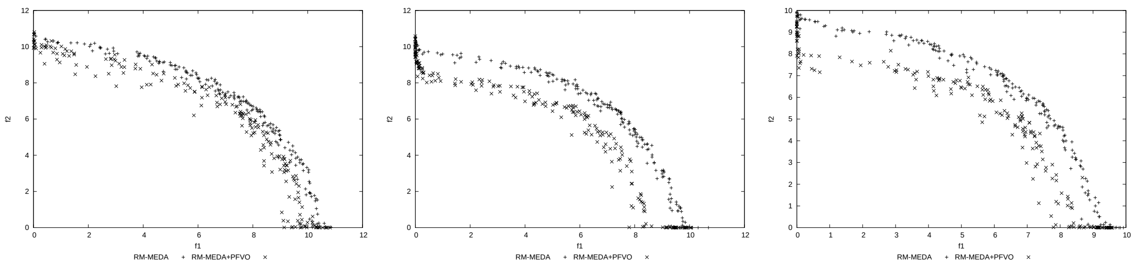
(h) RM-MEDA and “RM-MEDA + PFVO” on DTLZ3 (at FE 644, 11940 and 15938)



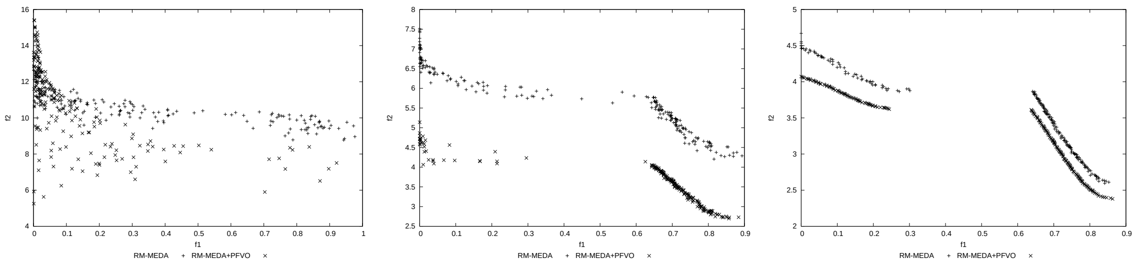
(i) RM-MEDA and “RM-MEDA + PFVO” on DTLZ4 (at FE 7720, 10618 and 14748)



(j) RM-MEDA and “RM-MEDA + PFVO” on DTLZ5 (at FE 710, 3242 and 6154)



(k) RM-MEDA and “RM-MEDA + PFVO” on DTLZ6 (at FE 3182, 9146 and 15554)



(l) RM-MEDA and “RM-MEDA + PFVO” on DTLZ7 (at FE 1066, 15220 and 51978)

5.3.1 Analysis

From these scatter plots, we see that the "Host Optimizer" combined with PFVO converges to the Pareto-front within a smaller number of generations compared to its normal convergence rate. Specially for ZDT problem sets, the gain is substantial. As an example, in Figure (b) in previous section, we see three generational snapshots of the problem ZDT2 at function evaluation 400, 5200 and 9600 respectively (from left to right). At function evaluation 400, we see both of them show similar distribution, after that, at 5200, the convergence is totally different. PFVO enhanced NSGA-II reaches the true Pareto-front at function evaluation 9600 where only NSGA-II is still converging. For the DTLZ sets, this speed up is promising specially for DTLZ4, DTLZ5, DLTZ6 and DTLZ7 (for almost all algorithms). In other cases, the improvement were small. However, from these generational snapshots, the actual performance gain can not be explicitly identified. For this purpose we need to measure the performance and conduct detailed statistical analysis using different indicators.

5.4 Performance Indicators

It was previously mentioned that there are two distinct goals in MOP: (i) discover solutions as close to the Pareto-optimal front as possible, and (ii) find solutions as diverse as possible in the obtained non-dominated front. In some aspects, these two goals are *orthogonal* to each other. The first goal requires search *towards* the Pareto-optimal region, while the second goal requires search *along* the Pareto-optimal front (refer to Figure 5.1) [1]. In this thesis, a diverse set of solutions is meant to represent a set of solutions covering the entire Pareto-optimal region uniformly. The measure of diversity can also be separated in two different measures of *extent* (i.e. the spread of extreme solutions) and *distribution* (i.e. relative distance among solutions) [2].

For analysis purposes, we have chosen two performance indicators used in the literature, known as **"Hypervolume Indicator"** [157], **"Epsilon Indicator"** [158]. We have also provided **"Attainment Surface Plot"** [159] for every algorithm. In the following sections we discuss the role of these metrics when measuring the performance.

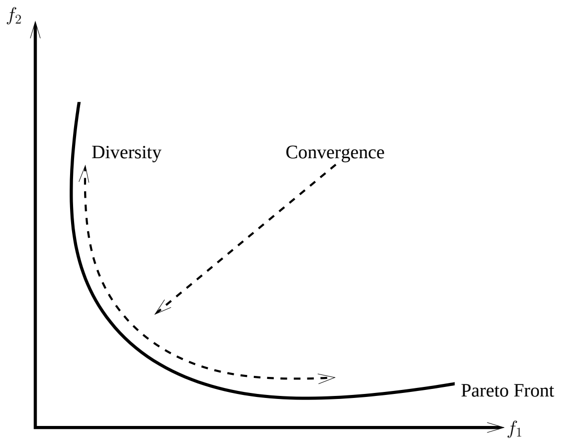


Figure 5.1: The aspect of MOEA performance metrics: diversity and convergence

5.4.1 Hypervolume Indicator

"Hypervolume" measures both diversity and convergence of an MOEA. This concept was first proposed in [157]. This indicator measures the hypervolume of that portion of the objective space that is weakly dominated by an approximation set A, and is to be maximized (refer to Figure 5.2). In order to measure this quantity, the objective space must be bounded. If it is not, then a bounding reference point that is (at least weakly) dominated by all points should be used, as shown as $\mathcal{W}$ in the Figure 5.2.

Note that, one can also consider hypervolume difference to a reference set R, this is referred as I_H^-. Given an approximation set A, the corresponding hypervolume is defined as -

$$I_H^-(A) = I_H(R) - I_H(A) \tag{5.1}$$

Where, for each solution $i \in A$, a hypercube v_i is constructed with a bounding point $\mathcal{W}$ (refer to Figure 5.2) and the solution i as the diagonal corners of the hypercube. Thus the hypervolume of a set A is calculated as -

$$I_H(A) = volume(\bigcup_{i=1}^{|A|} v_i) \tag{5.2}$$

The lower value of $I_H^-(A)$ indicates that the set R does not have better convergence and distribution along the Pareto-front than the set R. In our experiment, we have con-

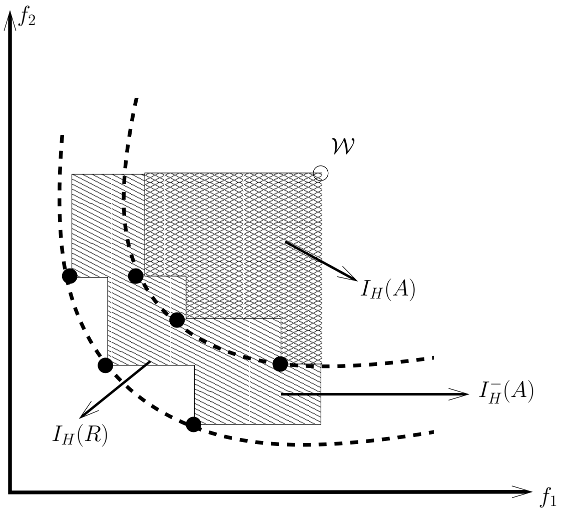


Figure 5.2: Difference of hypervolume from set A to R

sidered the Pareto-front generated by the hosting algorithm as A and the front generated by the PFVO integrated hosting optimizer as B (i.e. if the value of $I_H(B)$ is lower than $I_H(A)$, it will indicate the performance gain of the hosting optimizer due to integration of PFVO. Details related to this indicator can be found in [157], [159].

5.4.2 Epsilon Indicator

There are two ways in which the hypervolume indicator, used on its own, can be misleading. First, there is no way, from looking at I_H values in isolation, of inferring whether one set is actually better than another in a strict sense. Second, the choice of bounding point $\mathcal{W}$ is rather arbitrary, and this can affect the ordering of some pairs of sets. A method that avoids these particular difficulties is the **"Additive Binary Epsilon Indicator"** [158]. This takes a pair of non-dominated sets A and B and returns a pair of numbers ($I_{\epsilon+}(A)$, $I_{\epsilon+}(B)$).

$$I_{\epsilon+}(A) = I_{\epsilon+}[A,B] = \inf_{\epsilon\in\Re}\{\forall z^2 \in B, \quad \exists z^1 \in A : z^1 \preceq_{\epsilon+} z^2\} \tag{5.3}$$

$$I_{\epsilon+}(B) = I_{\epsilon+}[B,A] = \inf_{\epsilon\in\Re}\{\forall z^2 \in A, \quad \exists z^1 \in B : z^1 \preceq_{\epsilon+} z^2\} \tag{5.4}$$

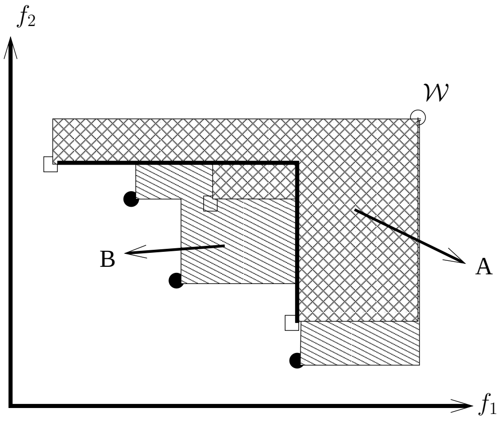


Figure 5.3: Two incomparable set A and B. Under the hypervolume B is better. But under epsilon indicator, A is better with respect to the reference set (the two (□) points are connected by the bold line), since $I_{\epsilon+}(A) = 1$ and $I_{\epsilon+}(B) > 1$. This discrepancy indicates hat the two sets must be incomparable.

Where, $z^1 \preceq_{\epsilon+} z^2$ iff $\forall j \in \{1, 2, \ldots k\} : z^1_i \leq \epsilon + z^2_i$, assuming minimization. (Note, $z^1 \preceq_{\epsilon+} z^2$ is read as z^1 ϵ-dominates z^2). A pair of numbers ($I_{\epsilon+}(A) \leq 0$, $I_{\epsilon+}(B) > 0$) indicates that A is strictly better than B [158]. A pair of numbers ($I_{\epsilon+}(A) > 0$, $I_{\epsilon+}(B) > 0$) indicates that neither set is strictly better than the other – they are incomparable [158]. However, if $I_{\epsilon+}(A)$ is less than $I_{\epsilon+}(B)$, then in a weaker sense, it is better because the minimum ϵ value needed so that approximation set A "ϵ-dominates" B is smaller than the value needed for to B "ϵ-dominates" A. There is also a "multiplicative" version of this indicator, here we have discussed the "additive" version that calculates the value of ϵ which should be "added" to make the two sets identical. In the case of "multiplicative" version, ϵ value would be "multiplied" to make the two sets identical. We have used both version in our experiments. More detailed analysis of this indicator can be found in [159]. We have used the PISA performance assessment library (`http://www.tik.ee.ethz.ch/sop/pisa/`) for our experiments.

5.4.3 Attainment Surface Plot

An **"Attainment Surface"** is the surface uniquely determined by a set of non-dominated points that divides the objective space into the region dominated by the set and the region that is not dominated by it [159]. Given n runs of an algorithm, it would be

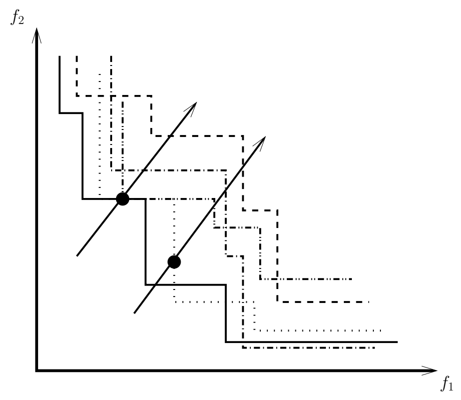


Figure 5.4: Five attainment surfaces are shown, representing the output of five runs of an optimizer. The two diagonal lines intersect the five surfaces at various points; in both cases, the circle indicates the intersection that weakly dominates at least $5 - 3 + 1 = 3$ surfaces and is also weakly dominated by at least three surfaces. Therefore, these two points both lie on the third summary attainment surface.

nice to summarize the results of the attainment surfaces obtained, using only one or two *summary* surfaces. Such summary attainment surfaces can be defined by imagining a diagonal line in the direction of increasing objective values cutting through the *n results* attainment surfaces (see Figure 5.4). The intersection on this line that weakly dominates at least $n - p + 1$ of the surfaces and is weakly dominated by at least of them, defines one point on the "p summary attainment surface". This surface is the union of all the goals that have been attained in at least p runs (independently). More details of the **Attainment Surface Plot** can be found in [73], [159], [160], [161]. For attainment surface plot, we have used the free software library available at: `http://dbkgroup.org/knowles/plot_attainments/`.

5.4.4 Statistical Analysis

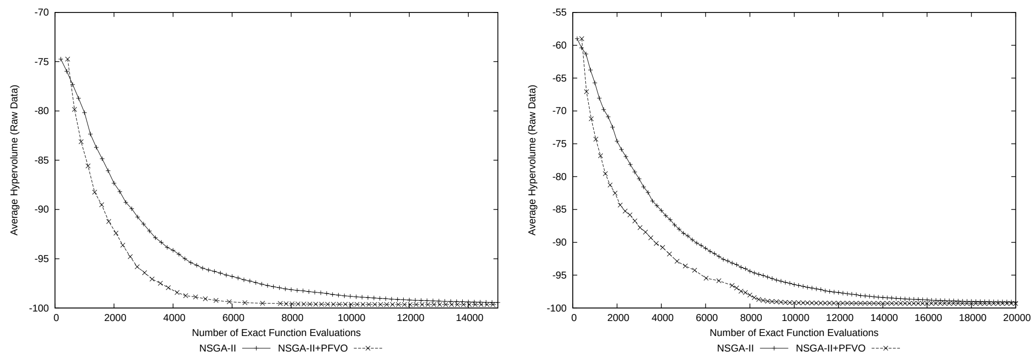
As we are dealing with a pair of algorithms (original optimizer and PFVO integrated optimizer), it is not always possible to infer that the modified algorithm is performing better than the original one. Even when the modified algorithm visually suggests a better performance measure. To make a stronger claim, we need to conduct statistical inference tests. There are numerous ways to perform such tests, however, it is another challenge to pick the best statistical inference methods. As suggested in [159], when we wish to compare multiple algorithms with multiple performance indicators, the **Kruskal-Wallis test** is appropriate. In our experiments, we have run each pair of the algorithms for 30 times and we have collected the value of Hypervolume and Epsilon (both additive and multiplicative) indicators at every generation (function evaluations) and conducted the Kruskal-Wallis test. This test implements a non-parametric test for differences between multiple independent samples. If and only if a first test for significance of any differences between the samples is passed, at the given α value (we have chosen $\alpha = 0.05$), then the output will be the one-tailed p-values for each pair-wise combination. For more details on statistical inference test, readers are referred to [162]. We have also used PISA statistics package (`http://www.tik.ee.ethz.ch/sop/pisa/`) for this purpose.

5.5 Performance Results: Hypervolume and Epsilon Indicators

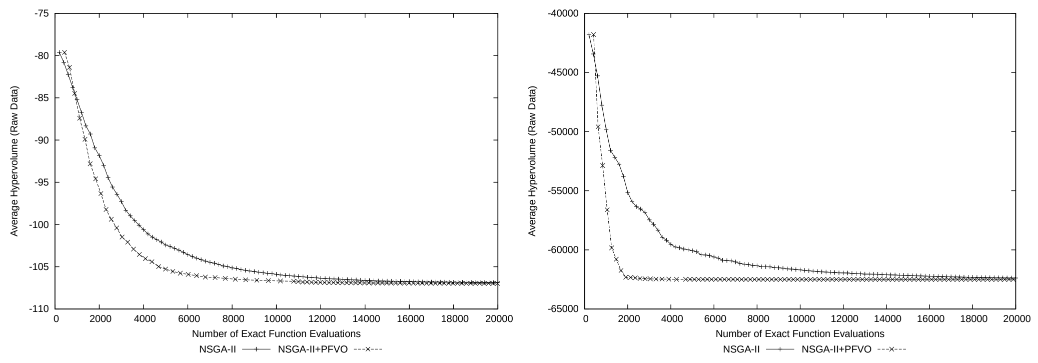
In this section we present Hypervolume (I_H) and Additive Epsilon ($I_{\epsilon+}$) indicator results of each algorithm (both stand alone and integrated version) on all problems. We have executed the algorithm pairs using same set of random seeds[20] for 30 times and the indicator values were averaged. The indicator values are presented with respect to the number of exact function evaluations (FE) so that we can make clear idea about the convergence gain. Please note that lower values of I_H indicates better front, so the algorithm which shows a lower value of I_H in a smaller number of FE is considered to be better. On the other hand, $I_{\epsilon+}(PFVO, HostOptimizer) \leq 0$, $I_{\epsilon+}(HostOptimizer, PFVO) > 0$ indicates "Host Optimizer + PFVO" is better than "Host Optimizer" (and vice-versa). If $I_{\epsilon+}(PFVO, HostOptimizer) > 0$, $I_{\epsilon+}(HostOptimizer, PFVO) > 0$; they are incomparable. We have also included the box plots and multiplicative Epsilon indicator plots in Appendix A. In every plot, (+) represent the values found from "Host Optimizer" and ($\times$) represent the values found from "Host Optimizer + PFVO".

[20]We have used the same pair of random seeds for the original algorithm and the PFVO enhanced algorithm. But the pair of seeds were different at 30 distinct runs

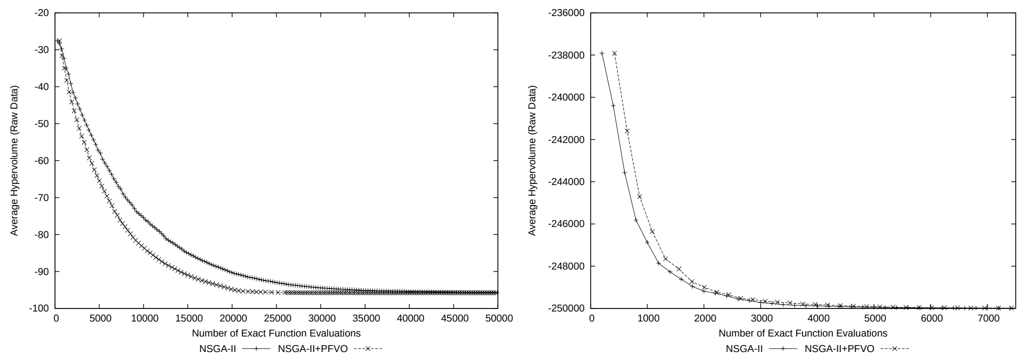
5.5.1 NSGA-II



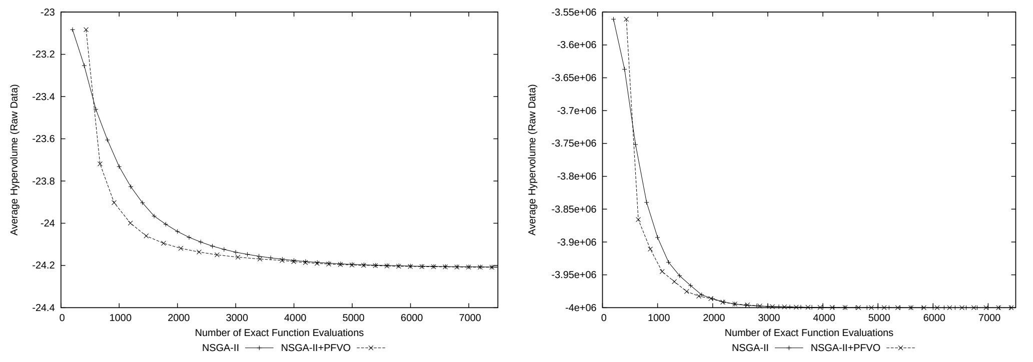
(a) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT1(Left) and ZDT2(Right)



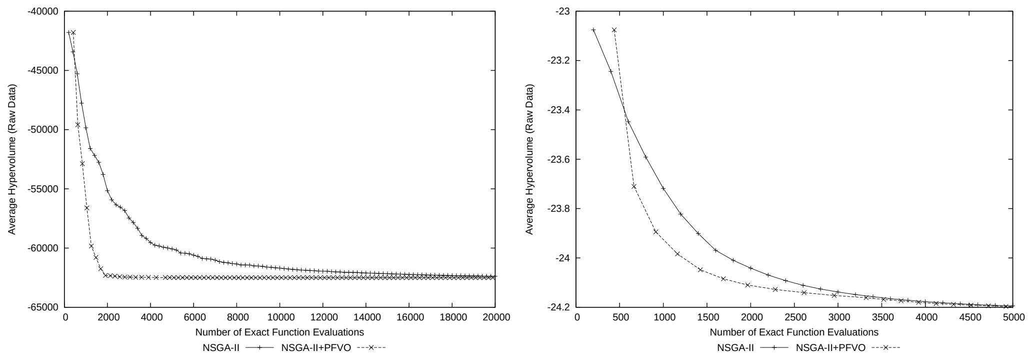
(b) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT3(Left and ZDT4(Right))



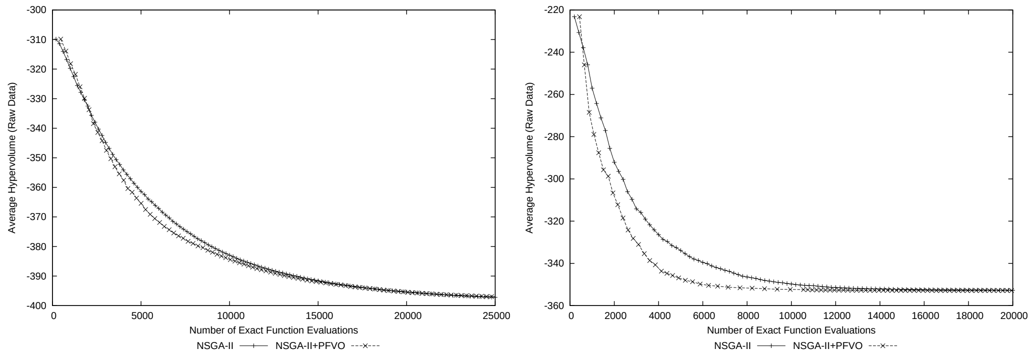
(c) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



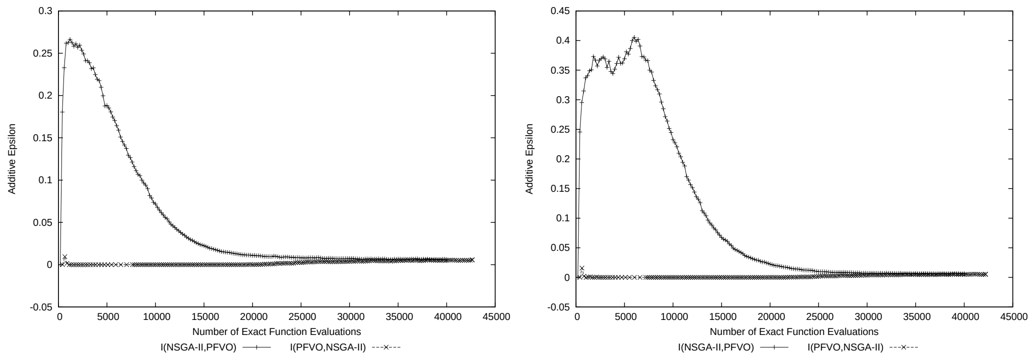
(d) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



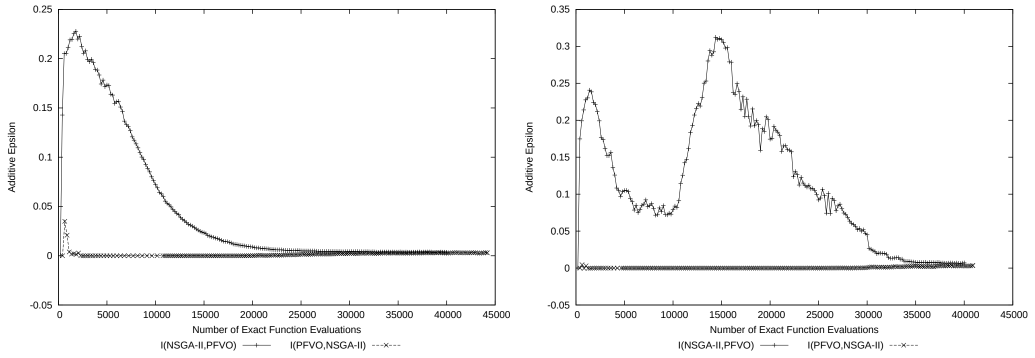
(e) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



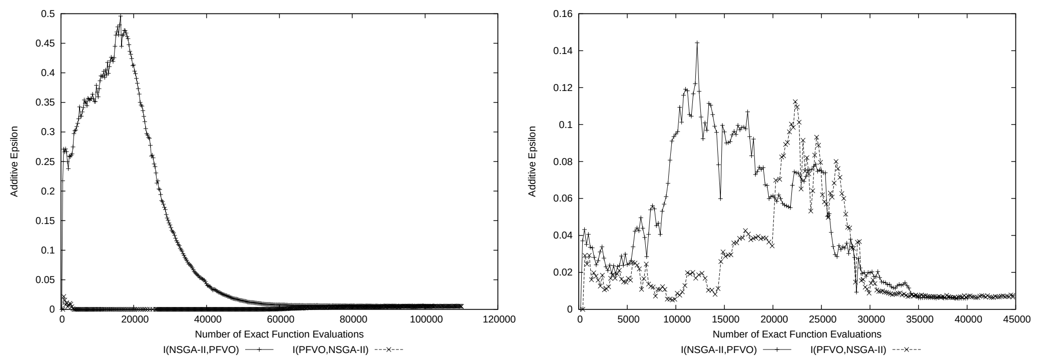
(f) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



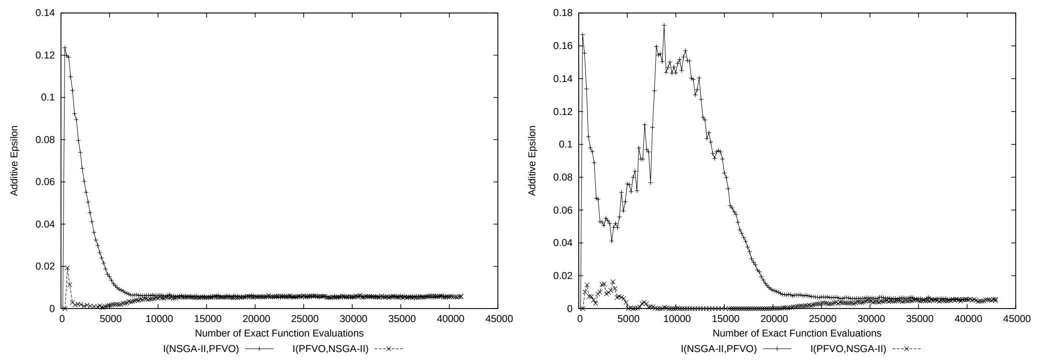
(g) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



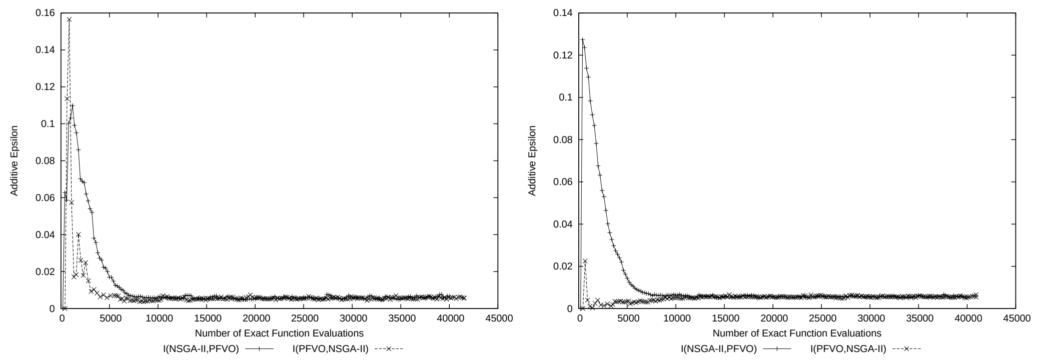
(h) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



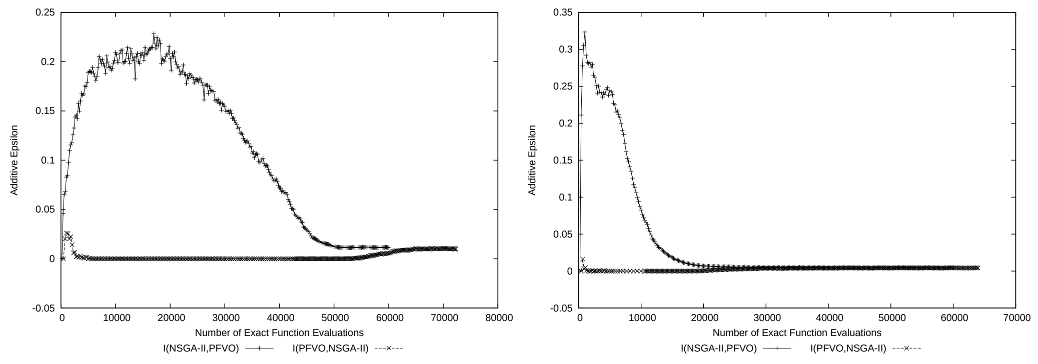
(i) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(j) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ2 (Left) and DTLZ3(Right)

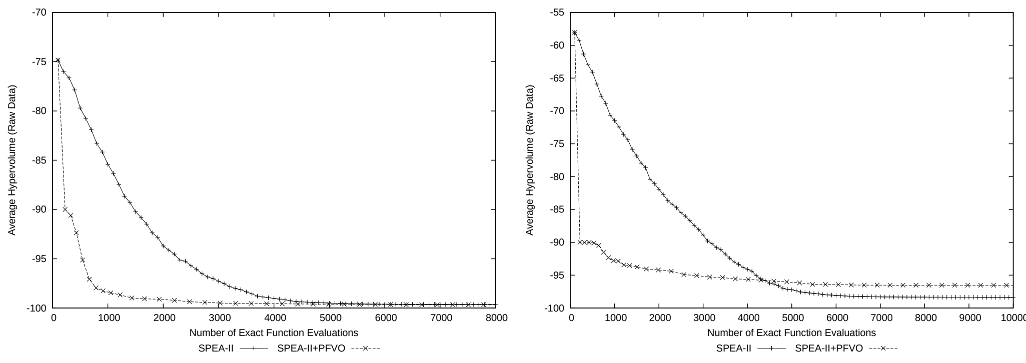


(k) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ4 (Left) and DTLZ5(Right)

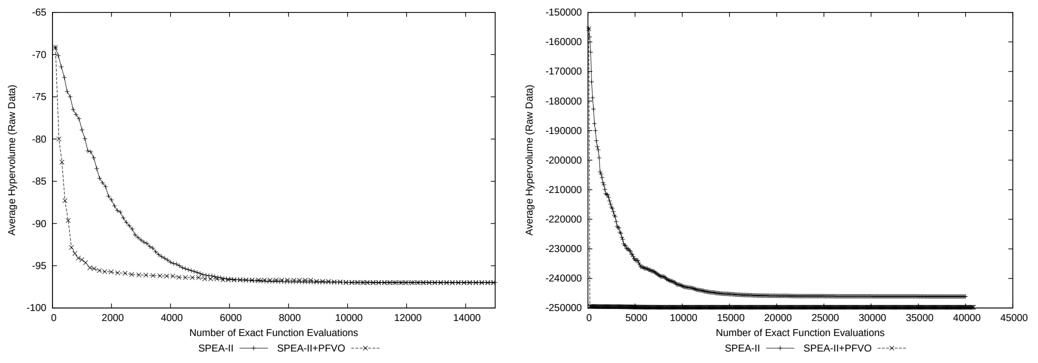


(l) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ6 (Left) and DTLZ7(Right)

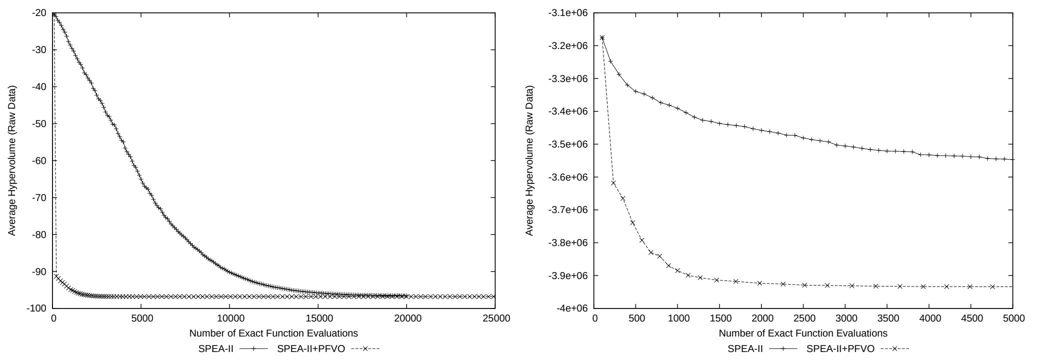
5.5.2 SPEA-II



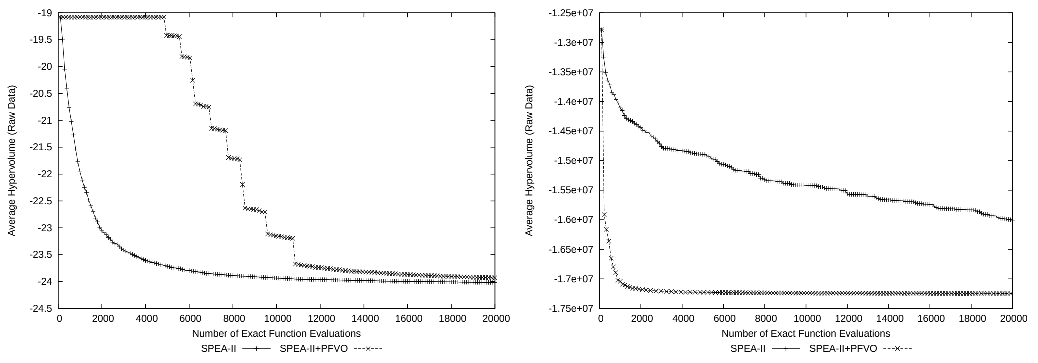
(a) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



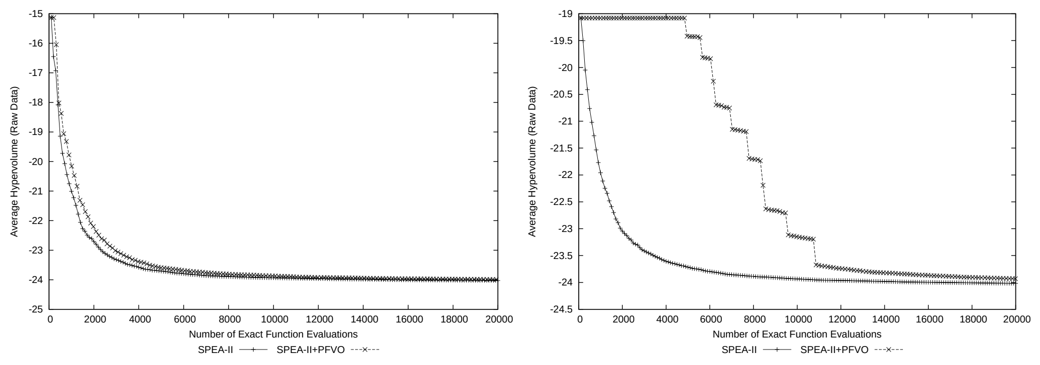
(b) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



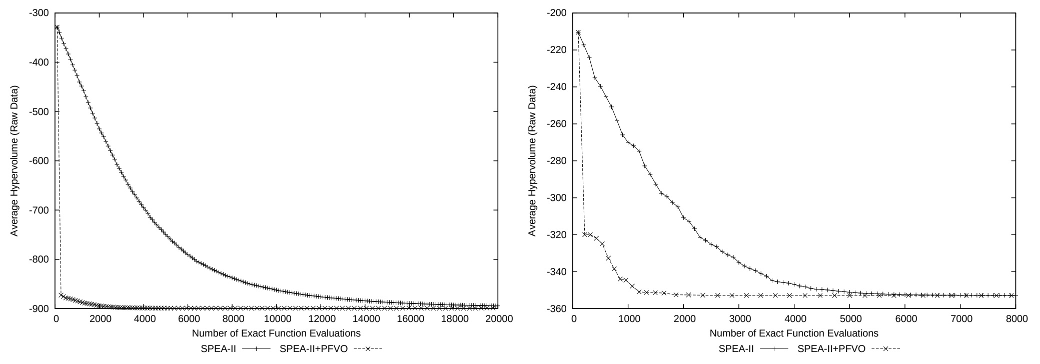
(c) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



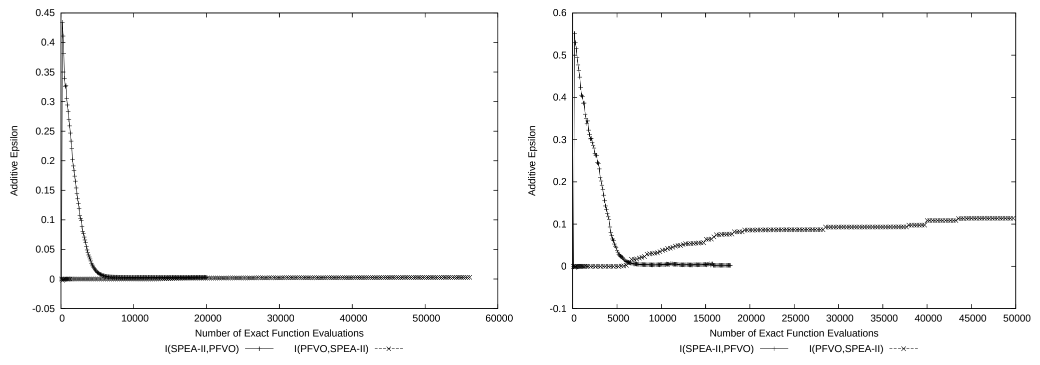
(d) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



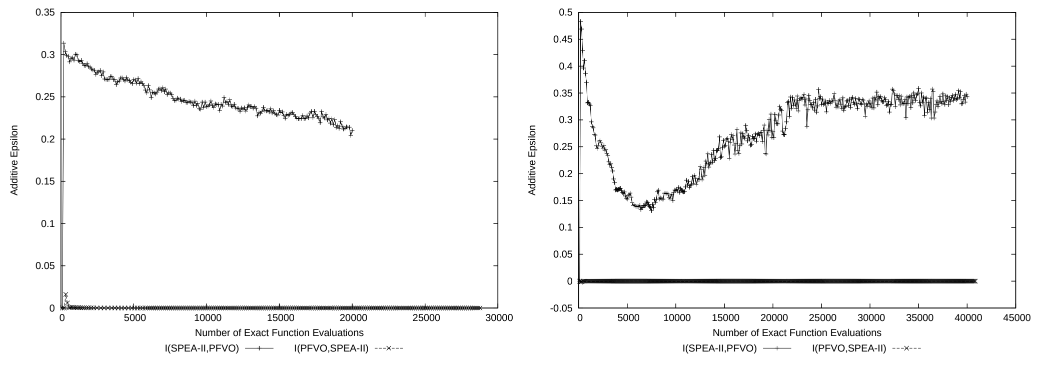
(e) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



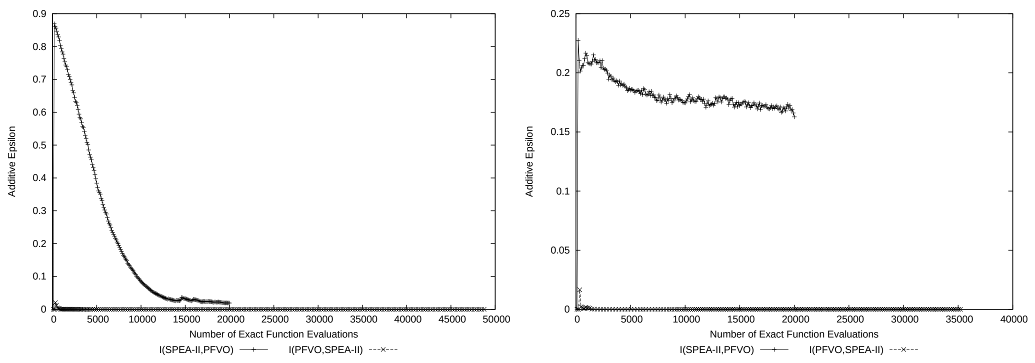
(f) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



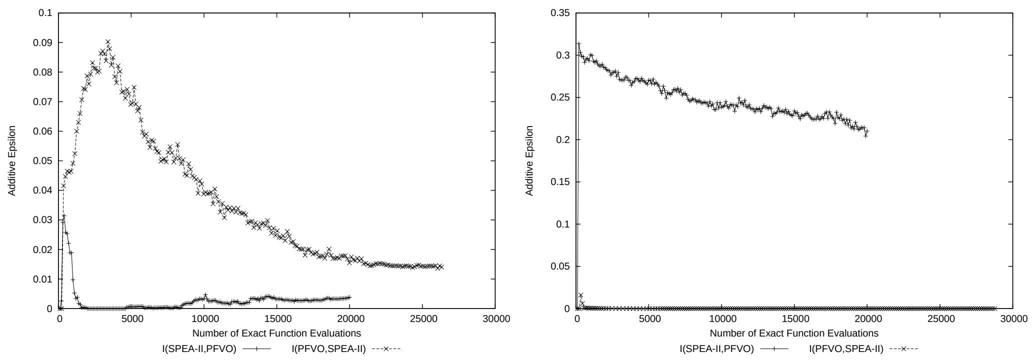
(g) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



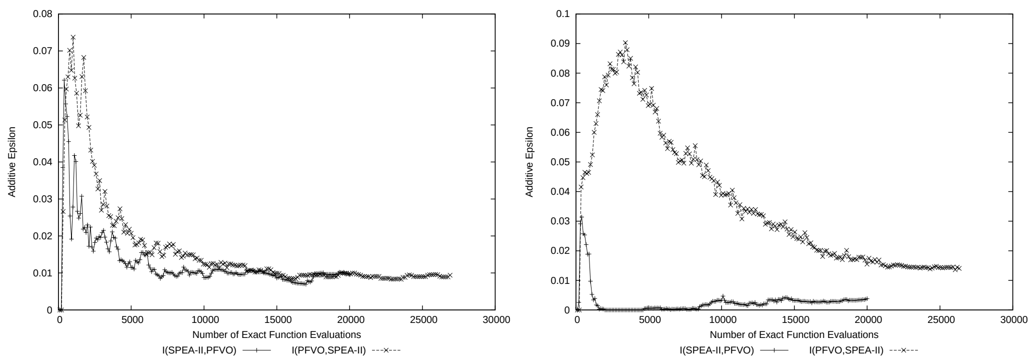
(h) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



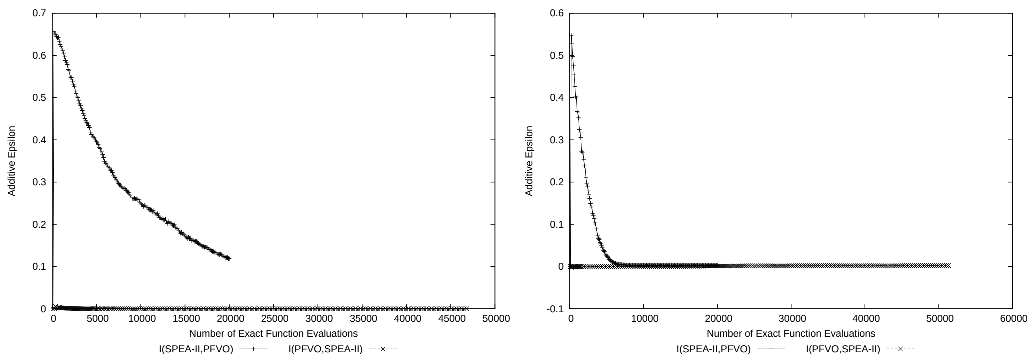
(i) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(j) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)

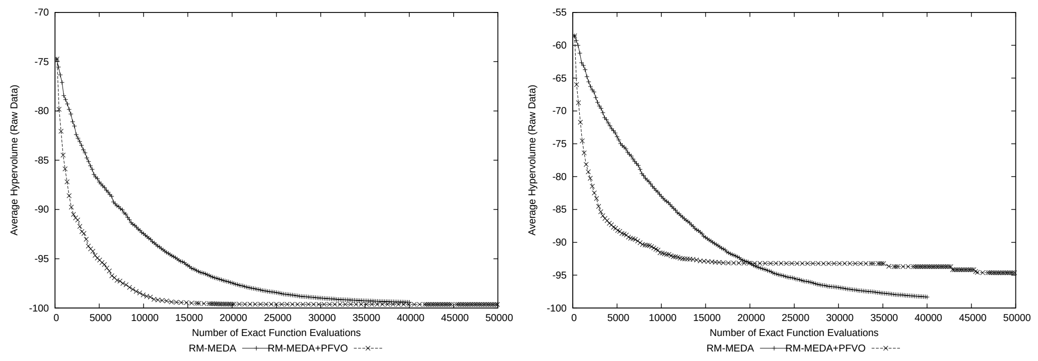


(k) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)

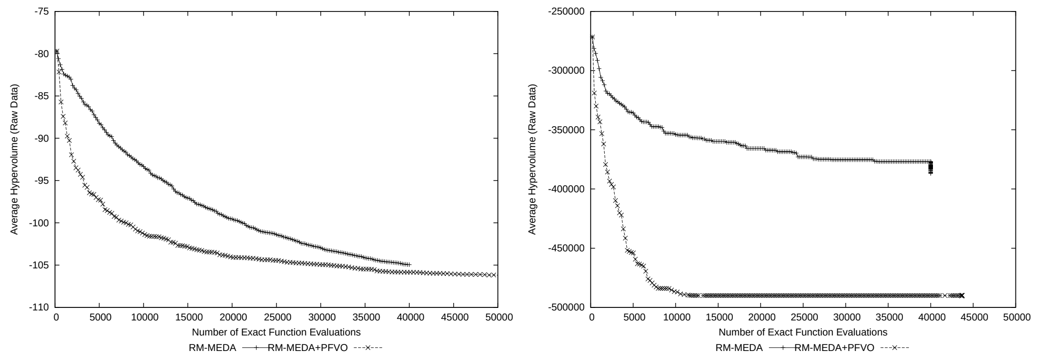


(l) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

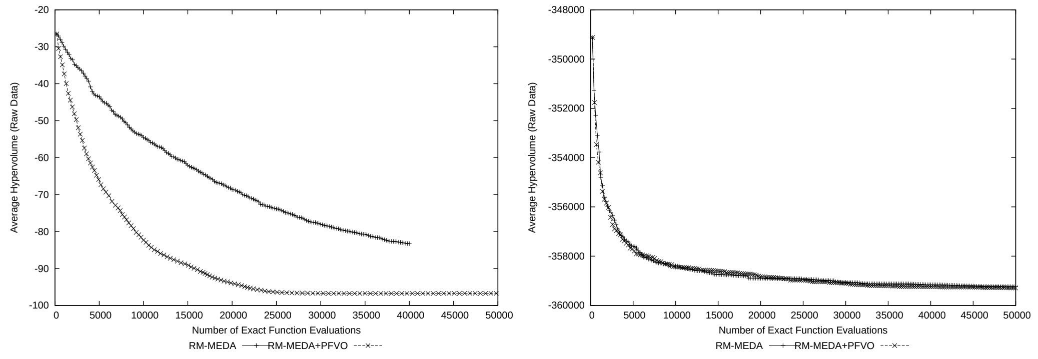
5.5.3 RM-MEDA



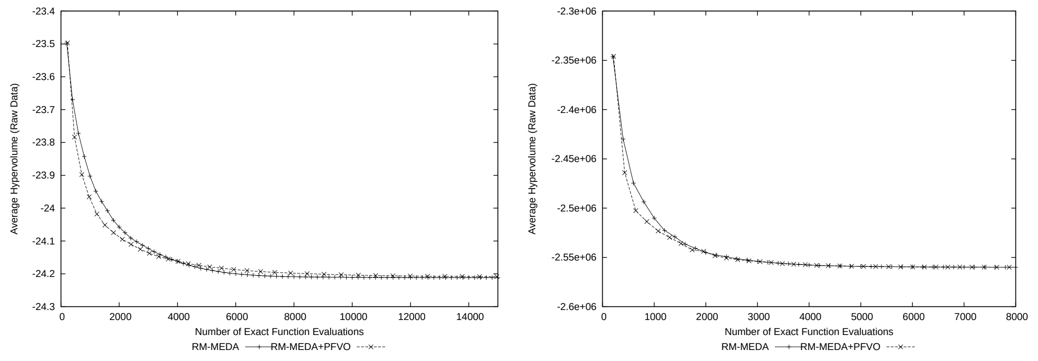
(a) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT1 (Left) and ZDT2 (Right)



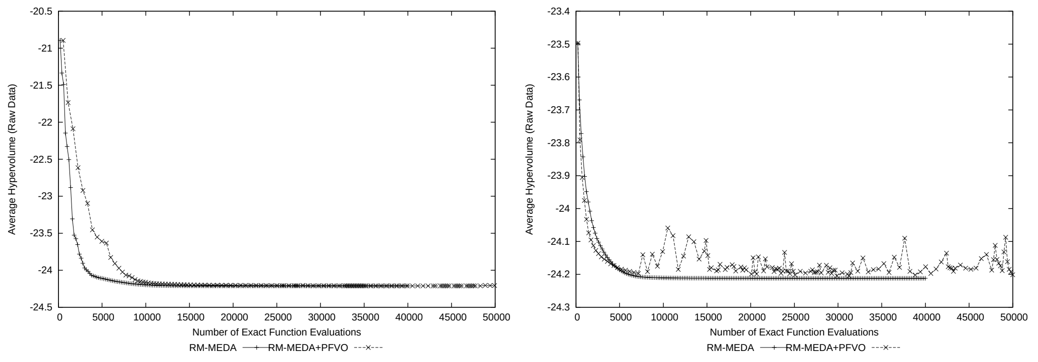
(b) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT3 (Left) and ZDT4 (Right)



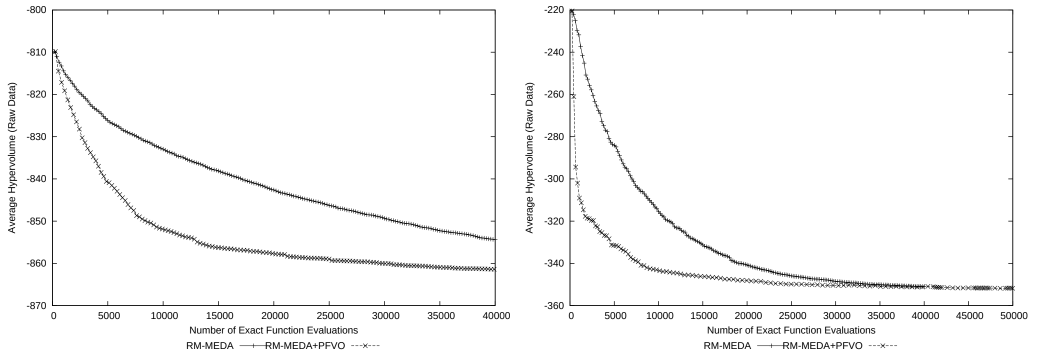
(c) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



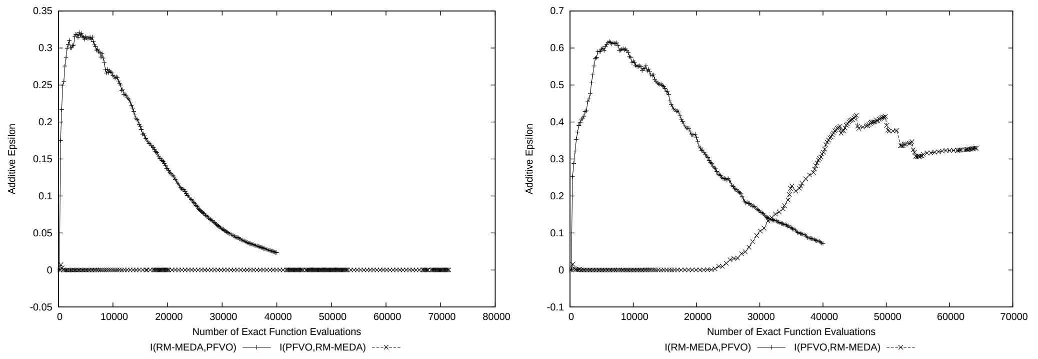
(d) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



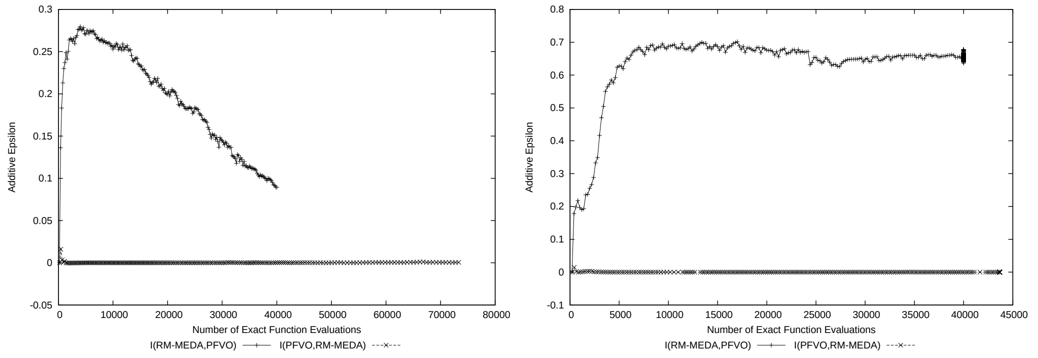
(e) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



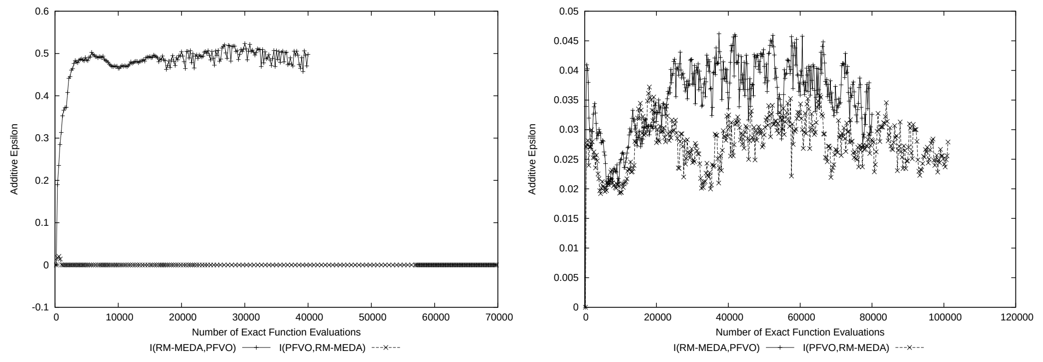
(f) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



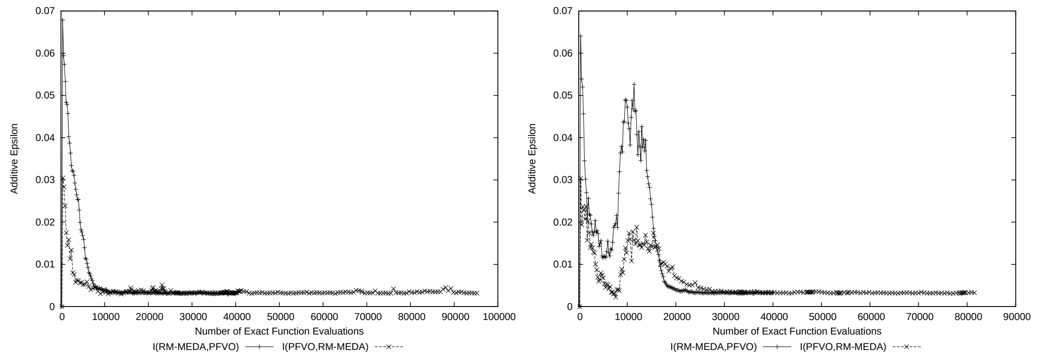
(g) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT1 (Left) and ZDT2 (Right)



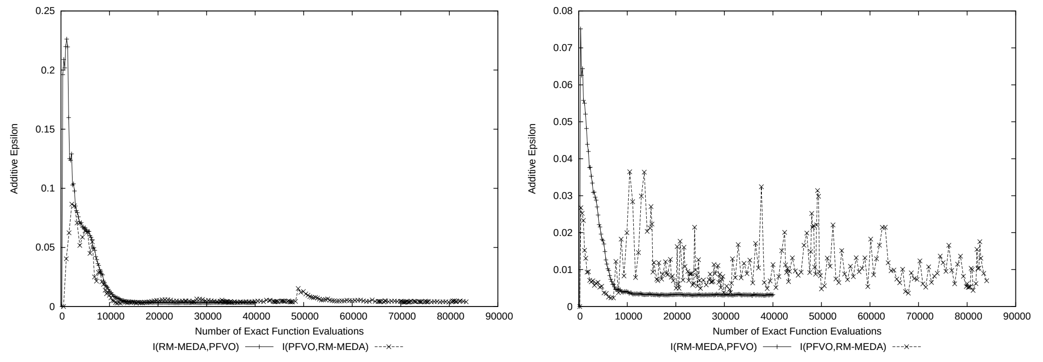
(h) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT3 (Left) and ZDT4 (Right)



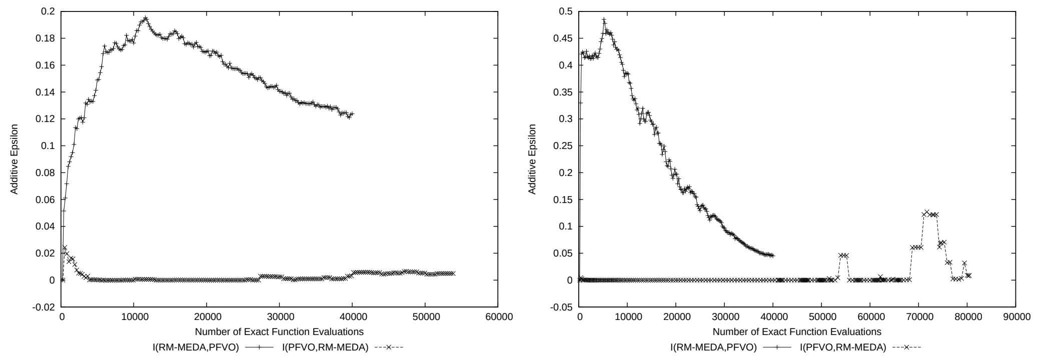
(i) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(j) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



(k) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



(l) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

5.5.4 Analysis

The I_H indicator plots provide us a clearer estimation of the convergence gain of the hosting optimizer. Specifically, in the case of "NSGA-II + PFVO" on the ZDT test suite, the convergence is significantly better (smaller I_H values in less function evaluations), on DTLZ3, DTLZ4, DTLZ5 and DTLZ7, we see same extent of improvement. For "SPEA-II + PFVO", the improvement is more promising, for most of the problems (except DTLZ2, DTLZ4 and DTLZ5) the convergence speed is significantly improved. For "RM-MEDA + PFVO", the result is similar to "NSGA-II + PFVO", the integration improve the performance on ZDT test suite. On the other hand, for DTLZ6 and DTLZ7, we also found significant improvement. On DTLZ5, I_H indicator starts to fluctuate very frequently after 7500 function evaluation. On DTLZ2, the integrated version shows better I_H until 24000 function evaluations and after that it gets worse.

For "NSGA-II + PFVO", the performance on DTLZ1 and DTLZ6 was not good in terms of I_H however the I_ϵ indicator value suggests that the integrated version performing better. For "SPEA-II + PFVO" we do not find such a discrepancy, its performance is not better in case of DTLZ2, DTLZ4 and DTLZ5 (same as I_H indicator value). For "RM-MEDA + PFVO" the $I_{\epsilon+}$ reflects similar performances on DTLZ1, DTLZ2, DTLZ3 and DTLZ5 as I_H.

For statistical analysis, we have executed each pair of algorithms on every problems for 30 times and the I_H and $I_{\epsilon+}$ values were collected for every generation. From these values, we have conducted the Kruskal-Wallis [162] test by setting α values at 0.05 (i.e. 95% significance). The detailed p-value results are provided in Appendix - A.

5.6 Approximation Accuracy

An important point should be taken into account when an approximation scheme is used for any kind of numerical optimization algorithm ; that is the notion of "approximation accuracy". "Approximation Accuracy" means the distance from the predicted values on the future Pareto-front and the values calculated by original fitness function. We have seen that PFVO can generate new promising solutions (fitness values in objective space)

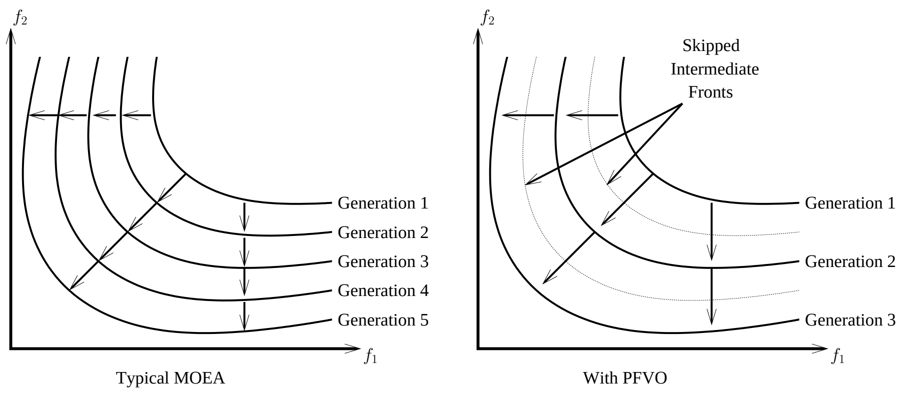


Figure 5.5: After integration of PFVO, the hosting optimizer can skip the intermediate fronts to reach the Pareto-front, thus the rate of convergence is increased.

in such a way that most of the predicted/re-calculated solutions lie on the next Pareto-front (i.e. most of the calculated solutions by PFVO are non-dominated to the previous front). In the case of PFVO, the approximation accuracy is not as good compared to ANN based models. However, the fact is that, for single objective optimization, the accuracy of the approximation scheme is a crucial concern, whereas, for MOEA, it is not so important. Since MOEA deals with a multiple number of solutions ; so, if we (somehow) could get a "rough" estimation of the promising solutions of the next (future) front from the existing ones, the hosting optimizer can utilize them to explore further (in a right direction). Subsequently, the hosting optimizer can skip the evaluation of solutions from some of the intermediate fronts in order to reach the Pareto-optimal. We think this is the possible explanation for the reported "gain in convergence speed".

Moreover, if the hosting optimizer is capable of exploring more promising solutions from the given estimated set, we think the use of highly complex approximation model (i.e. ANN) is not necessary. To clarify this issue, please refer to Figure 5.5. However, if the hosting optimizer lacks the ability to sample multiple number of good solutions by dividing them in to "Fronts", PFVO may not lead to an increased convergence rate[21]. However, the approximation accuracy plot (from a snapshot of a specific generation) for NSGA-II with PFVO on ZDT1 is presented in Figure 5.6.

[21] In our experiment we have also integrated PFVO with **"Pareto Archived Evolutionary Strategy (PAES)"** [44], [45] but we have found no significant difference with the original algorithm and the PFVO enhanced model. PAES is actually an **"Evolutionary Strategy (ES)"** based model and its working scheme is totally different from NSGA-II or SPEA-II.

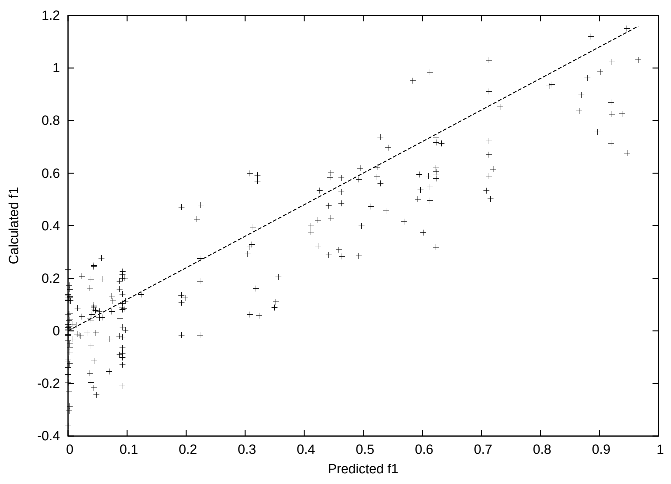


Figure 5.6: The predicted $f_1 - \Delta f$ and calculated $f_1 - \Delta f$. Here we have used fixed values of $\Delta f = 0.2$ to test the approximation accuracy. All the points under the straight line define promising solutions (moved towards the Pareto-front). Approximation is fair however the gain of promising solutions is significant

5.7 Limitations

Although PFVO can successfully speed up the convergence rate of an MOEA, it has some shortcomings as described below -

5.7.1 Singular Matrix

Since we are using QR factorization to calculate the parameters of dynamic system (Optimizer) β_{ij}, sometimes it becomes impossible to calculate this value due to the formation of singular/non-invertible matrix. In such cases, we had to discard the matrix $\mathbf{\Phi}$ and $\mathbf{y}$ and start with the next set of $\mathbf{\Phi}$ and $\mathbf{y}$. To solve this problem, a more stable method such as **"Singular Value Decomposition (SVD)"** could be used.

5.7.2 Scalability

Readers have already noted that we are conducting all our experiments on only bi-objective problems. The fact is that PFVO can not be directly applied to problems with any number of objectives (i.e. more than two). If we are going to consider more than two objectives, we have to apply PFVO on pair-wise combinations of objectives[22]. So, it will increase the computational cost with a scale of $\begin{pmatrix} k \\ 2 \end{pmatrix} \mathcal{O}(nkN^3)$.

5.8 Conclusion

In this chapter, we have tested our model with three popular MOEA's in the literature. We have also conducted extensive performance measure and corresponding statistical analysis. We have found that PFVO can increase the convergence rate of NSGA-II, SPEA-II and RM-MEDA. In the next chapter apply the PFVO on **"Dynamic MOP"**. In the case of dynamic MOP, the Pareto-front changes with respect to time (or some other variable) to make hard for any MOEA to track correct Pareto-optimal front. Since our claim is that PFVO can increase the convergence rate of a MOEA, we suggest that it will also be able to track the change in Pareto-front more promptly than other algorithms.

[22]More precisely, we have to make combination of two objectives from k (i.e. $\begin{pmatrix} k \\ 2 \end{pmatrix}$ ways) and apply PFVO on every pair.

Chapter 6

The Pareto Following Variation Operator for Dynamic MOP

6.1 Introduction

In the previous chapter, we have seen how the Pareto Following Variation Operator can improve the convergence speed of typical MOEA. Although, PFVO seems to work for MOP's, however to investigate further, we have also applied PFVO for dynamic MOP's (DMOP). In the case of DMOP's, the Pareto-front is not stable; rather it changes with respect to another variable (generally time or generation cycle count). Our claim is that it is possible to reach the Pareto-front with better speed using PFVO. This raises the question; is it possible to track the changes in Pareto-front more promptly (specially in dynamic/uncertain environment).

In this chapter we focus on this issue. Recently optimization in dynamic/uncertain/noisy environment has become a "hot" topic due to its wide spread applicability (e.g. real time optimal control, on-line scheduling problems etc.). Although there have been many successful studies of static multi-objective problems, the number of papers focussed on dynamic problems is relatively small (for example, see [163], [164], [165], [166], [167], [168]). Inspired by the success of evolutionary algorithms when tackling dynamic scalar optimization problems [169], [170], [171], [172], a number of researchers have proposed and evaluated evolutionary algorithms for multi-objective optimization in dynamic and noisy environments [173], [174], [175]. Many algorithms for dynamic MOP's are extensions of algorithms designed for static problems. The key difference is that the dynamic counterparts simply tries to "re-start" and search for the new global optimum when there is a change in the fitness function(s) [163], [168]. Moreover, the incorporation of "forecasting models" into the dynamic algorithms has also been the subject of investigations [164], [166][23].

When a broad range of both static and dynamic MOP's are considered, there is no single algorithm that can work well for both of the problem classes. Clearly, the design and effectiveness of the variation operators are pivotal to the success of any evolutionary algorithm. In the case of static problems, specific variation operators (SBX [135], UNDX [132] and EDX [133]) have been designed to meet the constraint functions imposed by a particular problem. However, they typically do not have a "Pareto Following" property. Although there have been a few attempts to incorporate this property [56], [58], [55], [75], these models are only applicable to problems with **differentiable objective functions**.

6.2 Dynamic Multiobjective Optimization Problems (DMOP)

In the previous chapters we have already seen the basic formulation of MOP's. As in the case of MOP's, the benchmark DMOP's have not been extensively studied. Some trivial concepts on generating benchmark DMOP's are discussed in [176], however the simplicity of problems in [176] did not encourage us to investigate further. Rather we have chosen the benchmark problems provided in [177].

In the case of DMOP's, the objective vector $\mathbf{f}(\mathbf{x})$, the constraints, $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ will vary with respect to another variable t (generally t is time). Thus, the objective and the constraints are replaced by $\mathbf{f}(\mathbf{x}, t)$, $g_i(\mathbf{x}, t)$ and $h_i(\mathbf{x}, t)$ respectively. For such problems, the goal is to find all non-dominated fronts which vary with time. In the case of DMOP there are four possible ways a problem can demonstrate a time-varying change [177].

- Type I) The optimal design variables $\mathcal{PS}$ changes, whereas the optimal objective values $\mathcal{PF}$ does not change.
- Type II) Both and change.
- Type III) $\mathcal{PS}$ does not change, whereas $\mathcal{PF}$ changes.
- Type IV) Both $\mathcal{PS}$ and $\mathcal{PF}$ do not change, although the problem can change.

[23]It is interesting to note that [164] uses ANN for forecasting.

As we have discussed in previous chapter, PFVO is capable of approximating the design variables of the next front from the current and previous objective values. As a consequence, PFVO is suitable for Type-III problems, where the optimal design variable $\mathcal{PS}$ does not change with t.

6.3 Modeling the PFVO for Dynamic MOP's

The concept of dynamic system identification only works on **"Linear Time Invariant System (LTI)"** [149], [150], [178]. **"Linearity"** suggests that input and the output of the system satisfies the *superposition property*. If the input to the system is the sum of two component signals (we are using the same notations described in chapter 4) -

$$x^p(\phi) = c_i x_i^p(\phi) + c_{i+1} x_{i+1}^p(\phi) + \dots \tag{6.1}$$

Then output of the system will be

$$f^p(\phi) = c_j f_j^p(\phi) + c_{j+1} x_{j+1}^p(\phi) + \dots \tag{6.2}$$

Where c_i and c_j are arbitrary constants. Generally during the evolutionary search, the distance (i.e. convergence step) between the solutions[24] in cartesian co–ordinate (X–Y–Z co–ordinate system) are small, hence we can assume that the position of the design variables (input) and corresponding objective values (output) in different fronts during exploration are piece-wise linear. So the above condition holds for a specific values of the constants c_i and c_j.

On the other hand, **"Time Invariance"** means that whenever we apply an input to the "system" (i.e. "optimizer") now or T seconds from now (i.e. current front or T fronts from the current front), the output will be identical, except for a time delay of the T seconds (i.e. T front gap). If the output due to input $x_i^p(\phi)$ is $f_j^p(\phi)$, then the output due to input $x_i^p(\phi - T)$ is $y_j^p(\phi - T)$. More specifically, an input affected by a time delay (i.e. front gap) should cause a corresponding time delay (i.e. front gap) in the output, hence

[24]More specifically, the distance of the fronts generated by the single pass of the non-dominated sorting.

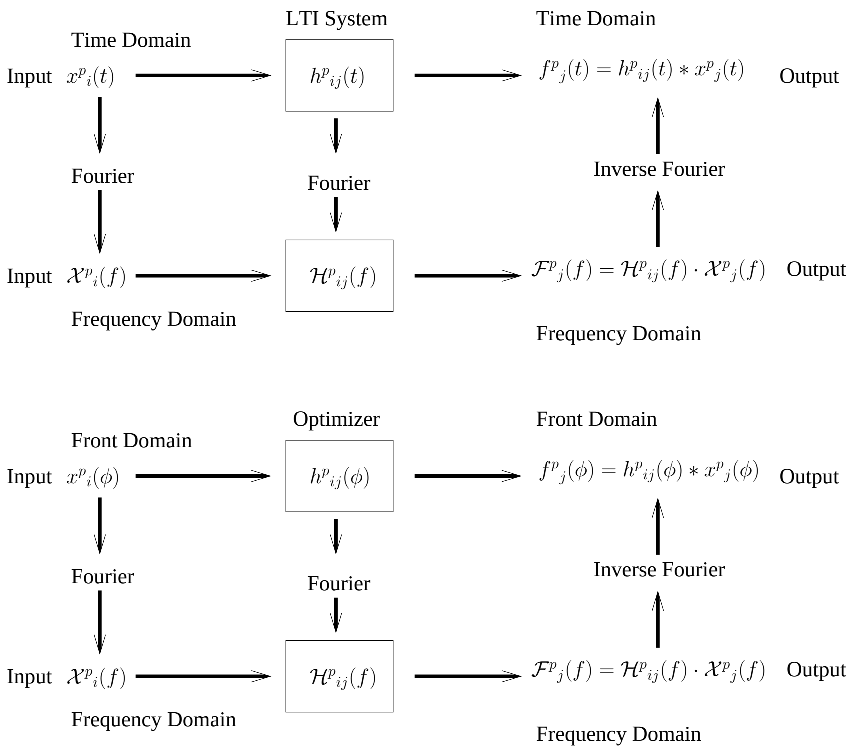

Time Domain
LTI System
Time Domain
Input $x^p{}_i(t)$
$h^p{}_{ij}(t)$
$f^p{}_j(t) = h^p{}_{ij}(t) * x^p{}_j(t)$
Output
Fourier
Fourier
Inverse Fourier
Input $\mathcal{X}^p{}_i(f)$
$\mathcal{H}^p{}_{ij}(f)$
$\mathcal{F}^p{}_j(f) = \mathcal{H}^p{}_{ij}(f) \cdot \mathcal{X}^p{}_j(f)$
Output
Frequency Domain
Frequency Domain
Front Domain
Optimizer
Front Domain
Input $x^p{}_i(\phi)$
$h^p{}_{ij}(\phi)$
$f^p{}_j(\phi) = h^p{}_{ij}(\phi) * x^p{}_j(\phi)$
Output
Fourier
Fourier
Inverse Fourier
Input $\mathcal{X}^p{}_i(f)$
$\mathcal{H}^p{}_{ij}(f)$
$\mathcal{F}^p{}_j(f) = \mathcal{H}^p{}_{ij}(f) \cdot \mathcal{X}^p{}_j(f)$
Output
Frequency Domain
Frequency Domain

Figure 6.1: Analogy between "Optimizer" and "System": concept of frequency domain and time domain interchange. $h_{ij}(\phi)/\mathcal{H}_{ij}(f)$ denotes transfer function of the optimizer for design variable i and objective j

time-invariant. It is trivial that for a specific values of c_i and c_j this assumption is true for any optimizer (i.e. "LTI system").

If the above conditions hold, then an optimizer (EA) can be replaced by an LTI system that takes design variables as input and generates objective values as output. The fundamental result in LTI system theory is that any LTI system can be characterized entirely by a single function called the system's **Impulse Response/Transfer Function** (which is denoted by $h(t)$ in time domain and $\mathcal{H}(f)$ in frequency domain). The output of the system is simply the **convolution** ("$*$" operator) of the input to the system with the system's Impulse Response/Transfer Function $h(t)$ (or $h(\phi)$ in front domain).

Equivalently, any LTI system can be characterized in the frequency domain by the system's transfer function, which is the **Laplace Transform** or **Fourier Transform** of the system's impulse response $h(t)$ (or **Discrete Fourier/Laplace transform** in the case of discrete-time systems). As a result of the properties of these transforms, the output of the system in the frequency domain is the product of the transfer function and the transform of the input. In other words, convolution in the time domain is equivalent to multiplication in the frequency domain. This concept is depicted in Figure 6.1.

As described in chapter 4, the remaining concepts are similar – once we know the transfer function $h(\phi)$ (or $\mathcal{H}(f)$) of the optimizer, the inverse realization of the transfer function $\mathcal{H}^{-1}(f)$ can be used to "inverse-map" the approximated objective functions to corresponding design variables. This procedure can be applied to any number of objectives/design variable combinations. Now we can devise the algorithm as follows -

```
**Algorithm 5** Approximate($[x_i^p(\phi), x_i^p(\phi-1), \ldots, x_i^p(1)], [f_j^p(\phi), f_j^p(\phi-1), \ldots, f_j^p(1)], \Delta f$)
**Require:** Design variable $i$ and objective values $j$ of the current and past fronts:
    $[x_i^p(\phi), x_i^p(\phi-1), \ldots, x_i^p(1)]$ and $[f_j^p(\phi), f_j^p(\phi-1), \ldots, f_j^p(1)]$ respectively.
**Ensure:** Calculates the design variable $i$ of the next front $x_i^p(\phi+1)$
1: Find $\mathcal{X}_i(f)$ from $[x_i^p(\phi), x_i^p(\phi-1), \ldots, x_i^p(1)]$ using forward Fourier Transform,
   $[x_i^p(\phi), x_i^p(\phi-1), \ldots, x_i^p(1)] \to \mathcal{X}_i(f)$
2: Find $\mathcal{F}_j(f)$ from $[f_j^p(\phi), f_j^p(\phi-1), \ldots, f_j^p(1)]$ using forward Fourier Transform,
   $[f_j^p(\phi), f_j^p(\phi-1), \ldots, f_j^p(1)] \to \mathcal{F}_j(f)$
3: Find the transfer function $\mathcal{H}_{ij}(f) = \frac{\mathcal{F}_j(f)}{\mathcal{X}_i(f)}$. Since $\mathcal{F}_j(f) = \mathcal{H}_{ij}(f) \cdot \mathcal{X}_i(f)$.
4: Approximate the next front $f_i^p(\phi+1) = f_i^p(\phi) - \Delta f$ using adaptive $\Delta f$ calculation as
   described in chapter 4.
5: Find $\mathcal{F'}_j(f)$ from $[f_j^p(\phi+1), f_j^p(\phi), \ldots, f_j^p(2)]$ using forward Fourier Transform
6: Find $\mathcal{X'}_i(f)$, $\mathcal{X'}_i(f) = \frac{\mathcal{F'}_j(f)}{\mathcal{H}_{ij}(f)}$
7: Apply inverse Fourier Transform, $[x_i^p(\phi+1), x_i^p(\phi), \ldots, x_i^p(2)] \leftarrow \mathcal{X'}_i(f)$
8: Returns new approximated design variable $i$: $x_i^p(\phi+1)$
```

6.3.1 Integration Mechanism

For integration, we have chosen the modified NSGA-II-A as the "Host Optimizer" [163]. In the case of modified NSGA-II-A, the algorithm undergo a second fitness evaluation process whenever there is a change in 't'. Typically, 10% of the parent population is chosen randomly and they are evaluated and replaced. However, we did not apply the second phase of fitness evaluation since PFVO itself is able to tackle the change in $\mathcal{PF}$ automatically. Due to the fact that the PFVO predicts the next front from the existing ones, expensive fitness function evaluation is not required every time 't' changes. Thus, PFVO can save on computational cost due to "restart" of the algorithm.

6.4 Benchmark DMOP's

The benchmark DMOP's that we have used in our experiments are listed in Table 6.1 and Table 6.2 – **FDA** problem suite.

The FDA Test Problem Suite

Problem	Definition	Parameter Domains	Type
FDA1	$f_1(\mathbf{x}_I) = x_1$ $g(\mathbf{x}_{II}) = 1 + \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$ $G(t) = \sin(0.5\pi t), \quad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor$	$\mathbf{x}_I = (x_1) \in [0,1]$ $\mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1,1]$	I
FDA2	$f_1(\mathbf{x}_I) = x_1$ $g(\mathbf{x}_{II}) = 1 + \sum_{x_i \in x_{II}} x_i^2$ $h(\mathbf{x}_{III}, f_1, g) = \left(1 - \frac{f_1}{g}\right)^{\left(H(t) + \sum_{x_i \in \mathbf{x}_{III}} (x_i - H(t))^2\right)^{-1}}$ $H(t) = 0.75 + 0.7\sin(\pi t), \quad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor$	$\mathbf{x}_I = (x_1) \in [0,1]$ $\mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1,1]$ $\mathbf{x}_{III} = (x_2, \ldots, x_n) \in [-1,1]$	III
FDA3	$f_1(\mathbf{x}_I) = \sum_{x_i \in \mathbf{x}_I} x_i^{F(t)}$ $g(\mathbf{x}_{II}) = 1 + G(t) + \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$ $G(t) = \lVert \sin(0.5\pi t) \rVert$ $F(t) = 10^{2\sin(0.5\pi t)}, \quad t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_T} \rfloor$	$\mathbf{x}_I = (x_1) \in [0,1]$ $\mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1,1]$	II

Table 6.1: Three of the five FDA DMOP's. All objectives are to be minimized

The FDA Test Problem Suite

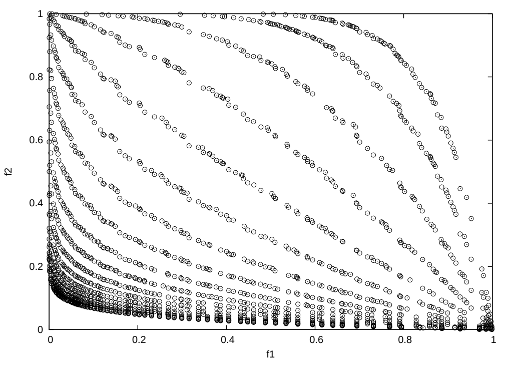
Problem	Definition	Parameter Domains	Type
FDA4	$f_1(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\prod_{i=1}^{M-1}\cos(\frac{x_i\pi}{2})$ $f_k(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\left(\prod_{i=1}^{M-k}\cos(\frac{x_i\pi}{2})\right)\sin(\frac{x_{M-k+1}\pi}{2})$ $f_M(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\sin(\frac{x_i\pi}{2})$ $g(\mathbf{x}_{II}) = \sum_{x_i\in\mathbf{x}_{II}}(x_i - G(t))^2$ $G(t) = \|sin(0.5\pi t)\|, \quad t = \frac{1}{n_t}\lfloor\frac{\tau}{\tau_T}\rfloor$	$x_i \in [0,1] \quad i = 1:n$ $\mathbf{x}_{II} = (x_M, \ldots, x_n)$ $k = 2 : M-1$	I
FDA5	$f_1(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\prod_{j=1}^{M-1}\cos(\frac{y_j\pi}{2})$ $f_k(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\left(\prod_{j=1}^{M-k}\cos(\frac{y_j\pi}{2})\right)\sin(\frac{y_{M-k+1}\pi}{2})$ $f_M(\mathbf{x}) = (1+g(\mathbf{x}_{II}))\sin(\frac{y_1\pi}{2})$ $g(\mathbf{x}_{II}) = G(t) + \sum_{x_i\in\mathbf{x}_{II}}(x_i - G(t))^2$ $y_j = {x_j}^{F(t)}$ for $j = 1,\ldots,(M-1)$ $G(t) = \|sin(0.5\pi t)\|$ $F(t) = 1 + 100\sin^4(0.5\pi t), \quad t = \frac{1}{n_t}\lfloor\frac{\tau}{\tau_T}\rfloor$	$x_i \in [0,1] \quad i = 1:n$ $\mathbf{x}_{II} = (x_M, \ldots, x_n)$ $k = 2 : M-1$	II

Table 6.2: Rest of the five FDA DMOP's. All objectives are to be minimized

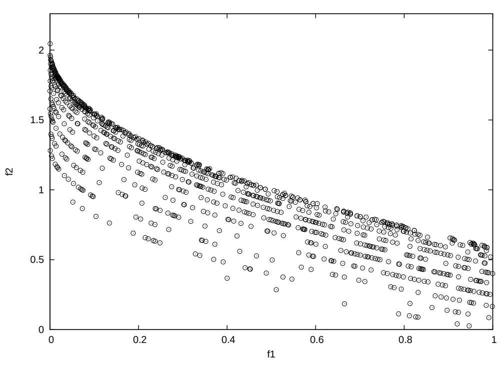
From these problems we have chosen FDA2, FDA3 and FDA5. We have modified FDA3 and FDA5 slightly to convert them into Type-III problems. For FDA3, we have replaced $g(\mathbf{x}_{II})$ with $g(\mathbf{x}_{II}) = 1 + G(t) + \sum_{x_i\in\mathbf{x}_{II}} x_i^2$ and for FDA5, we have replaced $g(\mathbf{x}_{II})$ with $g(\mathbf{x}_{II}) = G(t) + \sum_{x_i\in\mathbf{x}_{II}} x_i^2$. Here τ is the generation counter, τ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. We have used the same parameter values: $n = 2$, $\tau_T = 5$ and $n_t = 10$ as listed in [177].

6.5 Simulation Results: Generational Snapshots

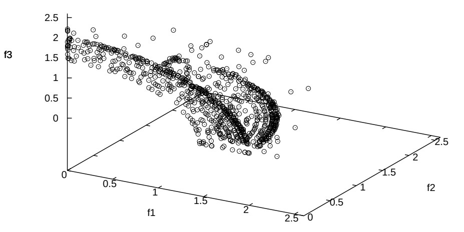
From the generational snapshots, it becomes evident that this new version of PFVO can track the moving optima. Here in each plot the x-axis refers to objective f_1 and y-axis



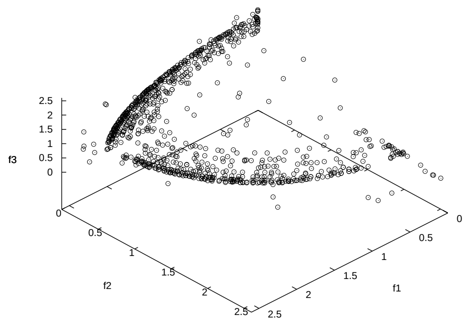
(a) $\mathcal{PF}(t)$ of problem FDA2



(b) $\mathcal{PF}(t)$ of problem FDA3



(c) $\mathcal{PF}(t)$ of problem FDA5



(d) $\mathcal{PF}(t)$ of problem FDA5 with different view point

Figure 6.2: Generational snapshots of different problems

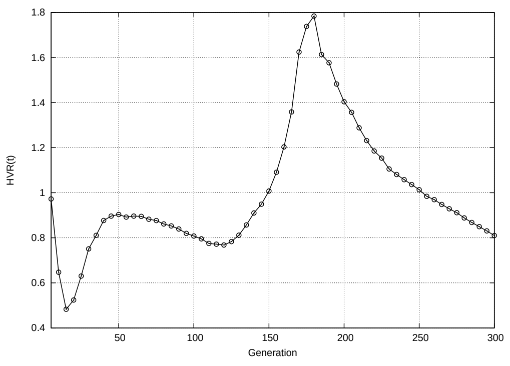
refers to objective f_2. The obtained fronts are the Pareto-front found at different time stamps t. For example, in the case of FDA2, our algorithm finds the same $\mathcal{PF}$, $f_2 = 1 - \sqrt{f_1}$ every time there is a change in 't'. The plot for FDA2 (Figure 6.2(a)) shows 20 different Pareto-fronts in different values of 't'.

6.6 Performance Measure: Generational Distance and Hypervolume Ratio

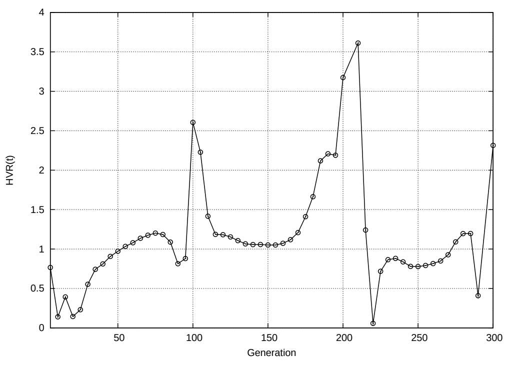
For performance measures, we have used the "Ratio of Hypervolume (HVR)" indicator. The HVR or I_{HR} defines the ratio of Hypervolume (I_H) found from two different algorithms to address the comparative performance. As there are no established performance metrics for dynamic MOEA, we have chosen this indicator from [168]. We have measured the I_{HR} performance of our algorithm at every time step where the front is stationary. I_{HR} at time step t is calculated as followed -

$$I_{HR}(t) = \frac{I_H(\mathcal{PF}_{true}(t))}{I_H(\mathcal{PF}(t))} \tag{6.3}$$

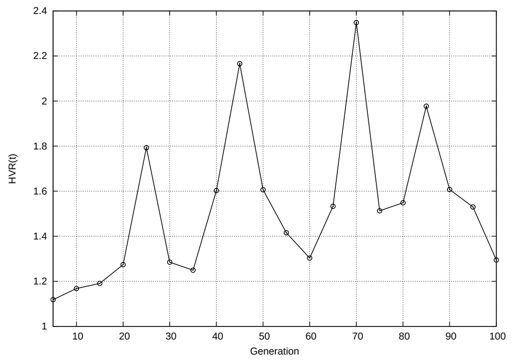
The definition of Hypervolume (I_H) was presented in chapter 4. Here we are considering $\mathcal{PF}_{true}$ as the objective values found by modified NSGA-II-A and $\mathcal{PF}$ as those found by PFVO integrated algorithm. So, at generation t, if $I_{HR}(t) > 1$, then it can be inferred that PFVO integrated algorithm is showing better performance, since a lower value of I_{HR} indicates a better front. We include the I_{HR} results in Figure 6.3 below.



(a) $I_{HR}(t)$ vs. Generation for problem FDA2



(b) $I_{HR}(t)$ vs. Generation for problem FDA3



(c) $I_{HR}(t)$ vs. Generation for problem FDA5

Figure 6.3: Ratio of hypervolume indicator results for problem FDA2, FDA3 and FDA5.

6.6.1 Analysis

From the results provided in Figure 6.2, we can see that PFVO is actually able to track the moving optima, where each of the lines represent different Pareto-front obtained by the algorithm at different time stamps. From Figure 6.3, we can also infer that the approximation accuracy and convergence speed gain by PFVO is promising. For problem FDA2 (Figure 6.3(a)), we can see that our approach performs better between generation 150 and 250. For FDA3(Type III), our algorithm performs better most of the time (Figure 6.3(b)). Significantly, our approach is always better than modified NSGA-II-A (Figure 6.3(c)) for FDA5(Type III).

6.7 Conclusion

In this chapter we have discussed the idea of implementing the PFVO in different way. The purpose of this extension of the PFVO was not aimed to design a new algorithm, rather we have conducted this experiment to illustrate the notion of "inverse mapping" using dynamic system identification procedures. Moreover, this new implementation of PFVO is not confined to only differentiable objective functions and can be extended to many objective problems as well.

Chapter 7

Integration of PFVO into Parallel MOEA

7.1 Introduction

In this chapter, we present a parallel multiobjective evolutionary algorithm enhanced with PFVO. In the previous chapters, it was shown that PFVO enhanced algorithms converge to the true Pareto-front within a smaller number of evaluations. However, to further strengthen our claim, we performed additional experiments using a distributed architecture. From the past two decades, as the popularity of the development of different MOEA's are significantly gaining importance, the desire to reduce their execution time and resource expenditure naturally leads to the consideration of parallel and distributed processing in MOEA's [8].

Developing efficient MOEA's that can be deployed in distributed environment is highly advantageous in solving complex MOP's [179] [180] [181]. For this reason, the idea of providing support for distributed execution of MOEA's has been investigated by many researchers [10] [182]. In particular, this topic has been thoroughly investigated for MOEA's and their different parallel execution models.

However, there are also some examples for surrogate enhanced parallel models [183] [114]. In this chapter we explore PFVO's capability of finding better solutions in a given number of function evaluations in a parallel environment.

7.2 Grid Oriented Technologies for Parallel EA Models

To test the efficacy of the PFVO assisted parallel model, we have used Grid enabled technologies for our implementation. Grid is a type of parallel and distributed system that enables the sharing, exchange, selection and aggregation of geographically distributed "autonomous" resources. Due to the space constraints, we are not going to elaborate the concepts behind Grid computing paradigm. For interested readers, please refer to [184] for more details on Grid computing paradigm.

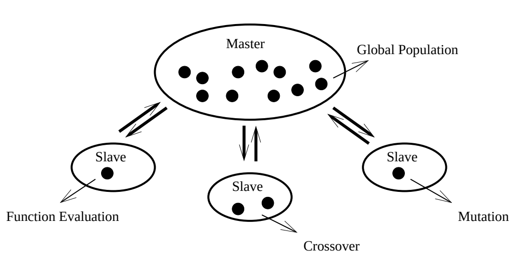
Different Grid based middleware and frameworks are now available for research concerning robust optimization algorithms. For example, Nimrod/O [185] is a tool allowing running distributed optimization problems by using any Nimrod based system, such as Nimrod/G [186], as distribution infrastructure. Nimrod/O allows users to take advantage of different optimization algorithms (BGFS, Simplex, Divide and Conquer, and Simulated Annealing). It requires users to specify the structure of the problem and the variable that needs to be optimized.

ParadisEO-MOEO [187] is an object oriented framework that provides a full featured object model for implementing distributed meta-heuristics, by focusing on code reuse and efficiency. It supports MPI, Condor-G, and Globus as distributing middleware technologies.

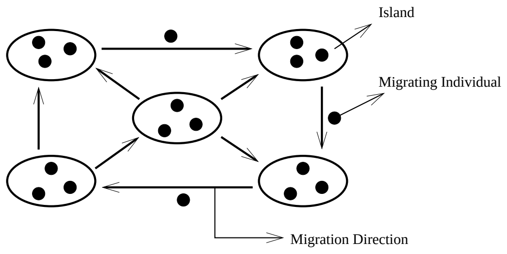
DREAM (Distributed Resource Evolutionary Algorithm Machine) [188] provides a software infrastructure and a technology for the automatic distribution of evolutionary algorithm processing. DREAM is based on a virtual machine that uses a P2P mobile agent system for distributing the computation.

There are also other grid based framework for evolutionary design experiments that drew immense attention of the communities. For example the GOLEM project [189] and the Polyworld project [190], where the research goal was to deploying parallel applications that will harness idle CPU power across the Internet[25] to perform massively distributed evolutionary computation to artificially evolve complex life forms in 3D environment.

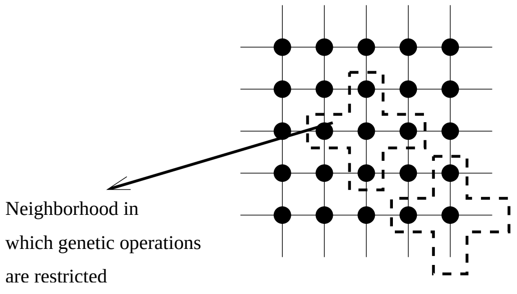
[25]The framework was a variant of peer-to-peer application and in our experiment we have also used similar framework known as ***Offspring***



(a) Single Population Master-Slave Model



(b) Island Model



(c) Diffusion Model

Figure 7.1: Different Parallel Models for MOEA

7.3 Parallel Models for Evolutionary Algorithms

Although parallel models can be implemented in different ways, their implementations have evolved into three basic architectures [8]. Collated from a thorough survey on the existing literatures, our study is summarized as follows -

7.3.1 Master-Slave Models

The *Master-Slave* [191] [192] [193] paradigm is easy to visualize from an algorithmic management perspective. Objective function evaluations are distributed among several slave processers while a master processor executes evolutionary operators and other miscellaneous overhead functions (e.g., computing the current Pareto front, distributing/collecting subpopulations, etc.). This paradigm is fairly simple to implement; its search-space exploration is conceptually identical to that of an MOEA executing on a serial processor. In other words, the number of processors being used is independent of the particular solutions being evaluated, but does affect execution time. This paradigm is illustrated in Figure 7.1(a).

7.3.2 Island Models

The *Island* [9] [194] [195] model is based on the phenomenon of natural populations evolving in relative isolation, such as those that might occur within some ocean island chain with limited migration. This model also called “distributed”, as they are sometimes implemented on distributed memory computers; they are also called multiple-population or multiple-deme. Finally, this paradigm is sometimes termed *coarse-grained* parallelism because each island (processor) contains a large number of individual solutions. The generic island paradigm is illustrated in Figure 7.1(b).

7.3.3 Diffusion Models

Like the master-slave, the *Diffusion* [8] paradigm deals with one conceptual population, however each processor holds only one to a few individuals. This leads some to refer to it as *fine-grained* parallelism. Figure 7.1(c) illustrates the individual distribution on a diffusion model. Genetic operations (crossover/mutaion) occur only within these (possibly overlapping) neighborhoods whose geometry can be a square, rectangle, cube, or other shape depending upon the number of dimensions associated with the model’s topological design. As “good” solutions arise in different areas of the local topology, they then spread or slowly *diffuse* throughout the entire population due to the overlapping or dynamically changing neighborhoods. It should be noted that the host parallel optimizer that we have enhanced with PFVO in this experiment is also a candidate of *Diffusion* model.

7.4 The Host Parallel Optimizer:Complex Network-Based MOEA

The host algorithm that we have used in this experiment is based on *complex networks* [196] [197] [198]. A complex network defines diffusion pattern of the underlying individuals. A network can be modeled as a graph $G(V, E)$ where V is a finite set of vertices and E a finite set of edges (links) such that each edge is associated with a pair of nodes i and j.

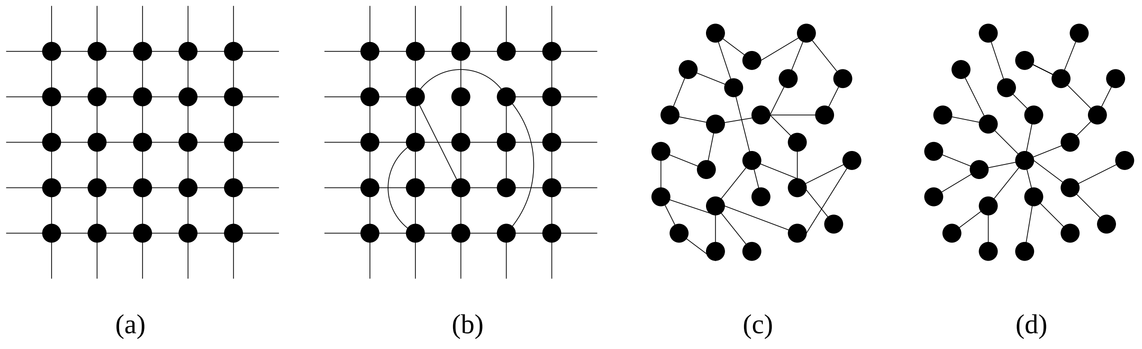


Figure 7.2: Different Diffusion Models applied in this experiment. (a) Regular 2D lattice, (b) Small-world, (c)Random, and (d)Scale-free

The degree k_i of a vertex i defines the total number of edges between vertex i and all other vertices. The "importance" of a vertex is characterized by the total number of edges connected to it. The vertex degree distribution function $P(k)$ defines the probability of randomly selected vertex has exactly k edges.

According to [196], we have also used four different network architectures in this study (see Figure 7.2): (a) the *regular network* defined as a nearest neighbor coupled network (lattice) in which every vertex in the network is joined by a few of its neighbors; (b) the *random network* created by specifying that each pair of vertices is connected by an edge with uniform probability p; (c) the *small-world* network created by randomly re-wiring each edge with some probability $p \ll 1$; and (d) the *scale-free* network characterized by the distinctive connectivity distributions – the probability that a node selected uniformly at random has a certain number of links (degree) follows a power law governed by the relationship $P(k) \sim k^{-\gamma}$. The scale free networks were generated using the preferential attachment model in which we specified the initial number of nodes (b) and the number of edges per node added (e).

7.4.1 The Algorithm

The complex network-based MOEA used in this study is an enhanced cellular evolutionary algorithm [199] [200] [201]. A key component of the model is the communication topology determined by the network architecture.

Here, the individuals are mapped to the nodes of alternative complex networks and interact in their local neighborhood. An important feature of the model is the variation in local neighborhood size between networks – and within particular networks. Typically, the number of neighbors is not constant across the whole network. That is, the size of the local neighborhood is determined by the degree k_i of the current vertex i. This in turn means that the selection pressure will also vary. The exception to this rule is when a 2D regular lattice with Moore neighborhood is used for which each individual has 8 neighbors.

In the selection phase, a relative non-dominance ranking mechanism is used to generate a pool of potential mates from the local neighborhood. A crowding measure is then used to rank individuals in the mating pool. Here, the least crowed individual is viewed as better. This selection regime results in the identification of a "best" mate, j, for the current individual i. After the recombination stage, the resulting offspring are mutated. The parent occupying vertex i and the resultant offspring are then compared using the dominance ranking mechanism. The nondominated individual is then copied into the auxiliary population.

In the event of a tie, one of the children or parent is selected randomly to enter the auxiliary population. all nodes in the network have been processed, the auxiliary population is copied to the main population and the evolutionary cycle continues. An external archive is maintained using the ϵ-dominance mechanism described in [202].

7.4.2 Integration of PFVO into Parallel Model

The integration mechanism is similar to the approach adopted for other sequential host algorithms in chapter 4. After the creation of the child population from the complex network, we apply PFVO on the auxiliary population. Due to this operation, the auxiliary population will contain approximated solutions for the next front. During the archive updating phase, the host algorithm again samples the best individuals from the auxiliary population (found by PFVO) to the complex network for the next generation. An overall schematic is presented in Algorithm 6.

Algorithm 6 Complex Network Based MOEA

```
Require: Randomly generated parent population $P_t$ at generation $t$ with population size $N$ and
    an archive of external population $R_t$
Ensure: After $t_{max}$ number of iteration, population $R_{t_{max}}$ will represent solution of the problem.
 1: Set $t = 0$
 2: Start with initial population, $P_t := \emptyset \quad R_t := \emptyset$
 3: Evaluate $P_t$
 4: Update the archive $R_t$ using $\epsilon$-dominance
 5: Create an auxiliary population $A$
 6: while $t \leq t_{max}$ do
 7:     Distribute the population $P_t$ on the complex network
 8:     for all Individual $i$ (vertex) in network do
 9:         in parallel
10:         Find the best neighbor of $i$, which is $j$
11:         Apply genetic operators on individual $i$, $j$ and create $o^*$
12:         Evaluate $o^*$
13:         Compare the individuals $i$ and $o^*$ w.r.t. non-domination and set the winner to $w$
14:         Add $w$ to the auxiliary population $A$
15:     end for
16:     Set $t := t + 1$
17:     Apply PFVO on the auxiliary population $A$
18:     Update the archive $R_t$ from $A$
19:     Copy $P_t := A$
20: end while
```

7.5 Deployment on *Aneka* Desktop Grid: Utilizing *Offspring*

The deployment of the PFVO enhanced complex network MOEA is done with a recently developed framework known as ***Offspring*** . ***Offspring*** is a plug-in based software environment that allows rapid deployment and execution of evolutionary algorithms on distributed computing environments such as Enterprise Grid. ***Offspring*** provides a more general approach and an extensible platform for creating distributed evolutionary algorithms. With ***Offspring*** researchers can define either the structure of the distributed algorithm or the single computation performed on each of the nodes. These tasks cannot be performed with Nimrod/O that simply provides a technique for partitioning the problem space and distribute the computation. For these reasons, ***Offspring*** is more similar to DREAM since it provides a distribution engine making the development of distributed evolutionary algorithms straightforward. The approach used by DREAM to distribute the computation is based on mobile multi-agent systems, while ***Offspring*** relies on the Enterprise Grid.

Offspring relies on ***Aneka*** [203] to distribute the computation of applications. Initially ***Aneka*** developed as a third generation grid technology in .NET environments. The recent advancement of ***Aneka*** introduced several new Cloud computing [204] capabilities, such as SLA oriented resource allocation and the MapReduce [205] programming model. The main features of the platform are: a configurable service container hosting pluggable services for discovering, scheduling, and balancing workload; a flexible and extensible framework/API supporting a variety of programming models such as threading, batch processing, MPI, MapReduce, and dataflow. These features allow the system administrator to fine tune the installation of the ***Aneka*** by carefully selecting the resources to use on each computational node. From the developer point of view, ***Aneka*** provides a rich programming interface that allows quickly enable applications with support for Cloud computing. Developers can choose between different execution models and select the abstraction that better fit their needs. Although we have also found other implementations of distributed MOEA in P2P environment [206] [192], the main advantage of using ***Aneka*** is its flexibility and robustness that can not be obtainable using Globus [207] or XtremWeb [208] middleware alone.

7.5.1 Implementation Details

Specifically, the original implementation is a hybrid approach that utilizes the notions from *Master-Slave*, *Island* and *Diffusion* models. In our case, the complex network based algorithm is executed on different computational nodes with different network topologies. These computational nodes are controlled by the *Slave* processes. The system also includes similar migration policies as *Island* models (i.e. promising populations are mi-

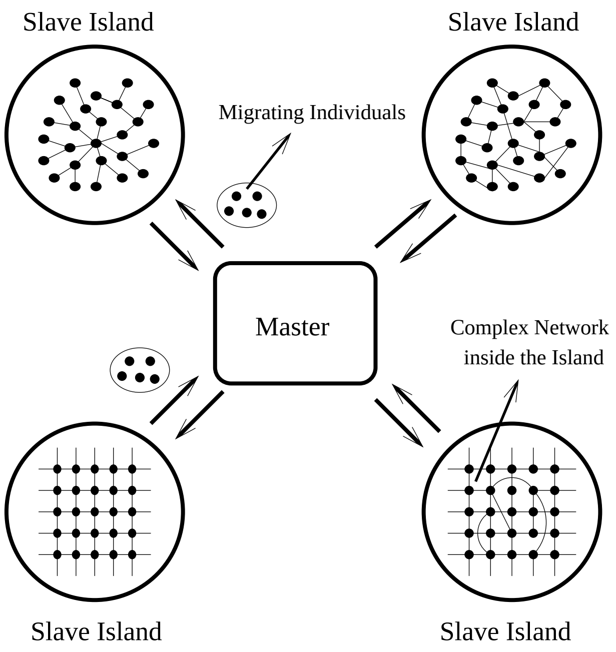


Figure 7.3: The deployment on the grid

grated from one computational node to other). The overall coordination model is controlled by a *Master* node. For this reason, this computational architecture can be regarded as "Hierarchical Model" [209] where a *Master-Slave* model residing on the top of *Island* model and the *Island* model coordinates different *Diffusion* models on different *Slave* nodes. An overview of the model is illustrated in Figure 7.3.

Here, in each island, we have deployed different complex network topologies for the evolutionary search, as stated in [200], the search power of an diffusion based evolutionary algorithm depends on the underlying network structure. In our case, we have used 10 different computing nodes[26] that are remotely connected with and managed by the ***Aneka*** enterprise Grid environment. The ***Offspring*** framework manages the task of the master node that distribute the populations to different working nodes. Each working

[26]Please note that each of the computing nodes contains only one instance of the different complex network topologies (i.e. random, mesh, scale free etc.). Since according to [199,200], the search capability of the diffusion genetic algorithm largely depends on the underlying network topology. Moreover, in the case of MOP, we should consider both convergence and diversity of solutions, for this reason, we have used different network topologies in different computing nodes to facilitate the diversity in searching. Here, each of the computing (slave) nodes was a single Intel Pentium IV (3.4 GHz) Windows XP (SP 2 and .NET 2.0) machine with 1.5 GB of RAM. The Master/***Aneka*** Scheduler node was a Intel Pentium IV (3.0 GHz) Windows XP (SP2 and .NET 2.0) machine with 3.25 GB of RAM.

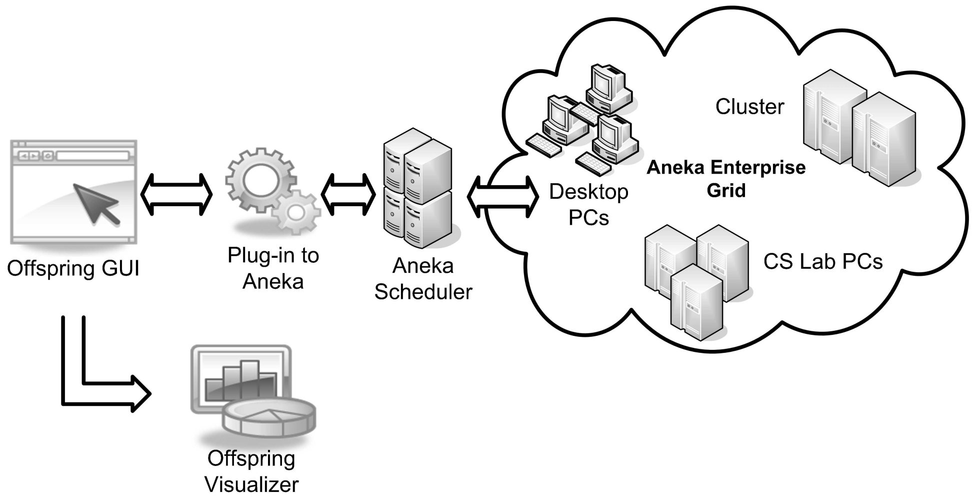


Figure 7.4: Deployment of distributed MOEA on ***Aneka*** enterprize grid using ***Offspring***

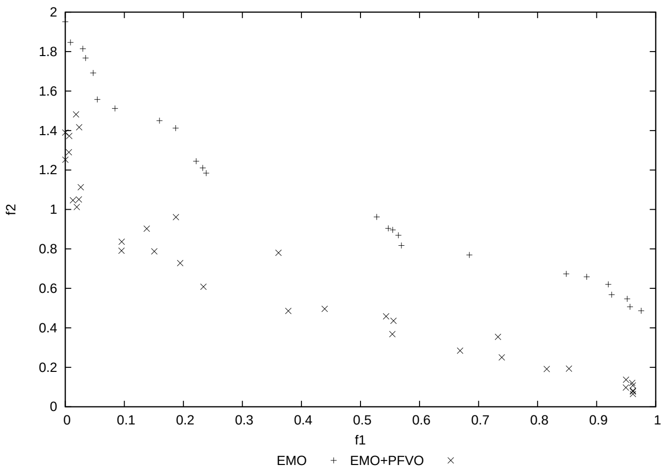
nodes contain equal size of population to be evolved on different network topologies. After each generation, the worker nodes send the best solutions to the master and the master updates the archive from the collected solutions (i.e. archive is maintained by the master node). After the archiving, it again randomly distribute the population to different workers for the next iteration. A high-level view of the deployment of the distributed MOEA using ***Aneka*** and ***Offspring*** is depicted in Figure 7.4.

7.6 Experiments

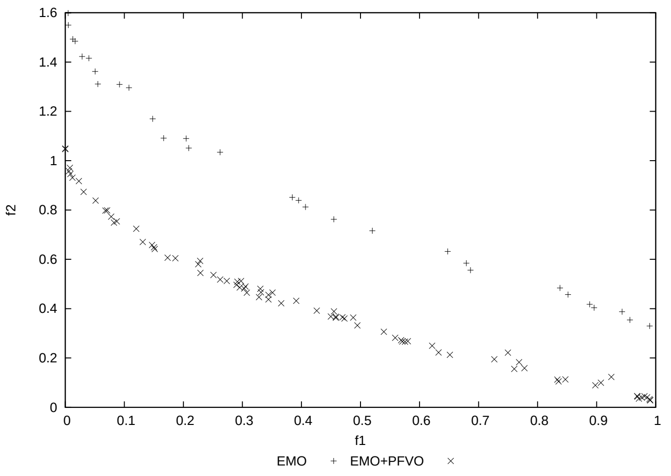
Here our experimentation goal was to analyze how PFVO interacts with parallel model for 2 test problems, ZDT1 and ZDT2. We also provide the generational snapshots and performance measure (I_H and $I_{\epsilon+}$ indicators) at every generation (function evaluations).

7.6.1 Generational Snapshots

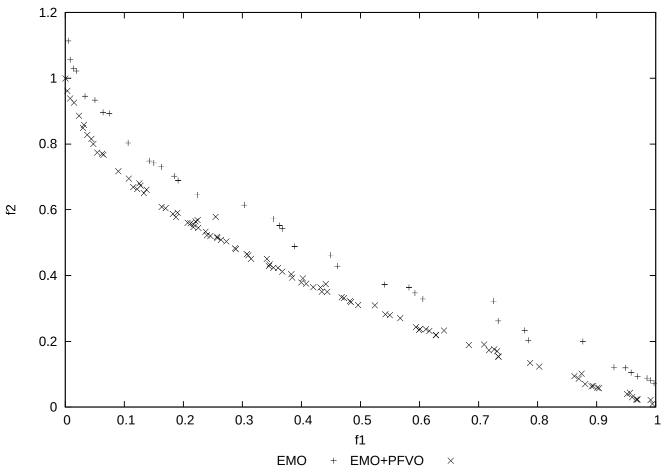
In this section we provide the generational snapshots of the two algorithms, the Parallel diffusion model and its PFVO enhanced version. The parallel model is denoted by the name “EMO” and PFVO enhanced version is denoted by “EMO+PFVO”. The presentation of the snapshots are same as described in chapter 5.



(a) Parallel Model and "Parallel Model + PFVO"on ZDT1 (at Function Evaluation 4600)

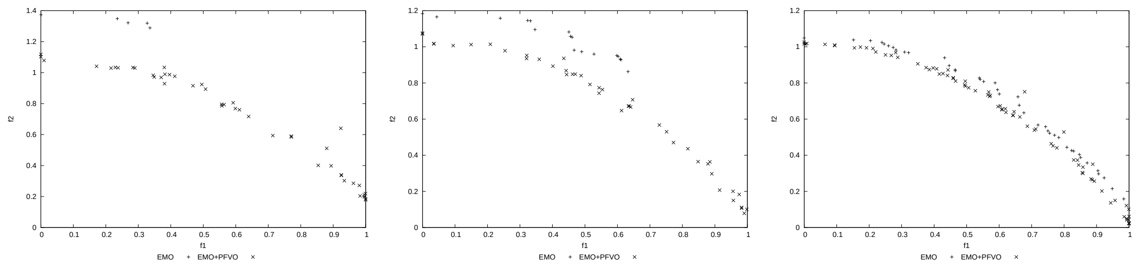


(b) Parallel Model and "Parallel Model + PFVO"on ZDT1 (at Function Evaluation 7400)



(c) Parallel Model and "Parallel Model + PFVO"on ZDT1 (at Function Evaluation 13000)

Figure 7.5: Generational snapshots of both algorithms on ZDT1 problem

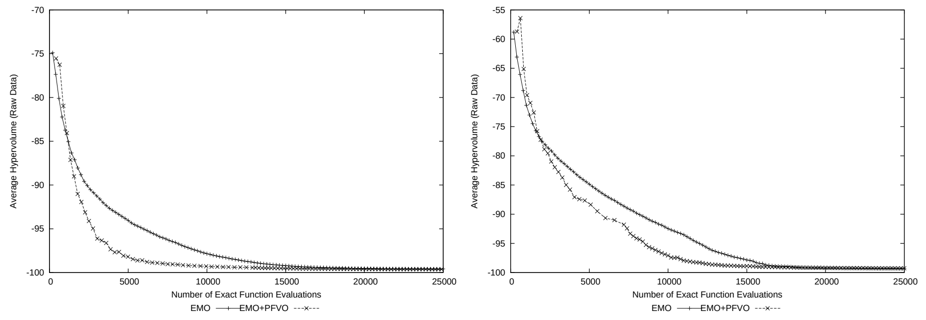


(a) Parallel Model and "Parallel Model + PFVO"on ZDT2 (at FE 7000, 9400 and 12400)

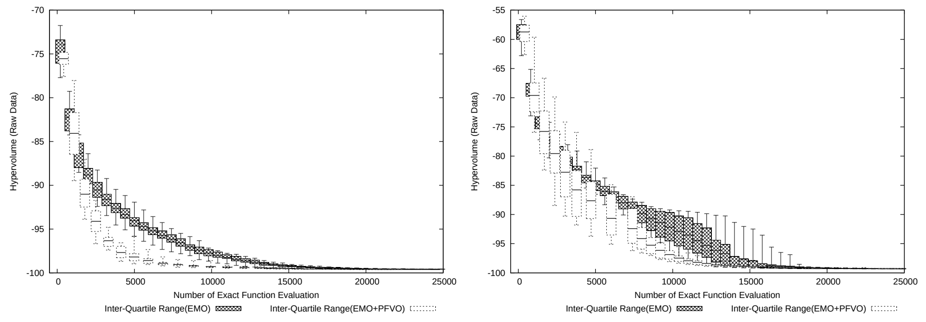
Figure 7.6: Generational snapshots of both algorithms on ZDT2 problem

7.6.2 Performance Results

For performance measure, we have also used Hypervolume (I_H) and Epsilon ($I_{\epsilon+}$) indicator values with respect to exact number of function evaluations. We have also provided both average and box plot of the performance results.

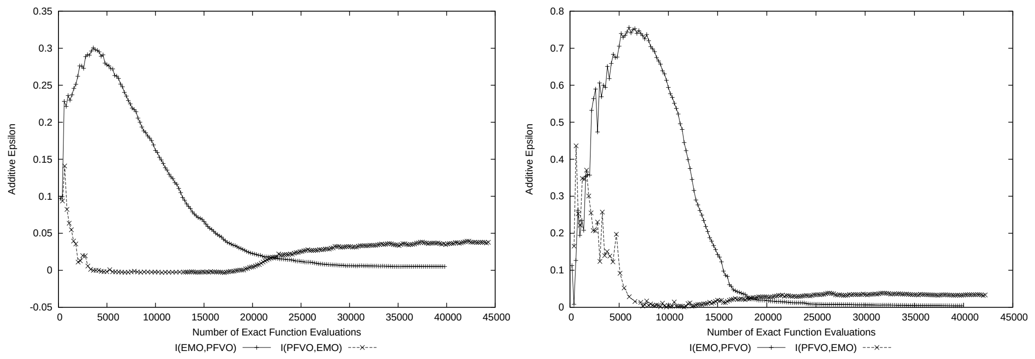


(a) I_H(Parallel Model) and I_H(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)

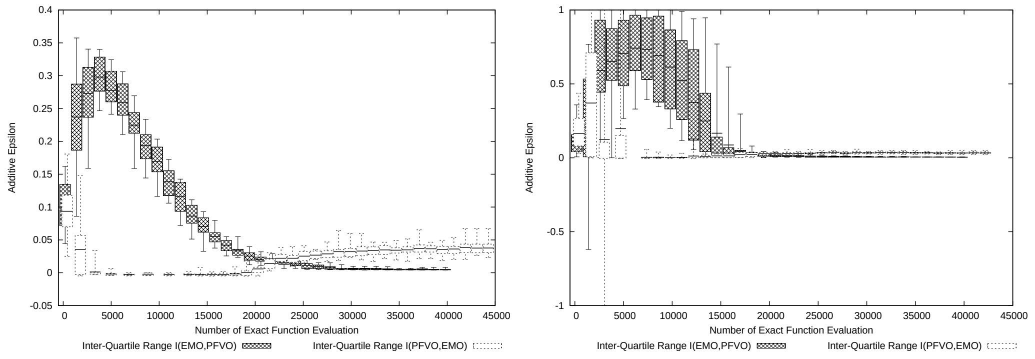


(b) I_H(Parallel Model) and I_H(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)

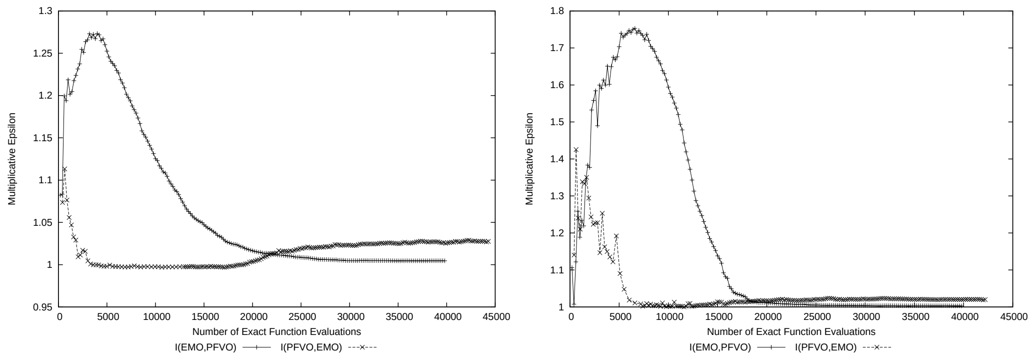
Figure 7.7: I_H vs Function Evaluations of both algorithms



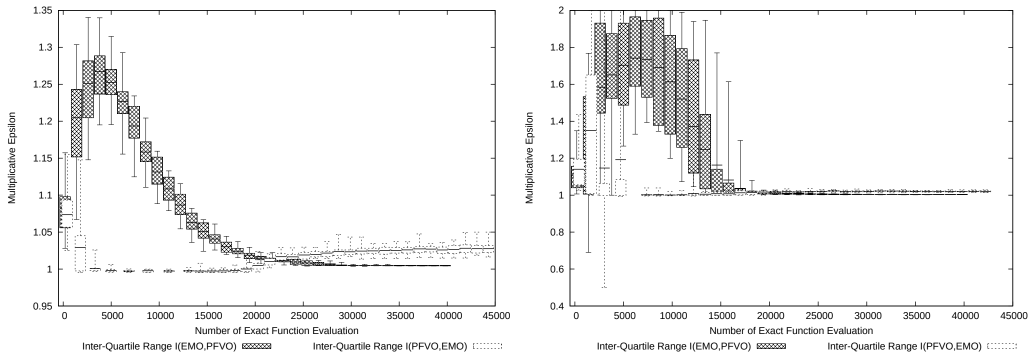
(a) $I_{\epsilon+}$(Parallel Model) and $I_{\epsilon+}$(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)



(b) $I_{\epsilon+}$(Parallel Model) and $I_{\epsilon+}$(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)



(c) $I_{\epsilon+}$(Parallel Model) and $I_{\epsilon+}$(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)



(d) $I_{\epsilon+}$(Parallel Model) and $I_{\epsilon+}$(Parallel Model + PFVO) on ZDT1(Left) and ZDT2(Right)

Figure 7.8: Additive (Figure (a), (b)) and Multiplicative (Figure (c), (d)) $I_{\epsilon+}$ vs Function Evaluations of both algorithms

7.6.3 Analysis

From these plots, we can see that PFVO enhanced parallel model has a better convergence gain with respect to the original parallel algorithm, especially for ZDT1 problem. However, for ZDT2, this gain is not as good as ZDT1. For I_ϵ indicator, an interesting aspect should be noted that, up to function evaluation 22500 (approximately), the PFVO enhanced model generates better solution than the original one. However, its performance shows to be degraded after 22500 function evaluations. In some experiments (in chapter 5), we can see similar results. The possible reason behind this phenomenon is that the total number of distinct fronts is reduced as the solutions become close to the true Pareto-front. Since PFVO always try to generate dominated solutions from the available solutions on the distinct non-dominated fronts, when the number of non-dominated fronts are reduced (at the proximity of the true Pareto-front), the approximation capability of PFVO is degraded. So, our suggestion is to use the PFVO during the initial generations of the host optimizer, as the solutions reach to the proximity of the true Pareto-front, the PFVO operation can be stopped. The rest of the search process can be carried out by the host optimizer itself.

7.7 Conclusion

In this chapter, we have shown that PFVO can be equally useful in Parallel model as well as in serial algorithms. However, some important considerations should be taken into account when integrating the PFVO into diffusion model. The diffusion model depends on the distribution of individuals on the complex network, and this distribution of individuals is gradually developed during the course of the evolutionary run. Therefore, if this distribution of solutions are suddenly perturbed during a generation, the performance of the diffusion will be affected. In our case, in Algorithm 6, when we generate new approximated individuals using PFVO, they are scattered randomly on the original complex network. If we could replace this procedure with more intelligent replacement, the performance would be much better compared to the current implementation.

Chapter 8

Conclusion

8.1 Introduction

Let us begin by recalling the research problems that we have addressed in this thesis, and our contribution towards solving those problems. While other existing MOEA's, such as NSGA-II, SPEA-II and RM-MEDA, provide a good spread and convergence of solutions, however for complex problems, most general base-line algorithms require a substantial amount of function evaluations to reach the true Pareto-front. To alleviate this problem, there has been a large number of **"Approximation/Surrogate"** models adopted to speed up the normal convergence rate. However, the basic principle of this kind of approach is to model the **"input-output"** relation of the **"objective function"** and to replace it with a cheaper mathematical substitute. In order to support this notion, most Approximation/Surrogate models consider the approximation problem as an instance of **"Multi-dimensional Curve-fitting"** or **"Regression"**.

In this thesis, our aim has been to treat this problem from a different perspective. Our goal was to establish the idea that modeling the input-output behavior of the **"stochastic optimizer"** can be more effective than modeling the **"objective function"** to speed up the convergence rate. We have borrowed techniques from **"Dynamic System Identification (DSI)"** to build the model. More specifically, we have considered the **"stochastic optimizer"** as a **"Dynamic System"** that takes **"design variables"** as **"input"** and generates **"objective values"** as **"output"**. Then from this **input-output** relation, we have modeled the behavior of the stochastic optimizer using **QR-Factorization**.

Previous implementations (a somewhat similar approach, in a sense) were mostly based on **"Inverse Artificial Neural Networks"**. In all MOP's, the number of objective is typically smaller than the number of design variables. For this reason, it is always hard to find a good ANN structure that can exactly mimic the behavior of the objective function. So, in the case of inverse ANN, the number of input nodes needs to be smaller than that of output nodes, which is an infeasible way to model the behavior of a dynamic system. Moreover, to apply inverse ANN, the ANN should be trained during the initial iterations[27] of the stochastic optimizer before starting the approximation procedure, which is also computationally expensive. To replace this idea, we have proposed a novel technique – the **"Pareto-Following Variation Operator"** that does not suffers from the above noted bottlenecks. This model needs smaller amount of learning compared to other existing surrogate models, it can identify the behavior of the optimizer from a single pass, since it utilizes the objective values/design variables from different fronts and tries to find out the possible trajectory that can be inferred from the distribution of the fronts in current generation (i.e. considers the solutions in the current worst front are tend to converge to the current best front). Moreover, as we did not replace the original objective function, our model is free from the limitations of other conventional surrogate models (described in section 3.2 in chapter 3). Even though, PFVO uses the original expensive functions to re-evaluate the individuals, it does not slow down the overall performance of the host optimizer. Since, PFVO is guaranteed to generate non-dominated solutions from the current dominated ones for the next iteration[28], this helps the host optimizer to skip redundant function evaluations on a large number of non-promising solutions.

We have integrated PFVO into several popular implementations of MOEA's found in the literature. Moreover, we have designed our algorithm in such a way that it can be plugged into any kind of population based stochastic optimizer, that uses non-domination based ranking to solve MOP's. As PFVO always generates promising solutions for the next iteration, it can also be considered as a variation operator – that follows the trajectory of the fronts. Hence it is termed – the **"Pareto-Following Variation Operator (PFVO)"**.

We have also conducted an exhaustive simulation experiments using several bench-

[27]In some cases, this training phase may be needed to be applied throughout the entire iterations/generations of the host evolutionary optimizer to enhance the performance of the surrogate.

[28]Please note that, this gain is prominent during the initial phase of the host-stochastic optimizer. See sub-section 8.2.3

mark MOP's and elaborate statistical analysis. The results clearly shows that PFVO can significantly improve the convergence rate of the underlying host MOEA.

To strengthen our hypothesis, we have also implemented PFVO in the **"Fourier Domain"** for dynamic MOP's. Since in the case of dynamic MOP's, the Pareto-front changes with respect to another variable (generally time t). If our claim is true, then the PFVO should help the host MOEA to track the changing Pareto-front more promptly. After applying PFVO on several dynamic MOP's, we have found that it improves the performance of the host MOEA. Moreover, according to the DSI theory, a system can be modeled in both spatial/time and frequency domain. As our claim was to prove that the behavior of a stochastic optimizer can be modeled using DSI technique, this second implementation of PFVO reinforces our hypothesis.

The final contribution of our research was to integrate the PFVO into a diffusion based MOEA in a distributed environment. We have used the ***Aneka*** Enterprize Grid environment for our implementation. We have used the ***Offspring*** plug-in to deploy the parallel model on the Grid. Since diffusion/cellular based MOEA's work in a different way to panmictic EA's, we were initially skeptic about the performance of the PFVO. However, after extensive simulation using different diffusion models, we came to the conclusion that PFVO performs surprisingly well in a parallel environment. Parallel enumerative search is generally applied to find the true Pareto-front[29] of benchmark problems, which is computationally very expensive. Our suggestion is that PFVO could be used to speed up the enumerative search.

In summary, the experimental results presented in this thesis demonstrate that the PFVO can help to solve complex and computationally intensive MOP's in smaller number of function evaluations. For this reason, PFVO can be used as a support tool for general MOEA's to solve computationally expensive real world multiobjective problems, especially, where real time solution and decision making have the highest priority.

[29]As a tool to test the performance of new MOEA's

8.2 Future Directions

In this section we outline some interesting directions for future research that can be undertaken in the context of solving complex MOP's, based on the contributions in this thesis.

8.2.1 Development of A Stand-Alone MOEA

Currently, PFVO is developed as an add-on tool for existing MOEA's. However, it is possible to design a stand-alone MOEA that uses PFVO, instead of the crossover/mutaion operator. To enhance the search capability of the stand-alone MOEA, it is also possible to adopt crowding and niche based selection schemes.

8.2.2 Using Singular Value Decomposition(SVD)

The current implementation of PFVO depends on the QR-Factorization technique to model the behavior of the optimizer. However, QR-Factorization is not always capable of finding pseudo-inverse. For singular matrix, QR-Factorization fails. To alleviate this problem, a more general and stable technique such as **"Singular Value Decomposition (SVD)"** can be used.

8.2.3 Controlling the PFVO

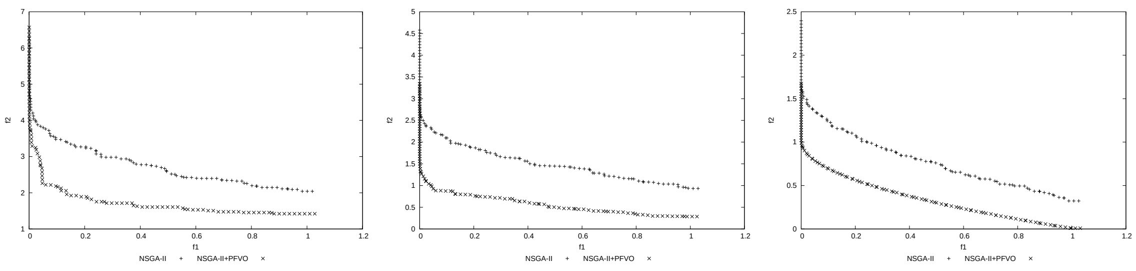
In most cases, PFVO performs better during the initial generations of the MOEA. As generations pass, the total number of distinct fronts are reduced, which leads to unsuccessful approximation. So, our suggestion is to apply PFVO during the initial iterations of MOEA and its invocation should be reduced as the solutions converges to the near Pareto-optimal front. This improvement can be added as an automated proceudre to the future version of PFVO.

Appendix A

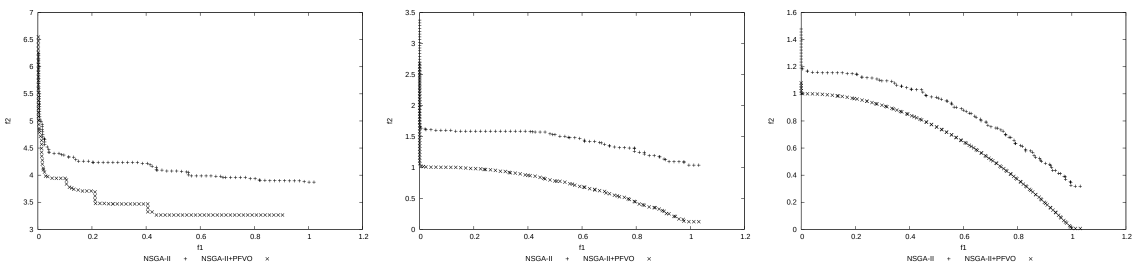
Detailed Results

A.1 Attainment Surface Plot

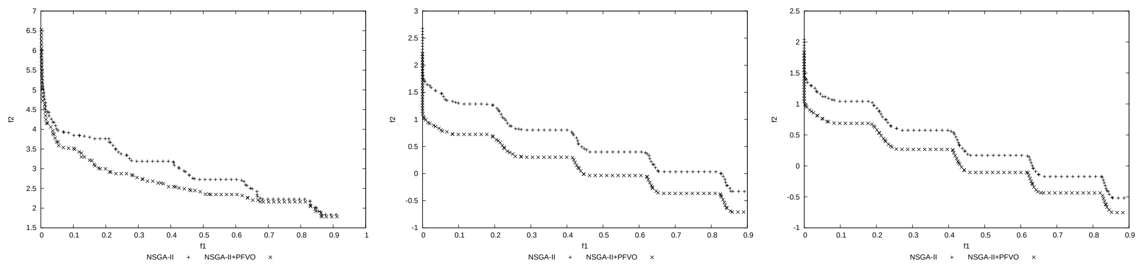
In this section we have included the median attainment surface plot for every algorithm on two test suites described in chapter 4. Due to space constraints, we are unable to provide every generational snapshots, so similar to scatter plot in chapter 4, we are going to present only three generational snapshots for every algorithm (on each problem) -



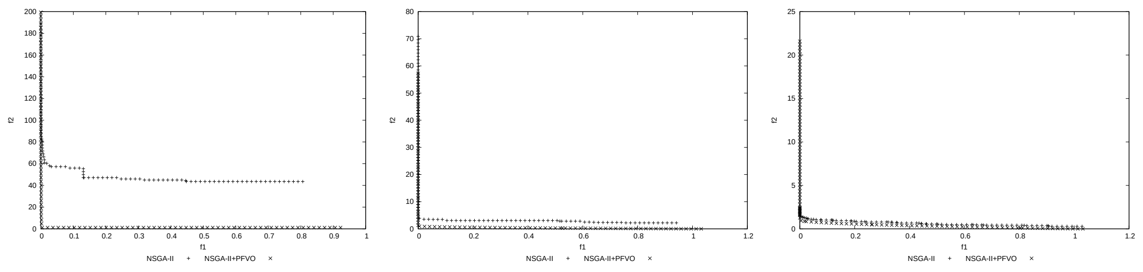
(a) NSGA-II and "NSGA-II + PFVO"on ZDT1 (at FE 800, 2600 and 5200)



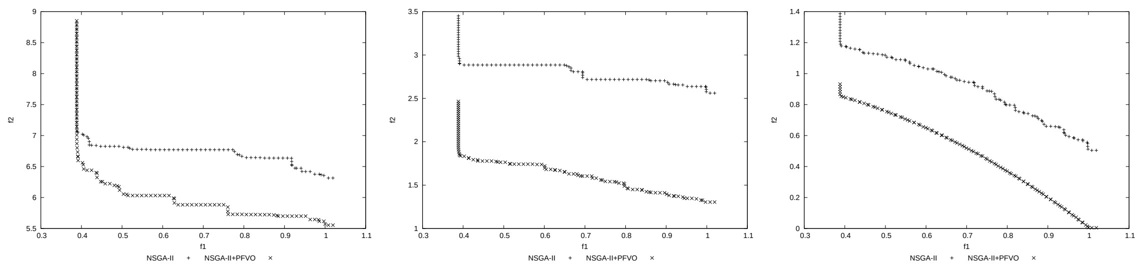
(b) NSGA-II and "NSGA-II + PFVO"on ZDT2 (at FE 400, 5200 and 9600)



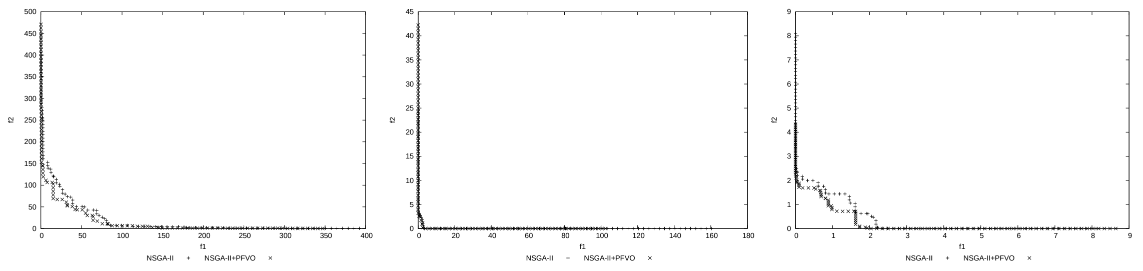
(c) NSGA-II and “NSGA-II + PFVO”on ZDT3 (at FE 600, 4800 and 6600)



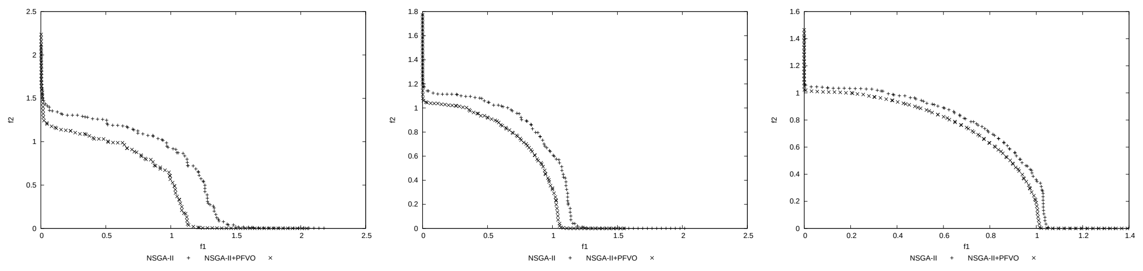
(d) NSGA-II and “NSGA-II + PFVO”on ZDT4 (at FE 800, 22600 and 24800)



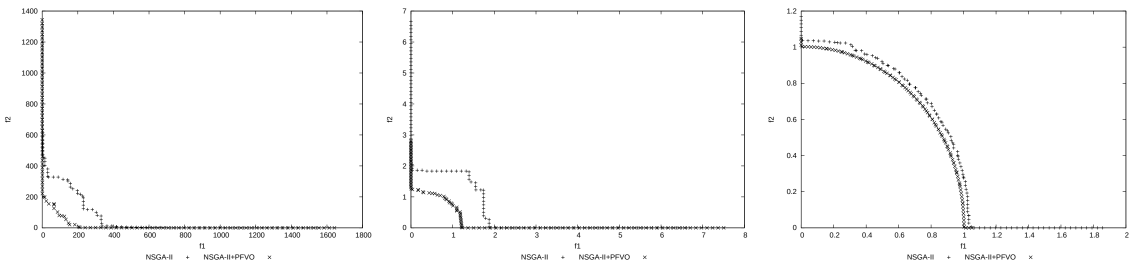
(e) NSGA-II and “NSGA-II + PFVO”on ZDT6 (at FE 1280, 8448 and 20992)



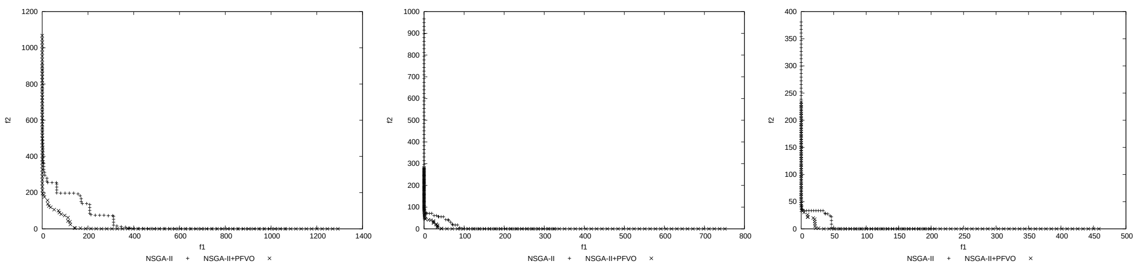
(f) NSGA-II and “NSGA-II + PFVO”on DTLZ1 (at FE 650, 9320 and 11514)



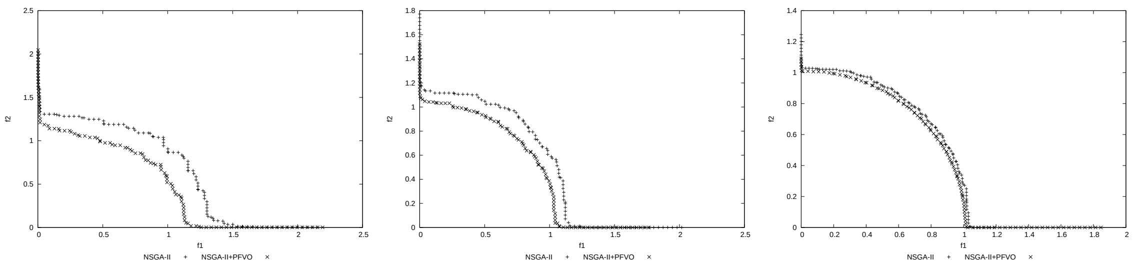
(g) NSGA-II and “NSGA-II + PFVO”on DTLZ2 (at FE 912, 2058 and 3996)



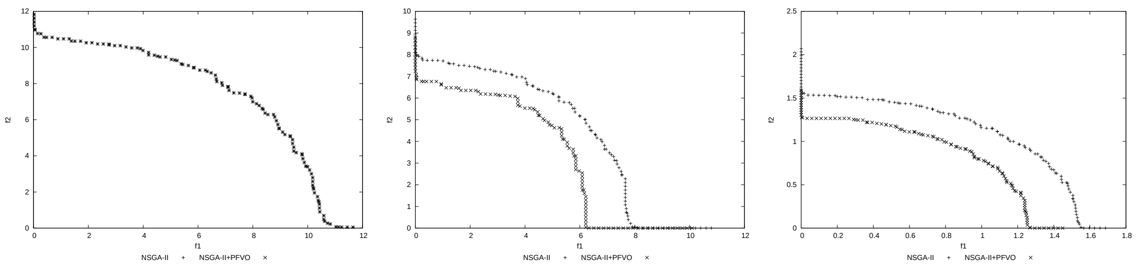
(h) NSGA-II and "NSGA-II + PFVO"on DTLZ3 (at FE 1304, 10188 and 18322)



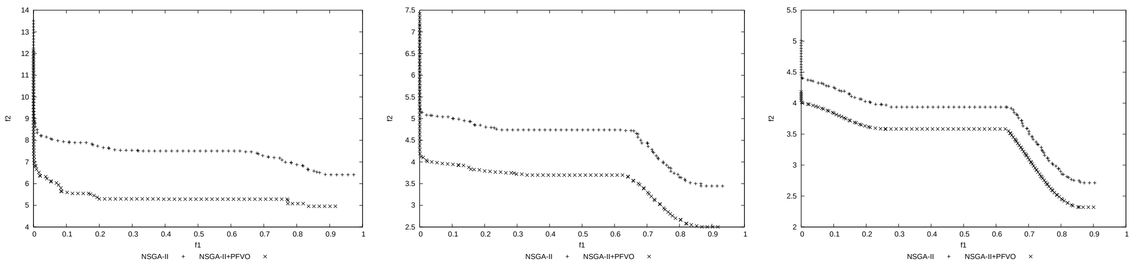
(i) NSGA-II and "NSGA-II + PFVO"on DTLZ4 (at FE 1574, 3114 and 4052)



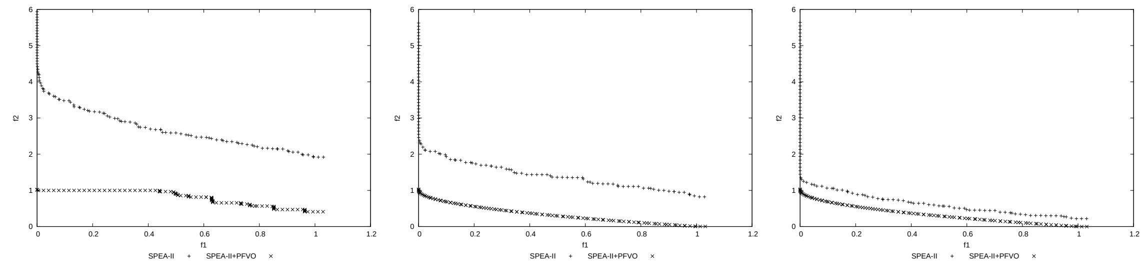
(j) NSGA-II and "NSGA-II + PFVO"on DTLZ5 (at FE 914, 1966 and 4122)



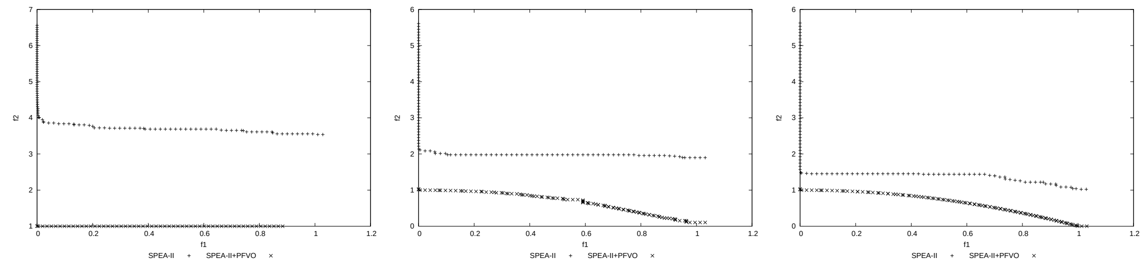
(k) NSGA-II and "NSGA-II + PFVO"on DTLZ6 (at FE 470, 4746 and 42446)



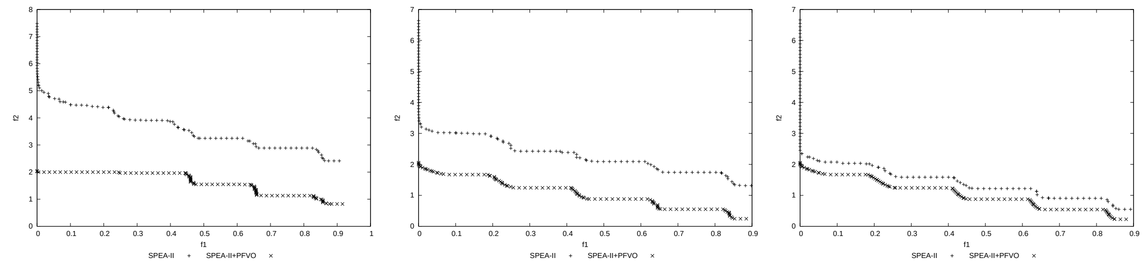
(l) NSGA-II and "NSGA-II + PFVO"on DTLZ7 (at FE 1728, 5534 and 11548)



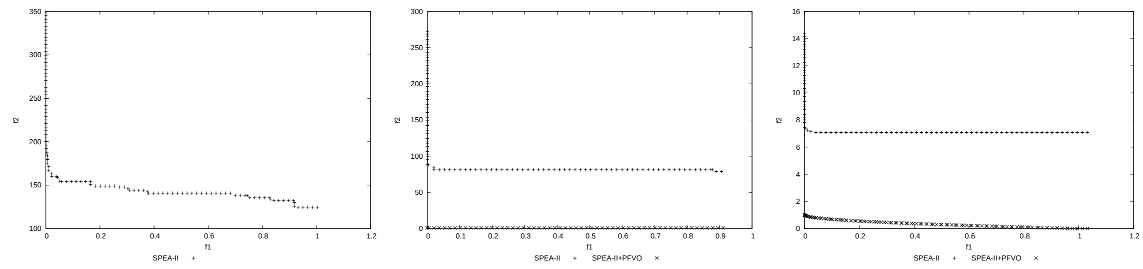
(a) SPEA-II and "SPEA-II + PFVO" on ZDT1 (at FE 544, 2752 and 6668)



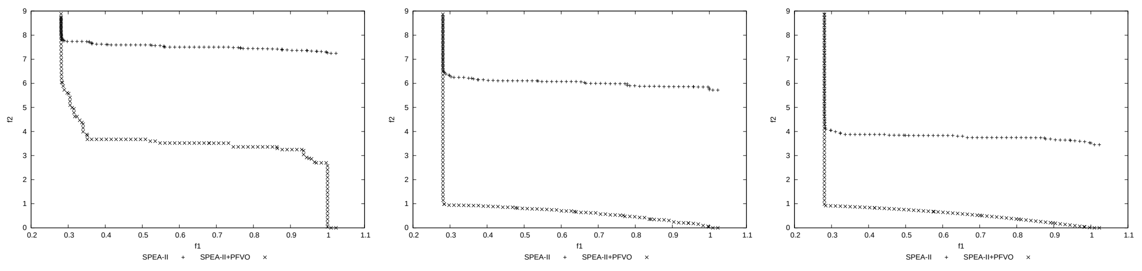
(b) SPEA-II and "SPEA-II + PFVO" on ZDT2 (at FE 534, 3146 and 6344)

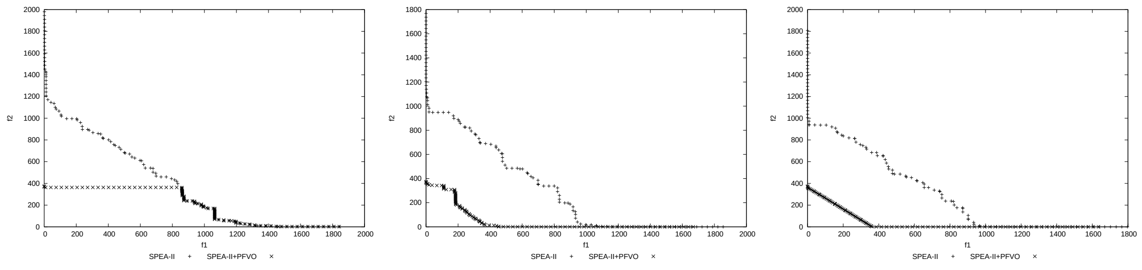


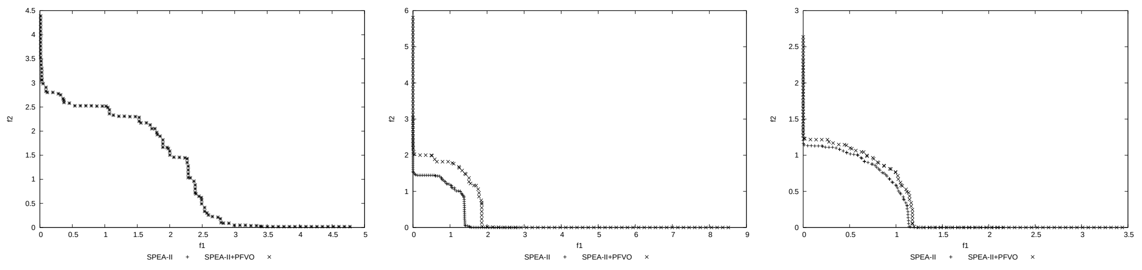
(c) SPEA-II and "SPEA-II + PFVO" on ZDT3 (at FE 532, 2684 and 6560)

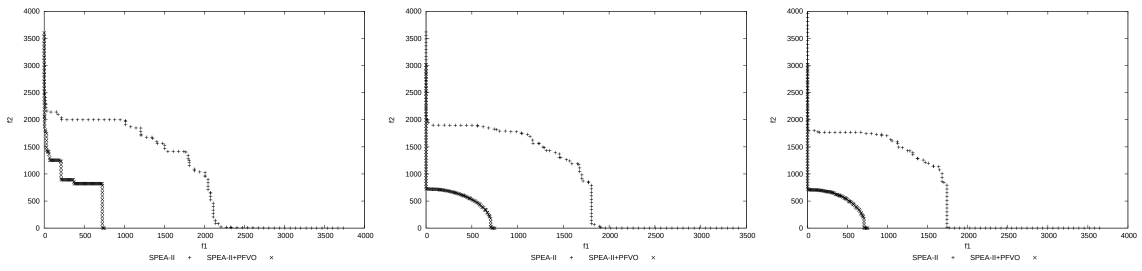


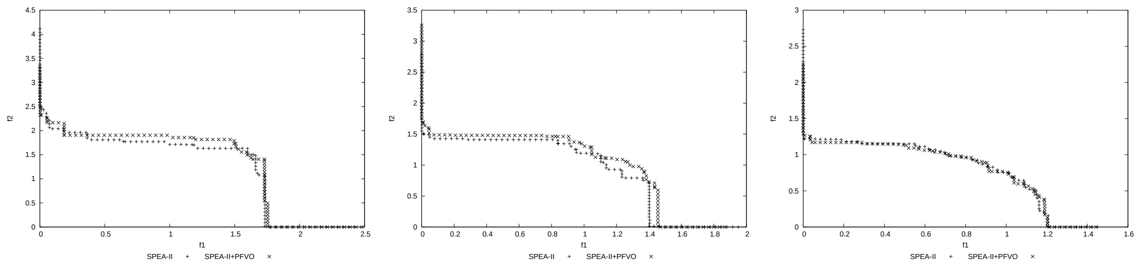
(d) SPEA-II and "SPEA-II + PFVO" on ZDT4 (at FE 636, 1774 and 39208)

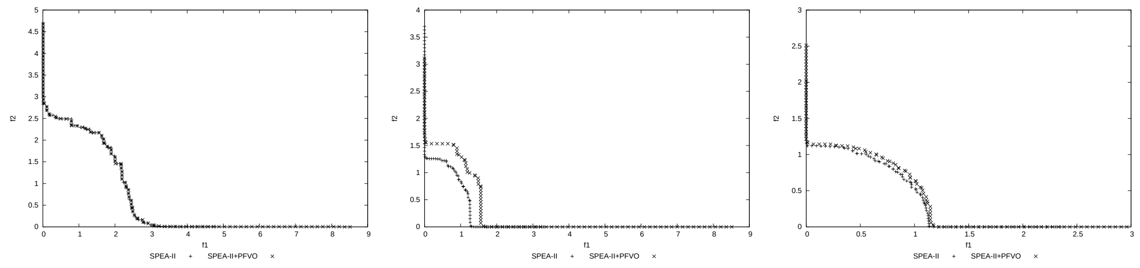

(e) SPEA-II and “SPEA-II + PFVO” on ZDT6 (at FE 910, 2676 and 7438)

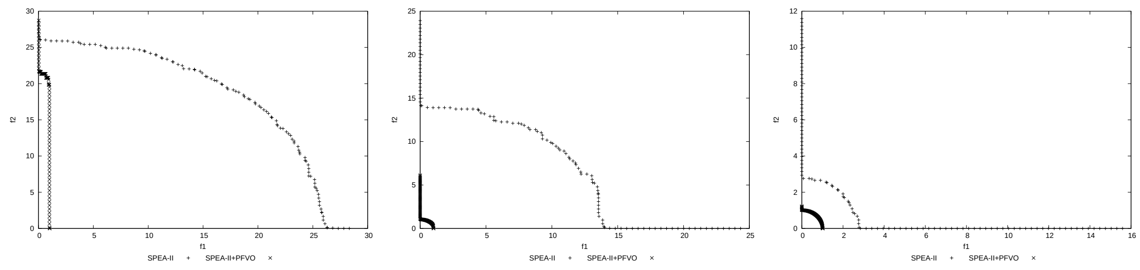

(f) SPEA-II and “SPEA-II + PFVO” on DTLZ1 (at FE 234, 1270 and 3080)

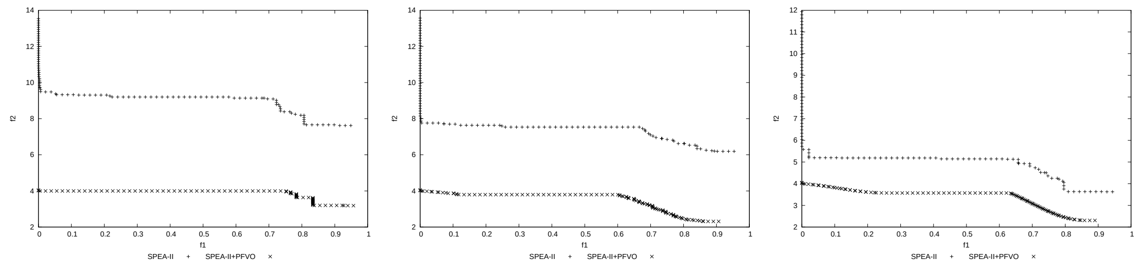

(g) SPEA-II and “SPEA-II + PFVO” on DTLZ2 (at FE 100, 2686 and 13676)

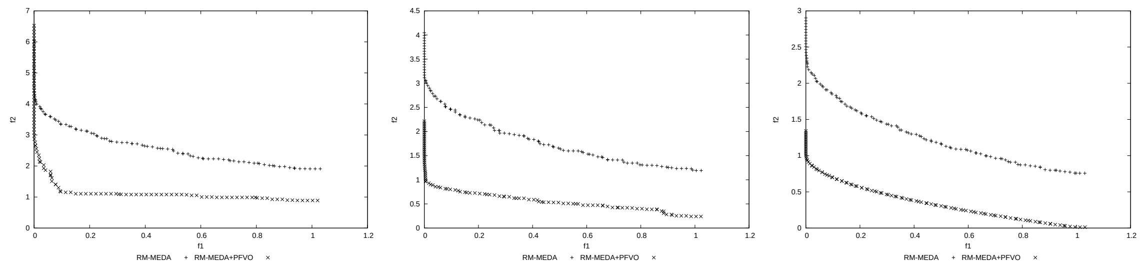

(h) SPEA-II and “SPEA-II + PFVO” on DTLZ3 (at FE 674, 2968 and 6104)

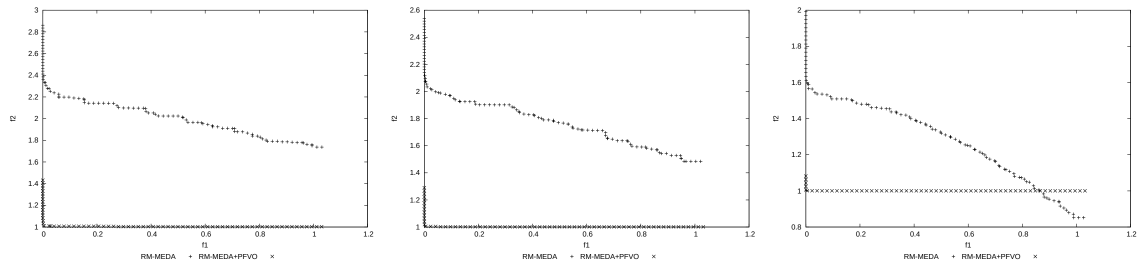

(i) SPEA-II and “SPEA-II + PFVO” on DTLZ4 (at FE 1380, 2832 and 6974)

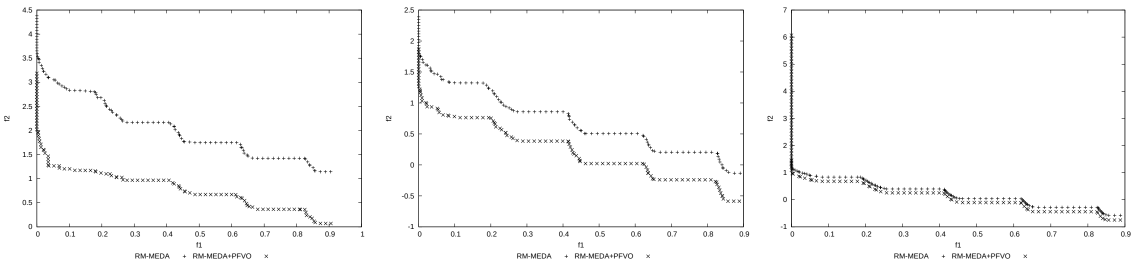

(j) SPEA-II and “SPEA-II + PFVO” on DTLZ5 (at FE 230, 5086 and 17358)


(k) SPEA-II and “SPEA-II + PFVO” on DTLZ6 (at FE 490, 5574 and 44548)

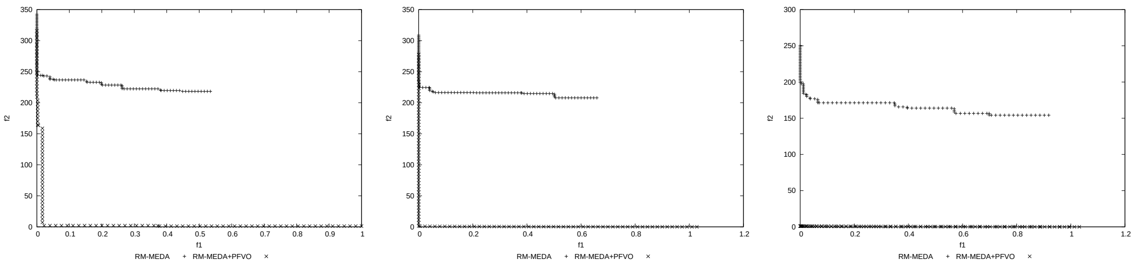

(l) SPEA-II and “SPEA-II + PFVO” on DTLZ7 (at FE 644, 1330 and 4196)


(a) RM-MEDA(+) and “RM-MEDA + PFVO”(×) on ZDT1 (at FE 2088, 6424 and 14836)

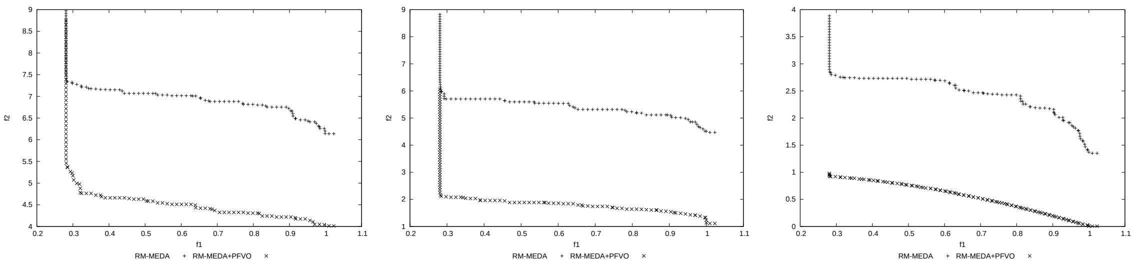

(b) RM-MEDA and “RM-MEDA + PFVO” on ZDT2 (at FE 11612, 14388 and 29496)



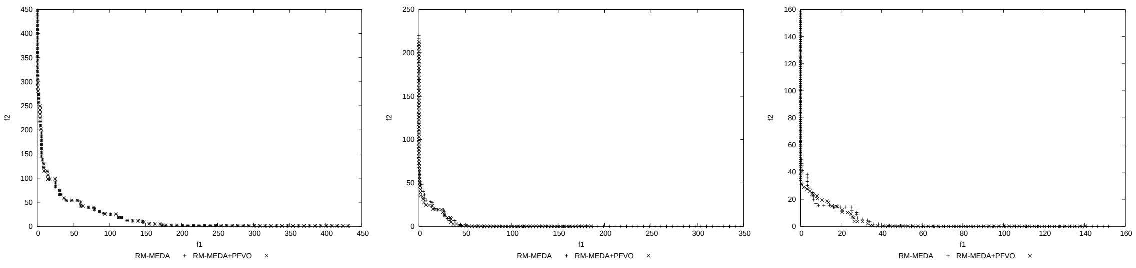
(c) RM-MEDA and “RM-MEDA + PFVO” on ZDT3 (at FE 5908, 30542 and 72144)



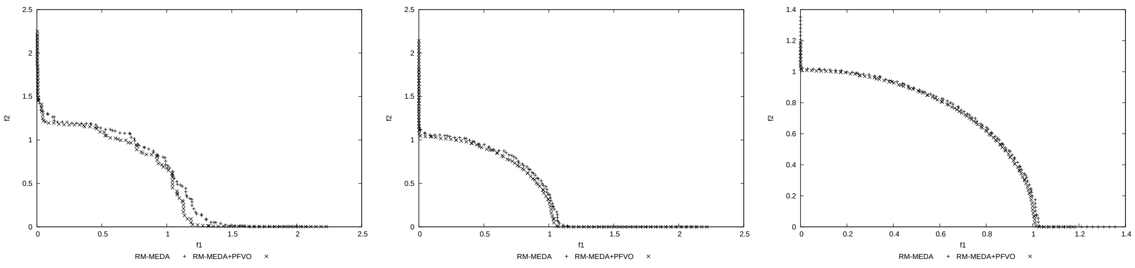
(d) RM-MEDA and “RM-MEDA + PFVO” on ZDT4 (at FE 4784, 6990 and 43668)



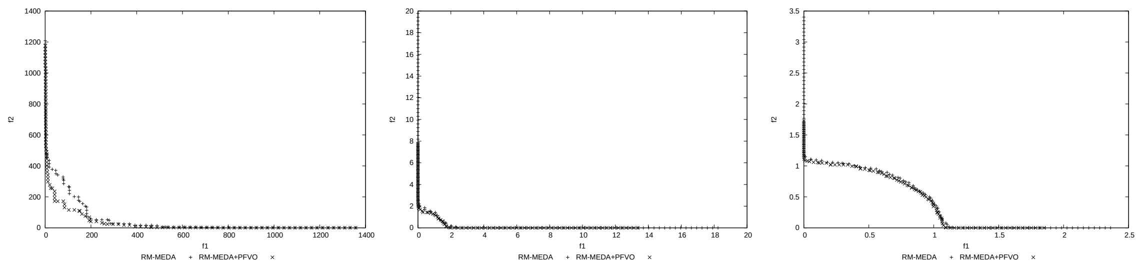
(e) RM-MEDA and “RM-MEDA + PFVO” on ZDT6 (at FE 3320, 11578 and 69758)



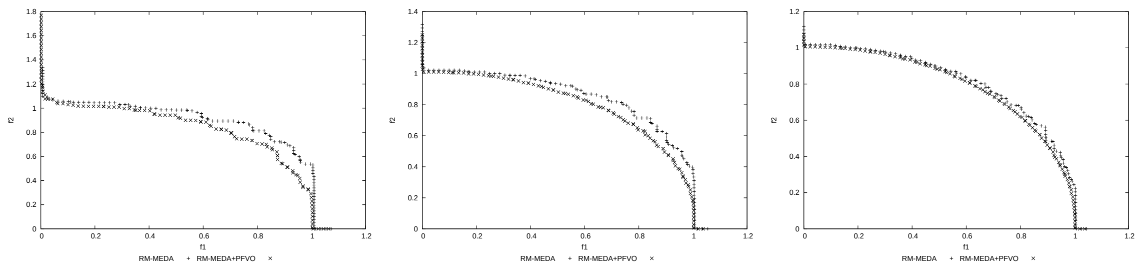
(f) RM-MEDA and “RM-MEDA + PFVO” on DTLZ1 (at FE 200, 20000 and 58600)



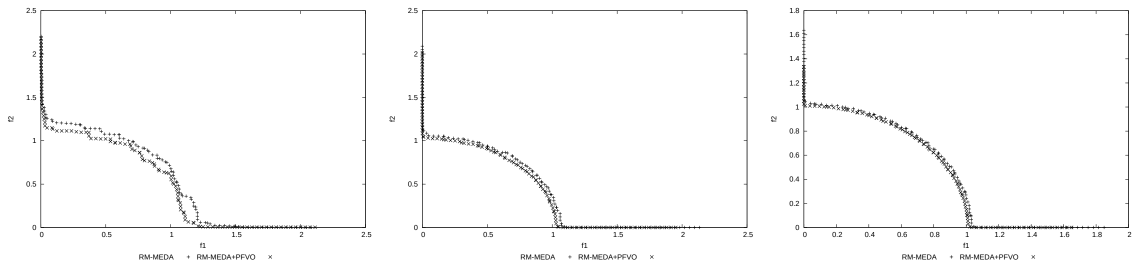
(g) RM-MEDA and “RM-MEDA + PFVO” on DTLZ2 (at FE 468, 3036 and 7342)



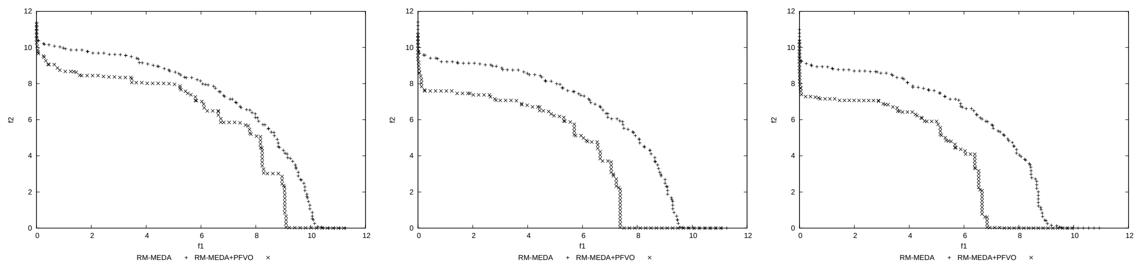
(h) RM-MEDA and “RM-MEDA + PFVO” on DTLZ3 (at FE 644, 11940 and 15938)



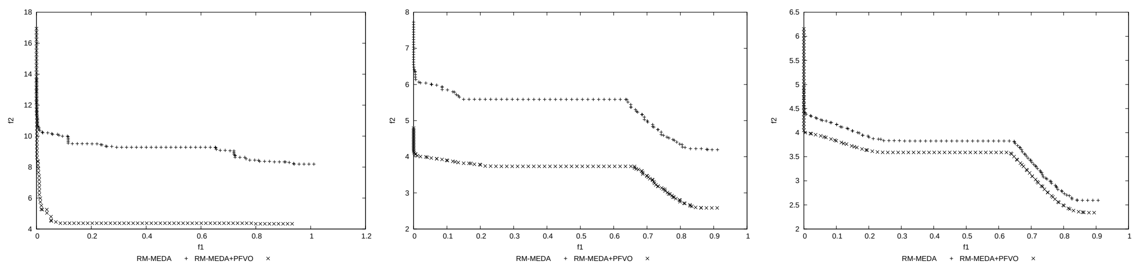
(i) RM-MEDA and “RM-MEDA + PFVO” on DTLZ4 (at FE 7720, 10618 and 14748)



(j) RM-MEDA and “RM-MEDA + PFVO” on DTLZ5 (at FE 710, 3242 and 6154)



(k) RM-MEDA and “RM-MEDA + PFVO” on DTLZ6 (at FE 3182, 9146 and 15554)

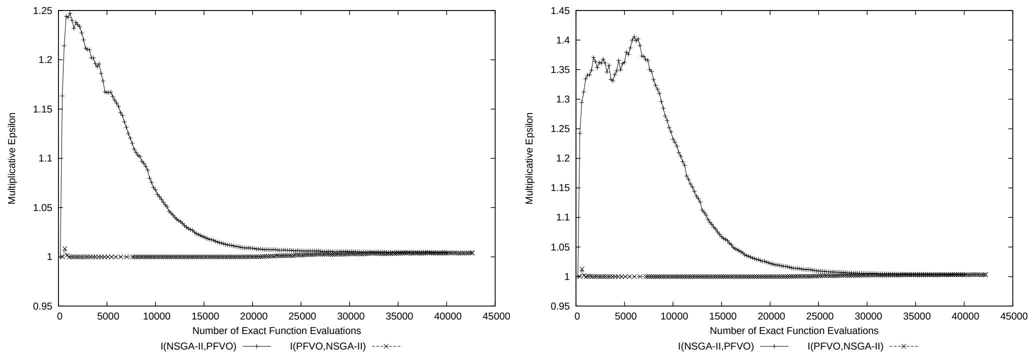


(l) RM-MEDA and “RM-MEDA + PFVO” on DTLZ7 (at FE 1066, 15220 and 51978)

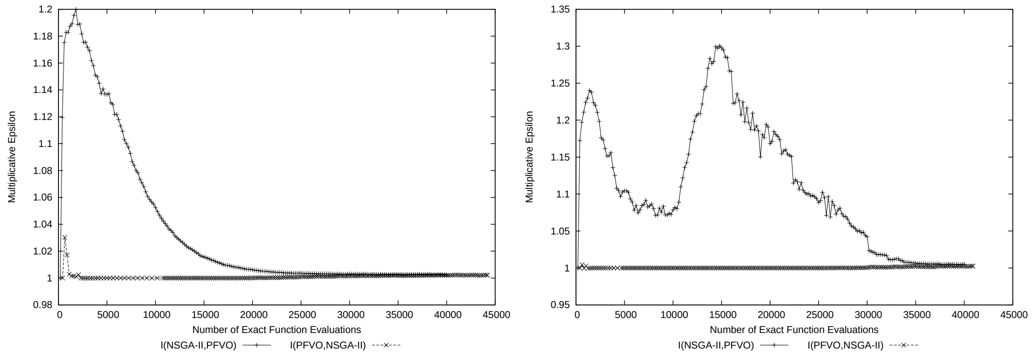
A.2 Performance Results: Multiplicative Epsilon Indicators

Here we provide the multiplicative epsilon indicator results. From these plots we can see that the multiplicative epsilon results are similar to additive version. We have used the source code provided in PISA performance assessment library (`http://www.tik.ee.ethz.ch/sop/pisa/`).

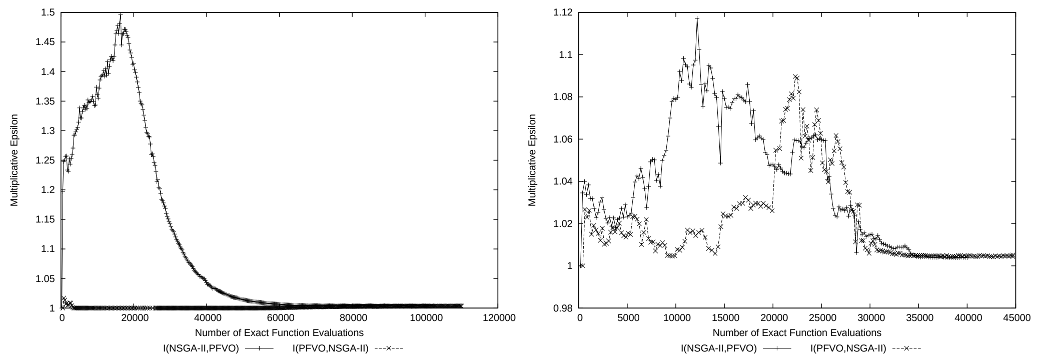
A.2.1 NSGA-II



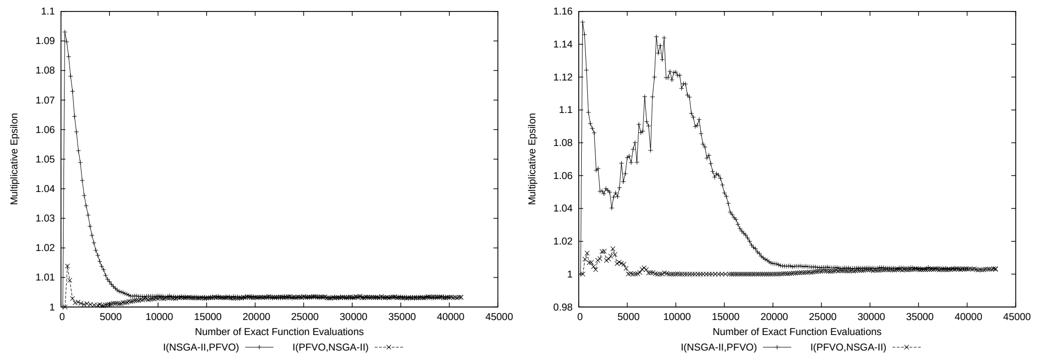
(m) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



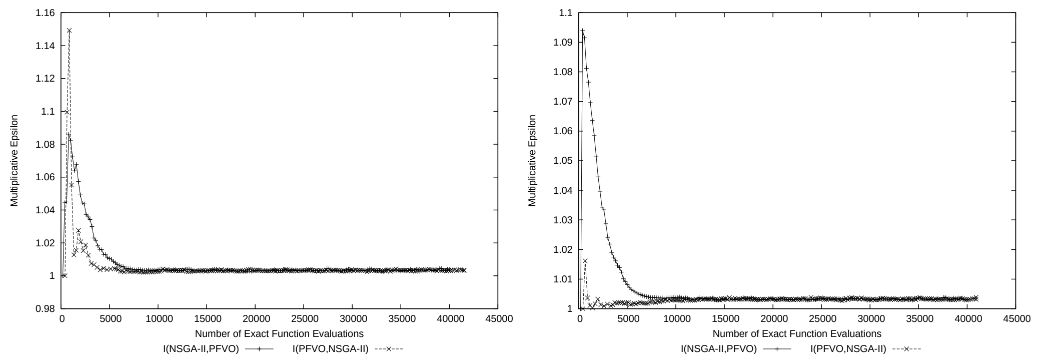
(n) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



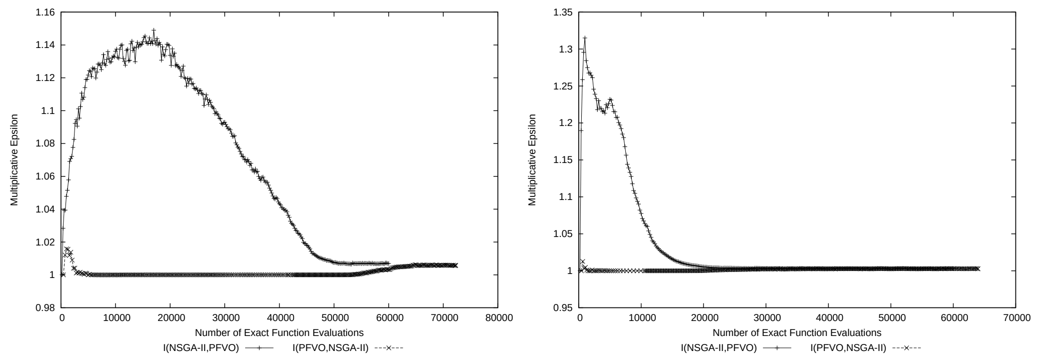
(o) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(p) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on DTLZ2 (Left) and DTLZ3(Right)

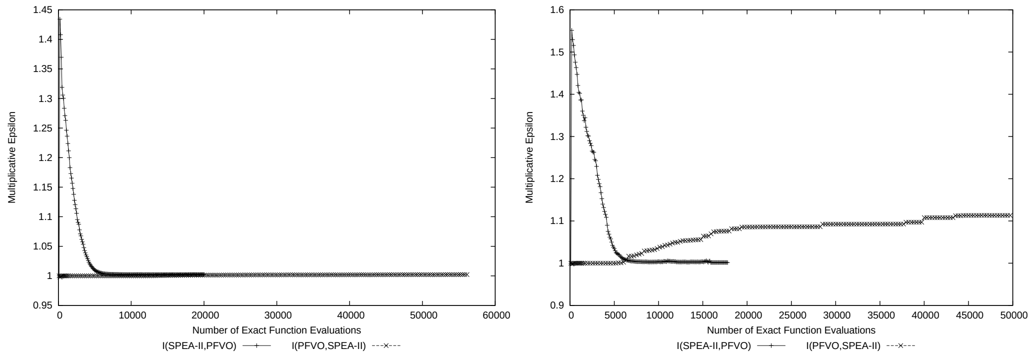


(q) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on DTLZ4 (Left) and DTLZ5(Right)

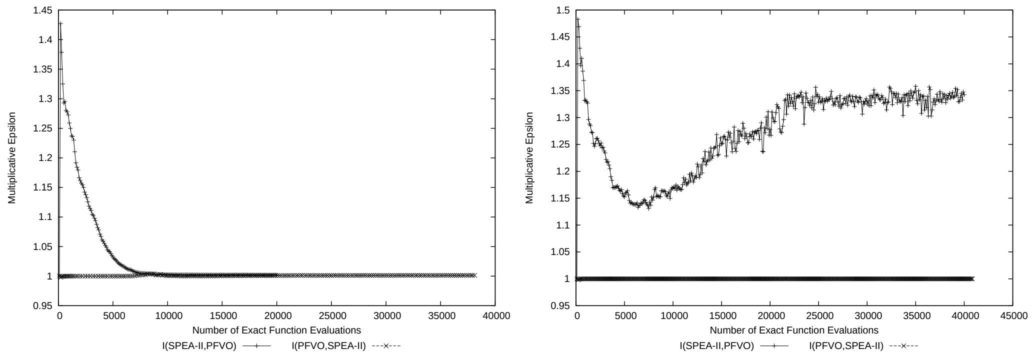


(r) I_ϵ(NSGA-II) and I_ϵ(NSGA-II + PFVO) on DTLZ6 (Left) and DTLZ7(Right)

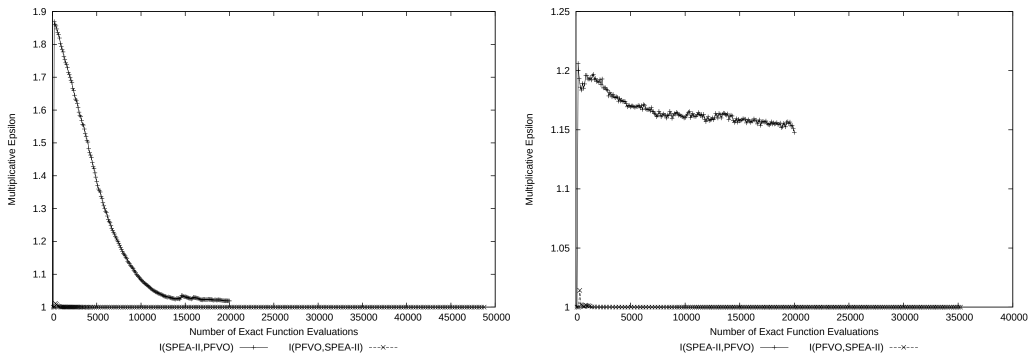
A.2.2 SPEA-II



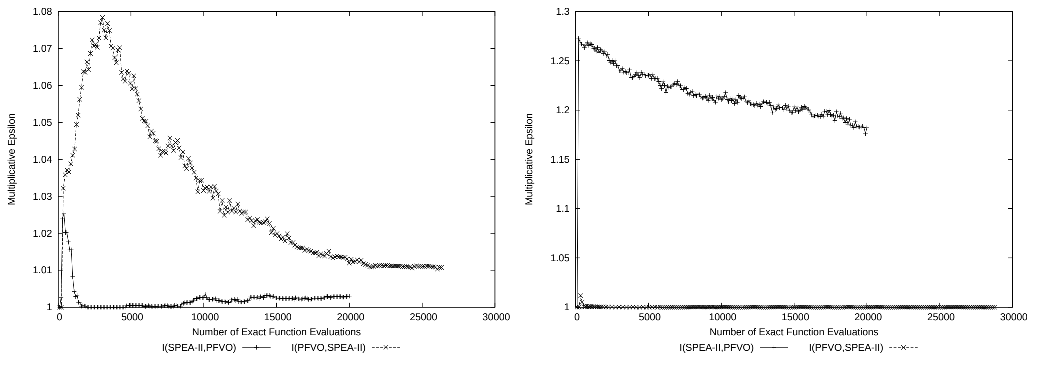
(a) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



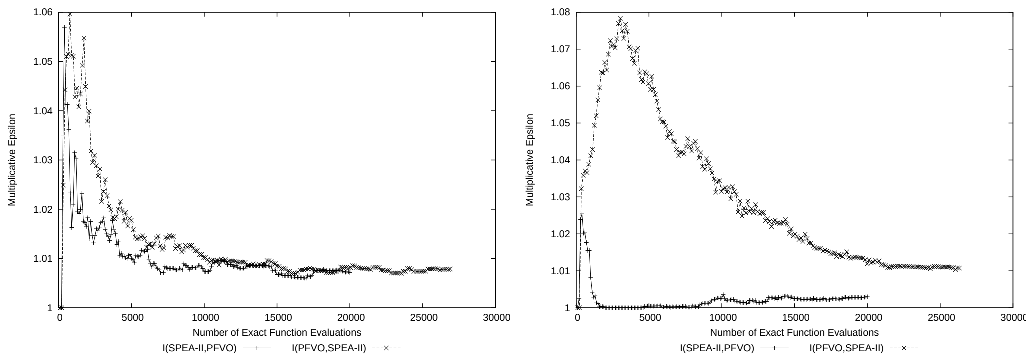
(b) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



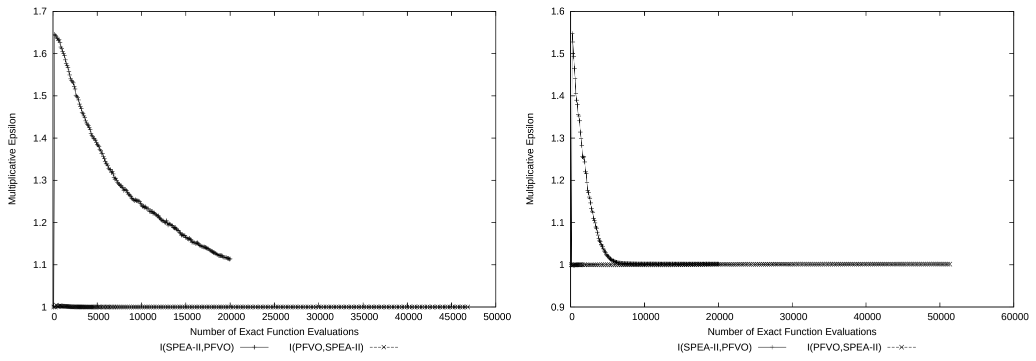
(c) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(d) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)

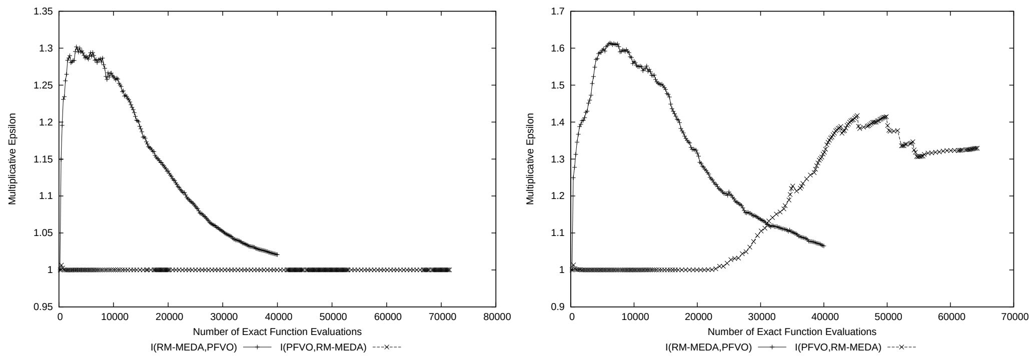


(e) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)

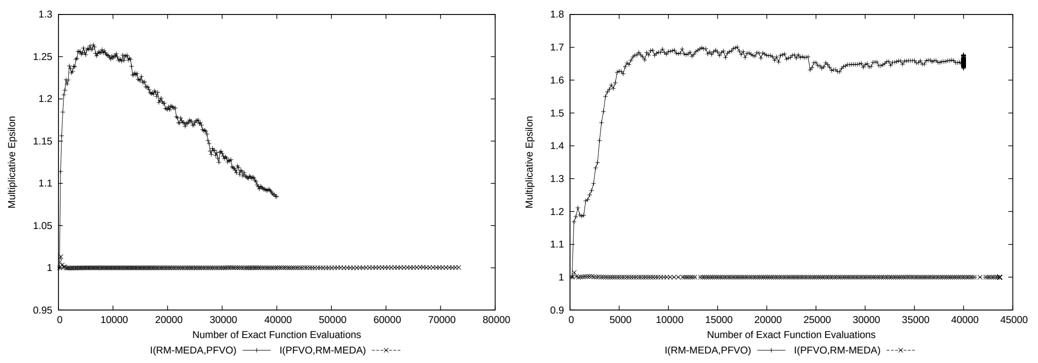


(f) I_ϵ(SPEA-II) and I_ϵ(SPEA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

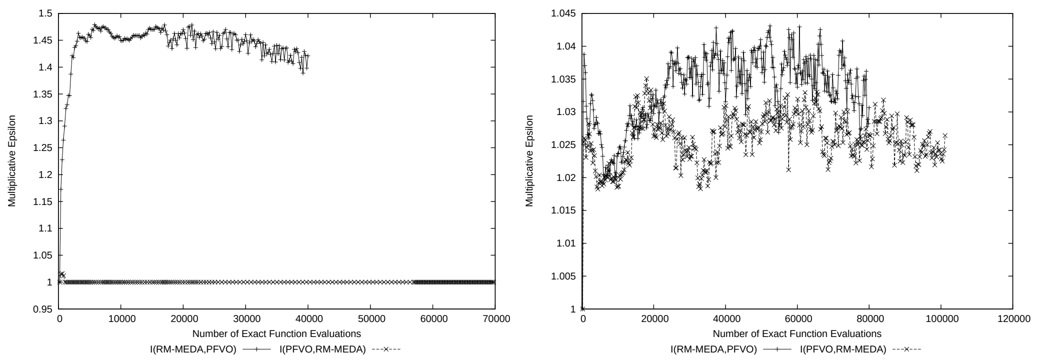
A.2.3 RM-MEDA



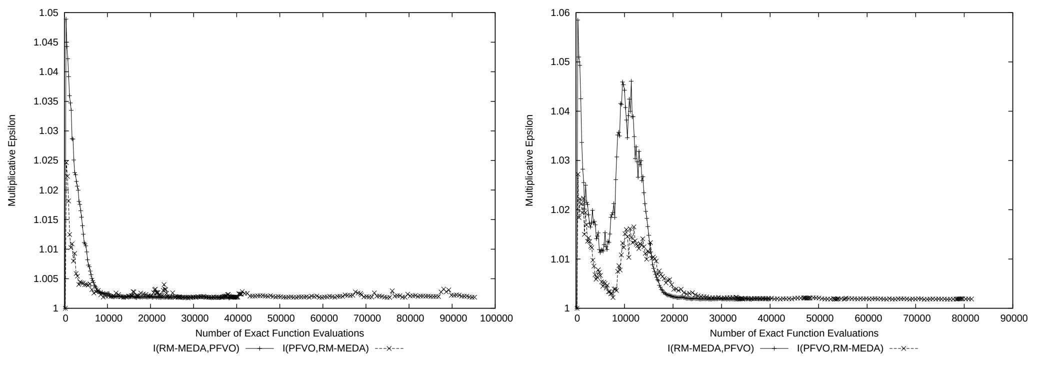
(a) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on ZDT1 (Left) and ZDT2 (Right)



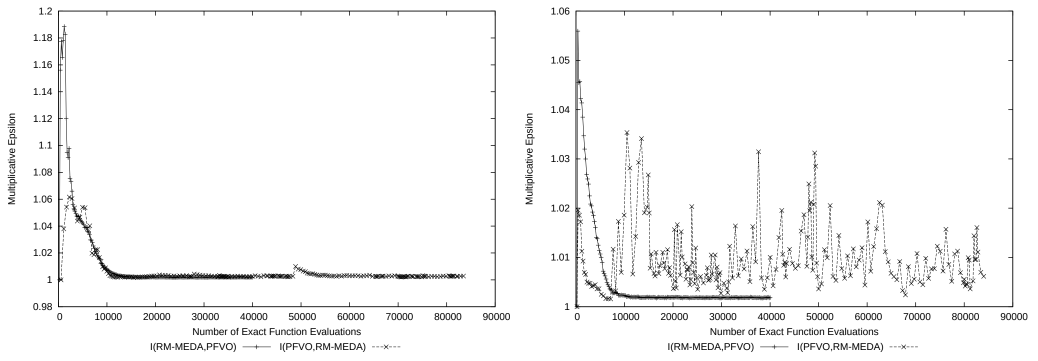
(b) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on ZDT3 (Left) and ZDT4 (Right)



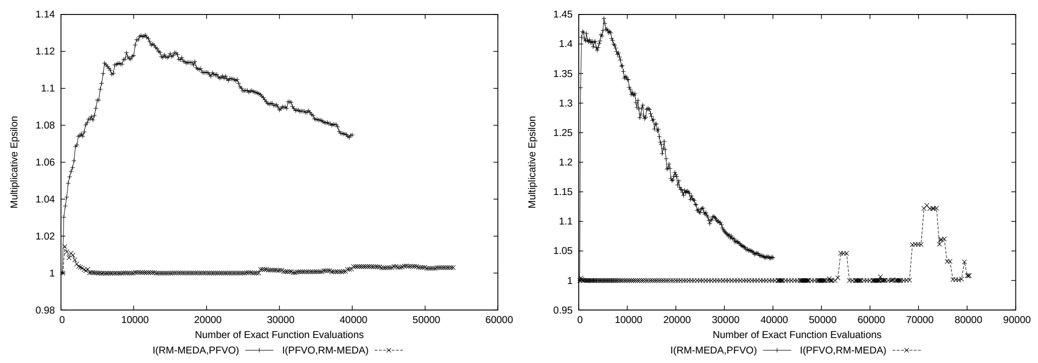
(c) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(d) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



(e) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)

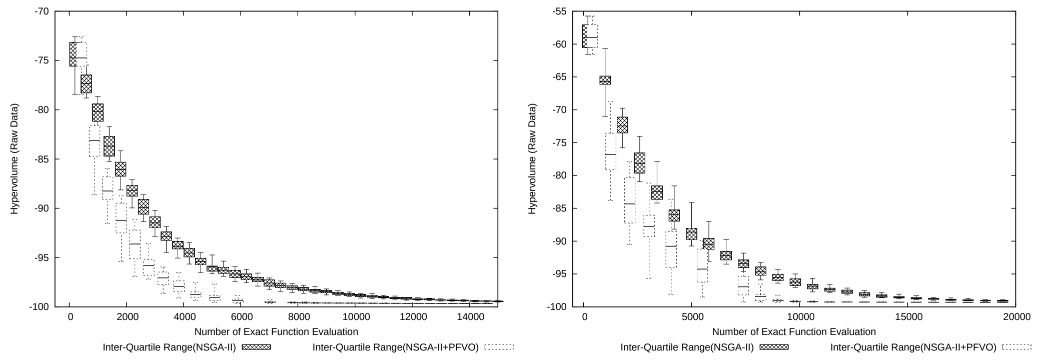


(f) I_ϵ(RM-MEDA) and I_ϵ(RM-MEDA + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

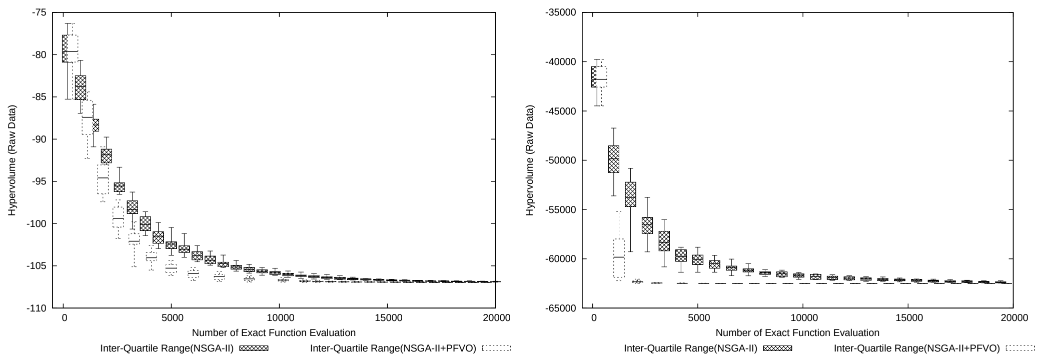
A.3 Performance Results: Box-plots

In this section we illustrate the box-plots of the indicator values provided so far. We have run each pair of the algorithms ("Host Optimizer" and "Host Optimizer + PFVO") for 30 times (with same random seed for a pair of algorithm but different seeds in 30 distinct runs) and indicator values are collected. From these box plots, we can get more explanatory illustration of the performance gain by PFVO. For the space constraints, we are not going to include the box plot results for multiplicative I_ϵ indicator. Interested readers are encouraged to refer to `http:\\khaled.ahsan.talukder.googlepages.com`.

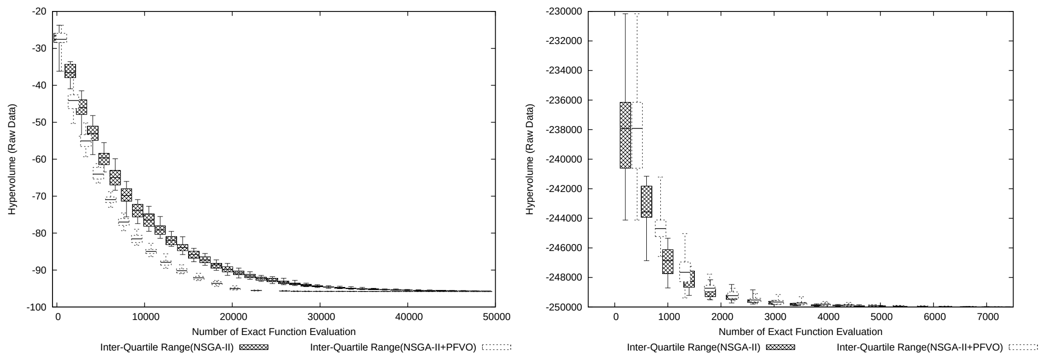
A.3.1 NSGA-II



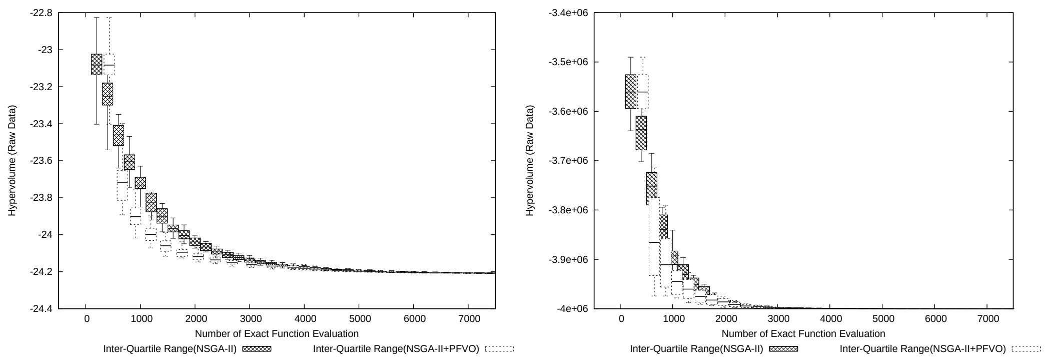
(g) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



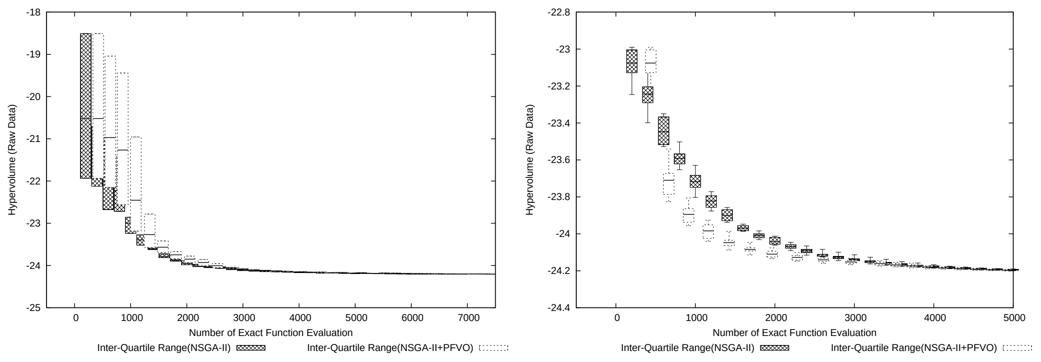
(h) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



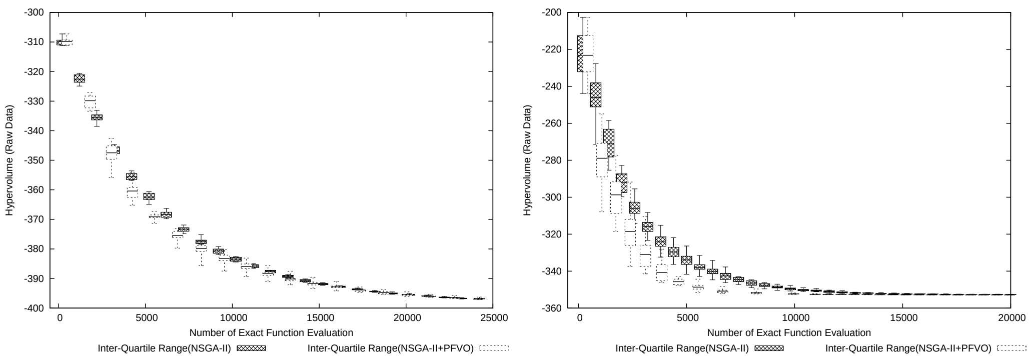
(i) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



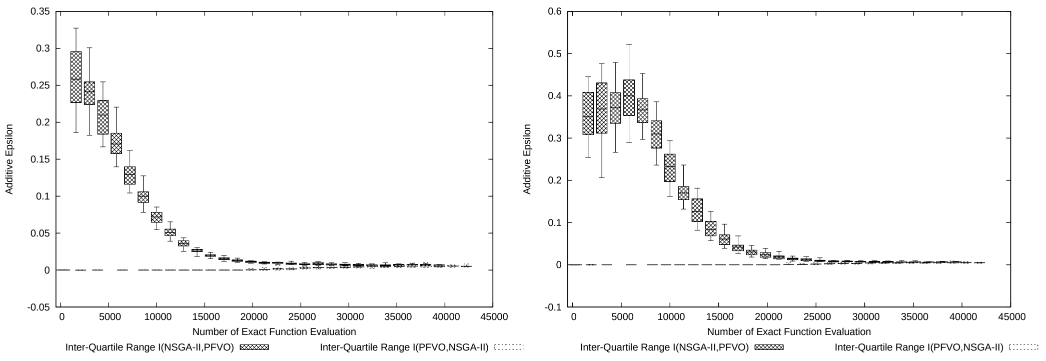
(j) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



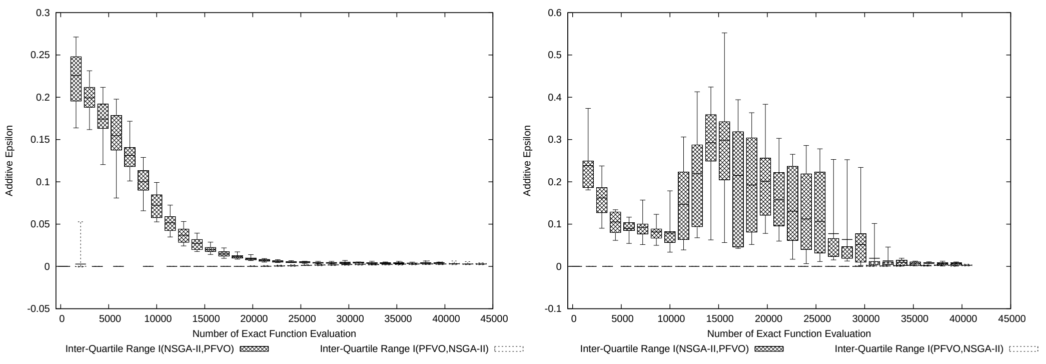
(k) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



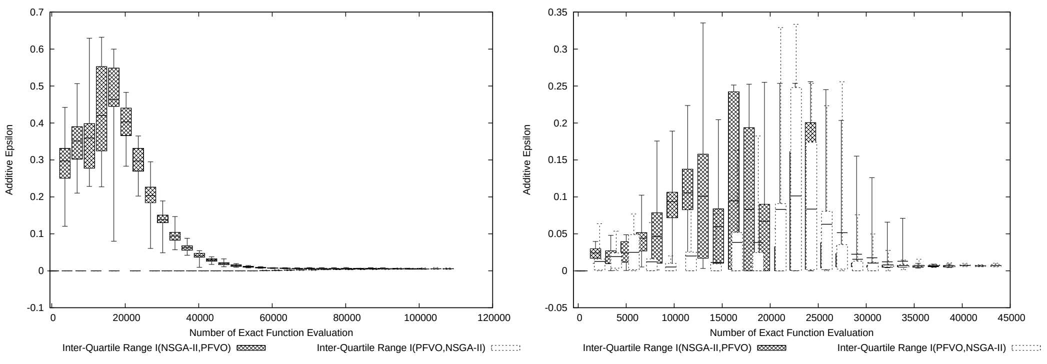
(l) I_H(NSGA-II) and I_H(NSGA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



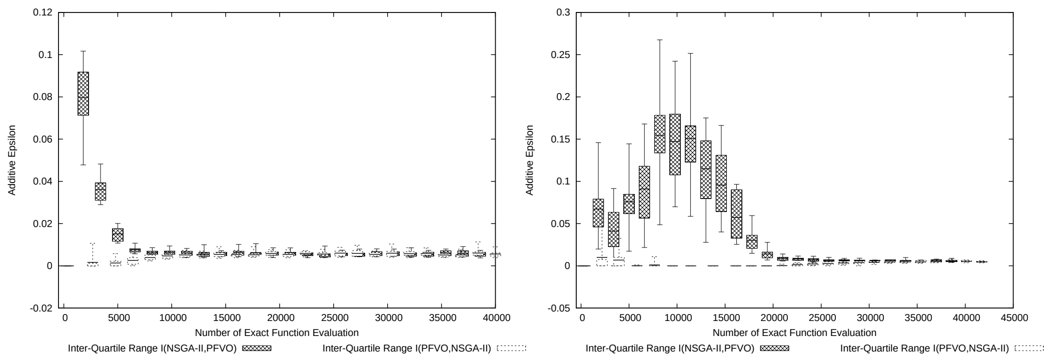
(m) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



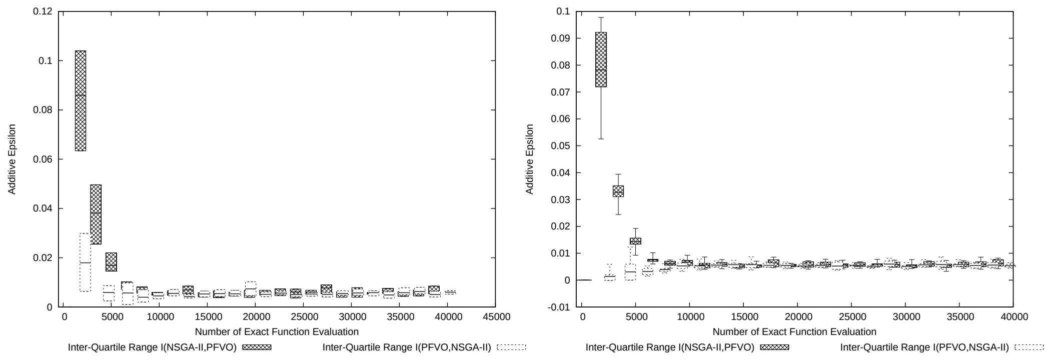
(n) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



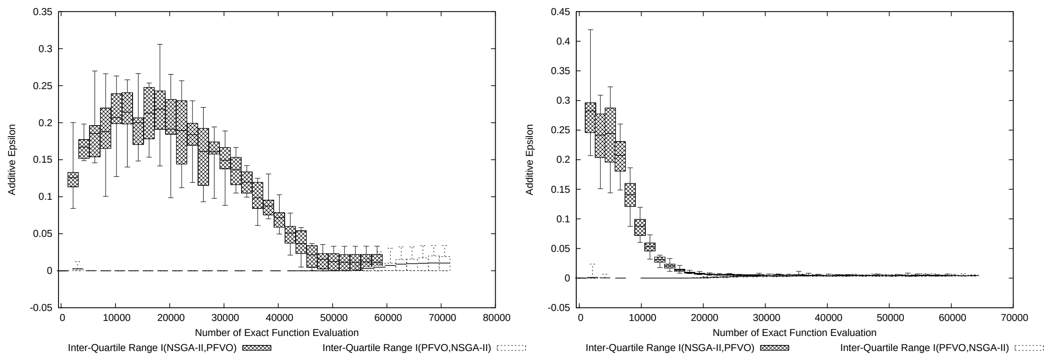
(o) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(p) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ2 (Left) and DTLZ3(Right)

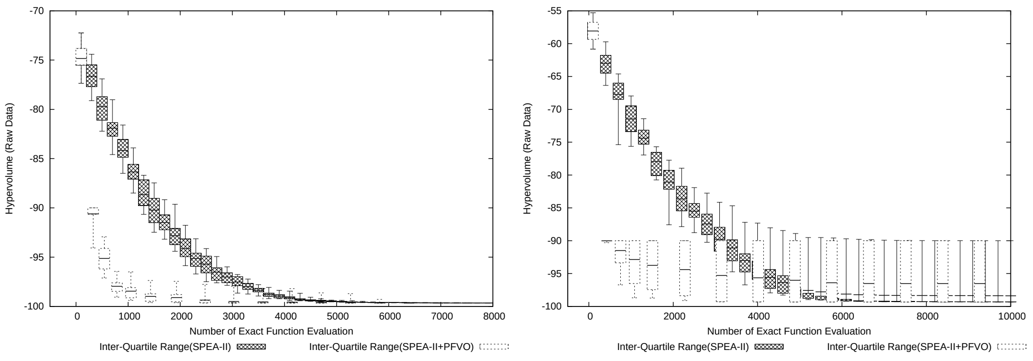


(q) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ4 (Left) and DTLZ5(Right)

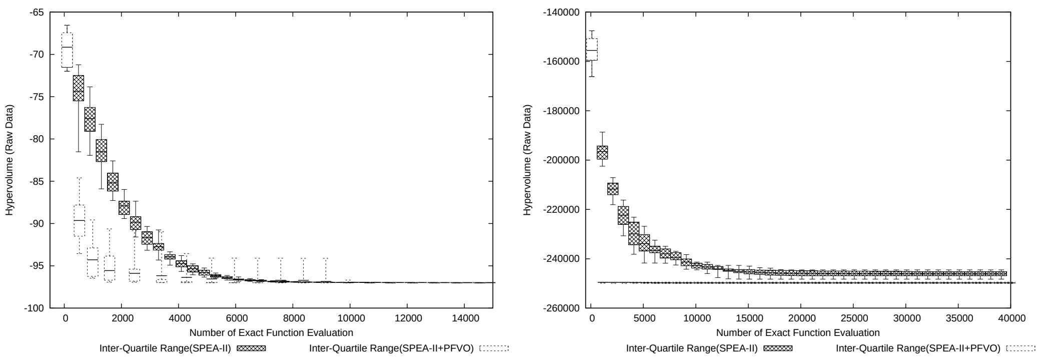


(r) $I_{\epsilon+}$(NSGA-II) and $I_{\epsilon+}$(NSGA-II + PFVO) on DTLZ6 (Left) and DTLZ7(Right)

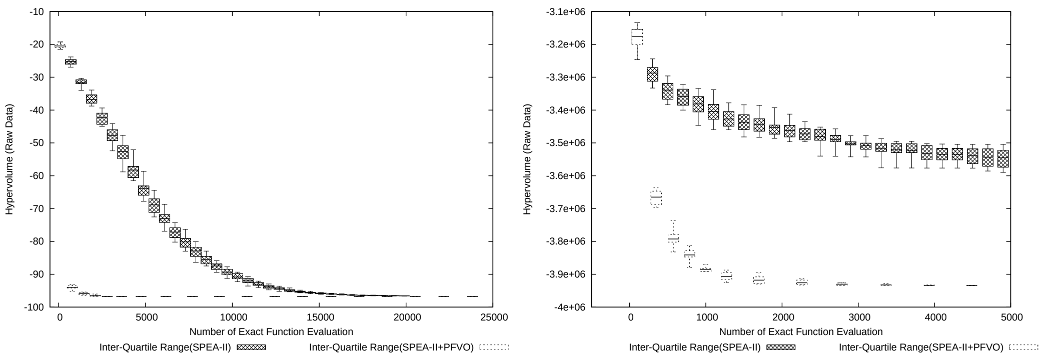
A.3.2 SPEA-II



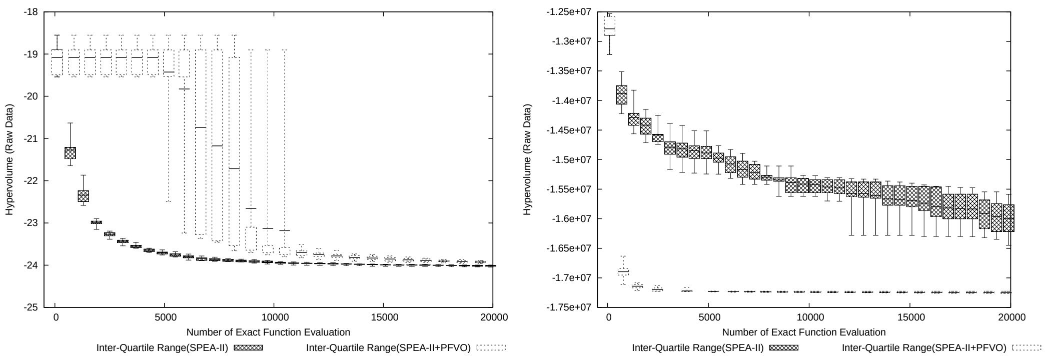
(a) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



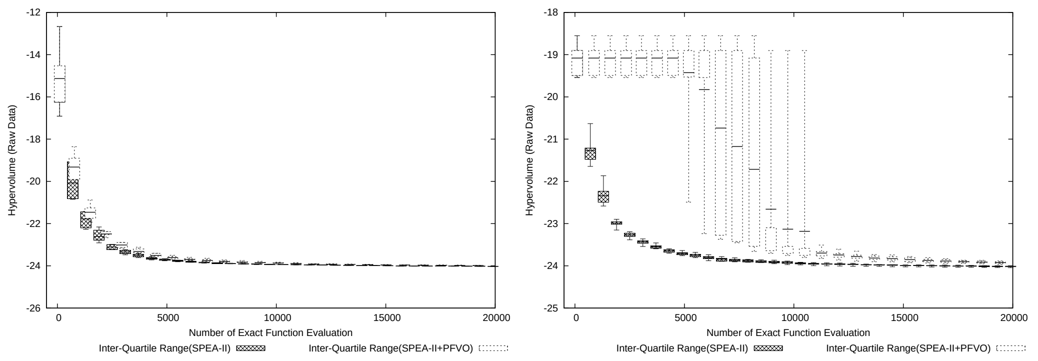
(b) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



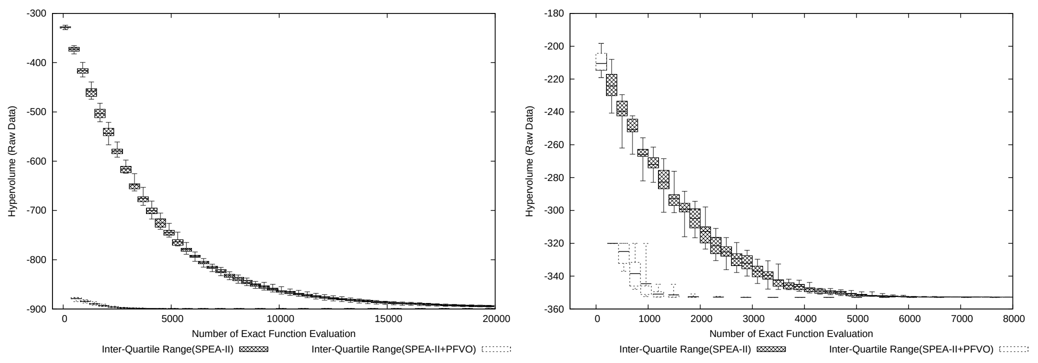
(c) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



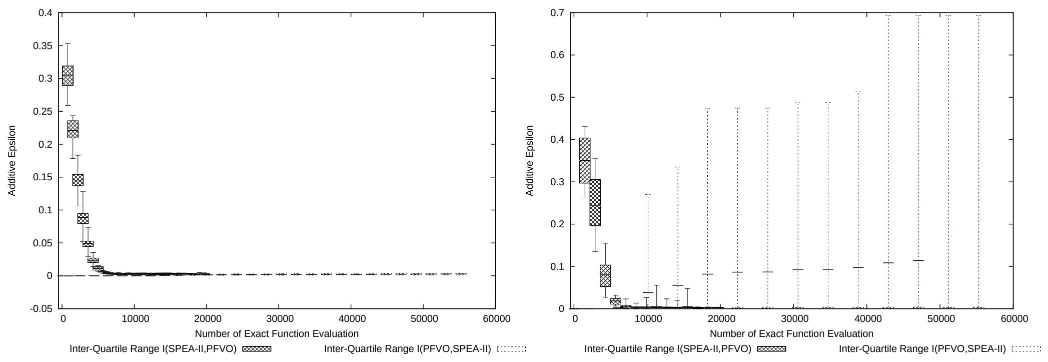
(d) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



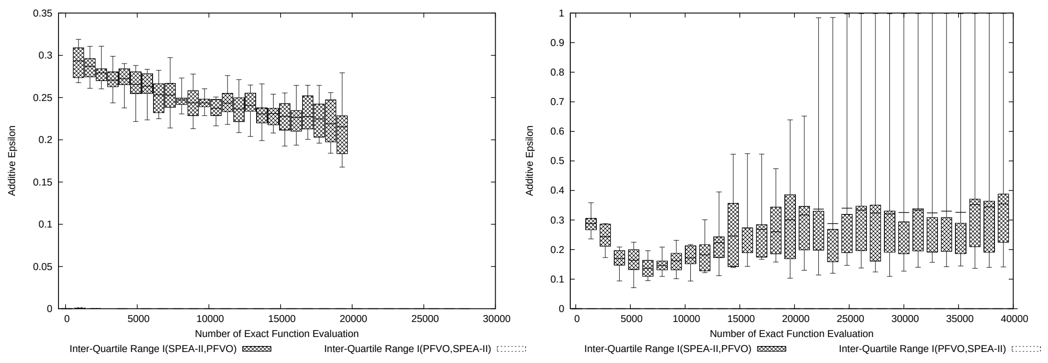
(e) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



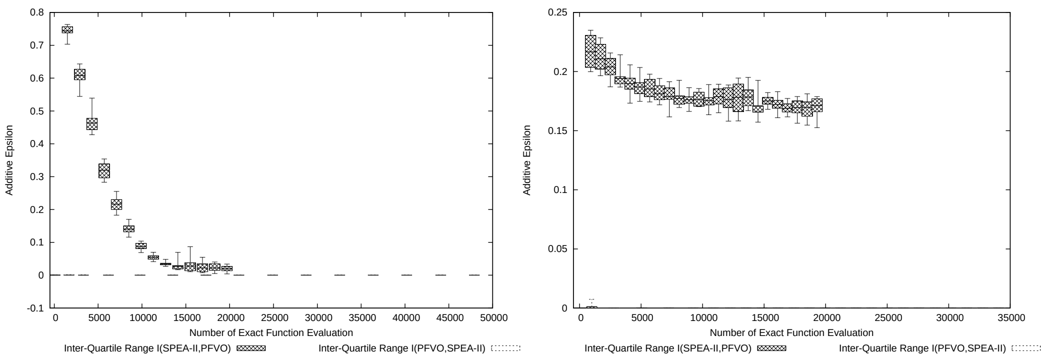
(f) I_H(SPEA-II) and I_H(SPEA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



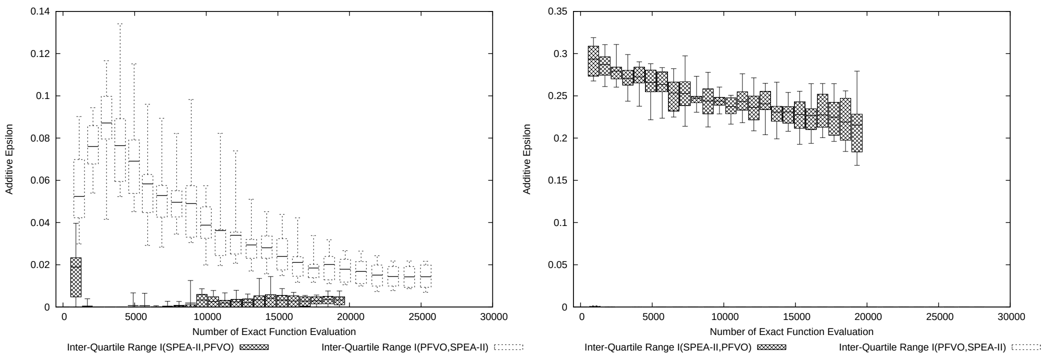
(g) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT1 (Left) and ZDT2 (Right)



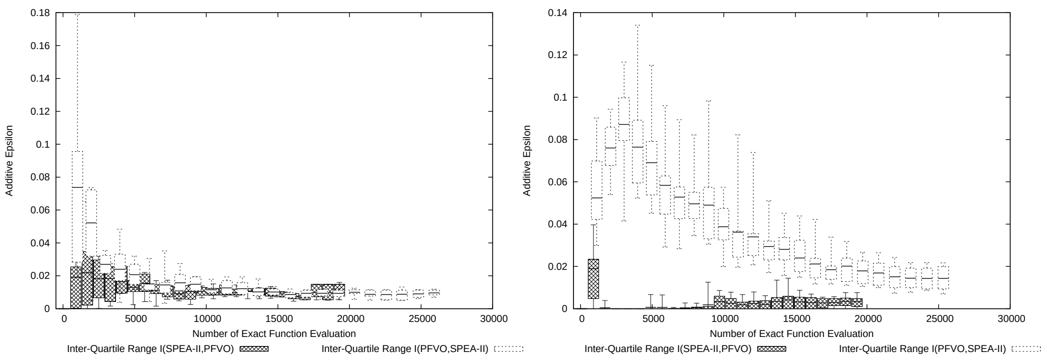
(h) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT3 (Left) and ZDT4 (Right)



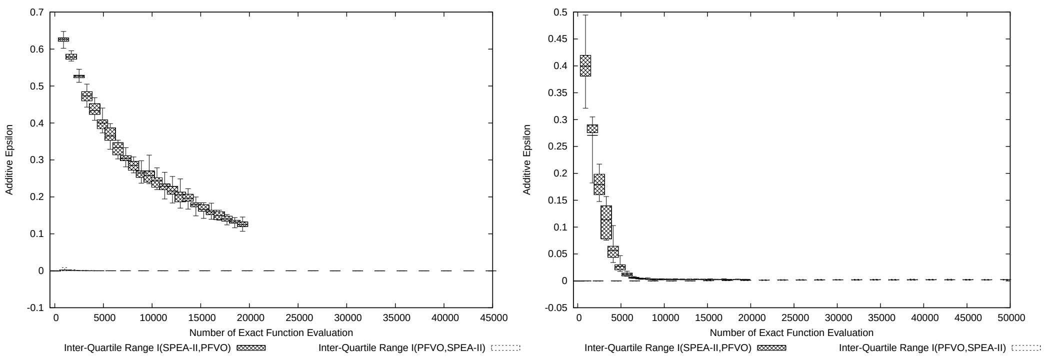
(i) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(j) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)

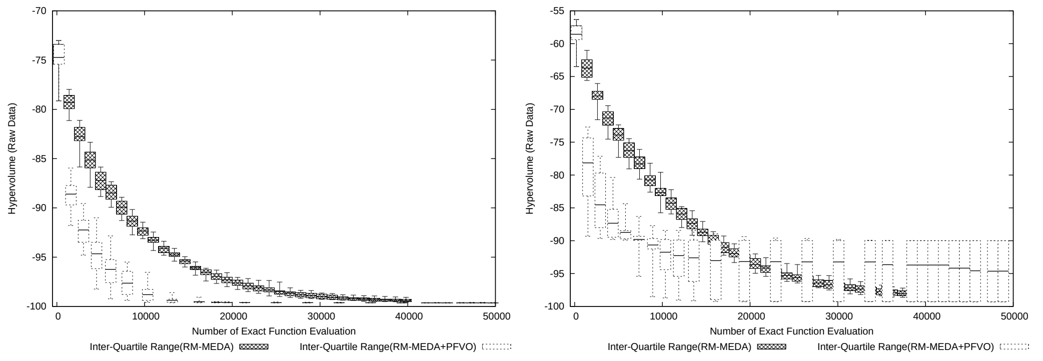


(k) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)

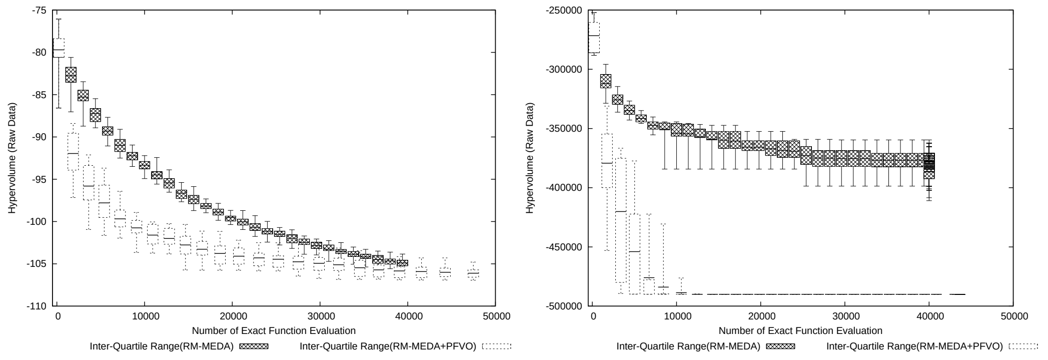


(l) $I_{\epsilon+}$(SPEA-II) and $I_{\epsilon+}$(SPEA-II + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

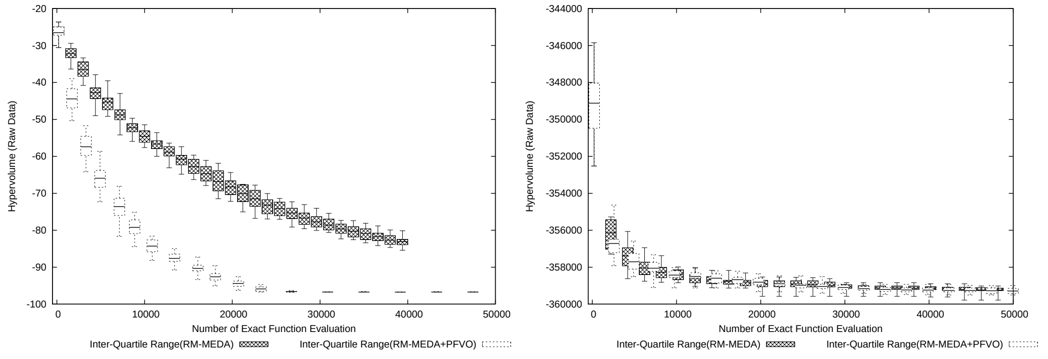
A.3.3 RM-MEDA



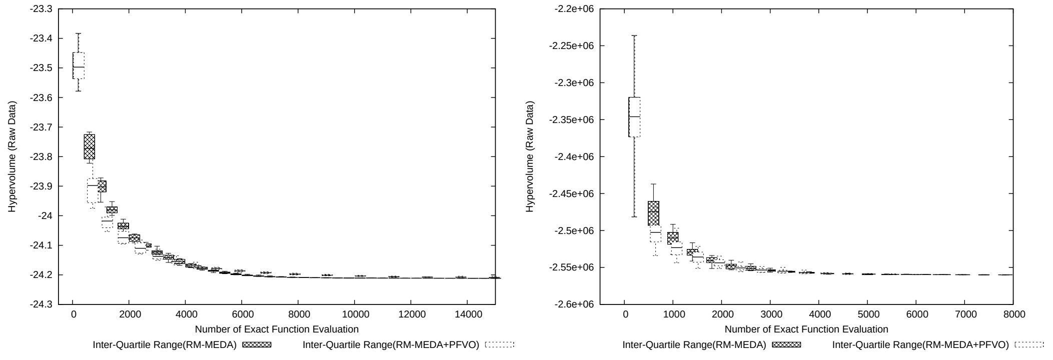
(a) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT1 (Left) and ZDT2 (Right)



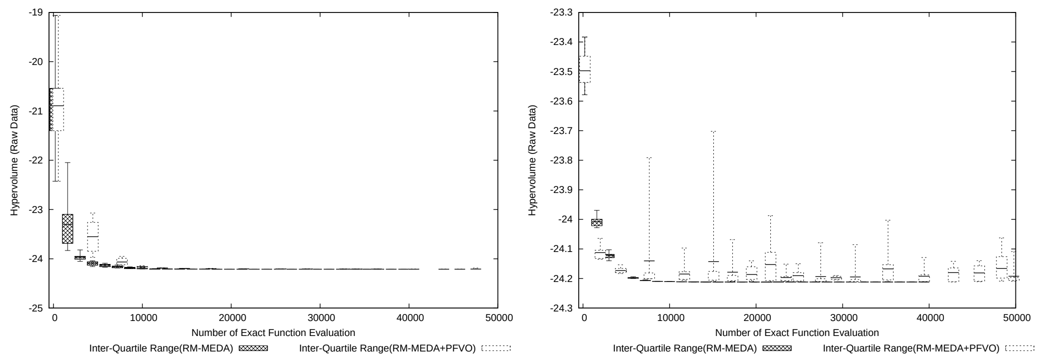
(b) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT3 (Left) and ZDT4 (Right)



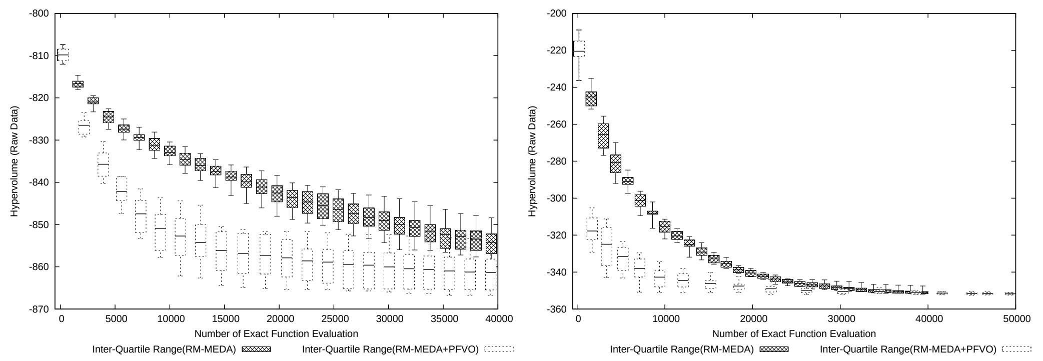
(c) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



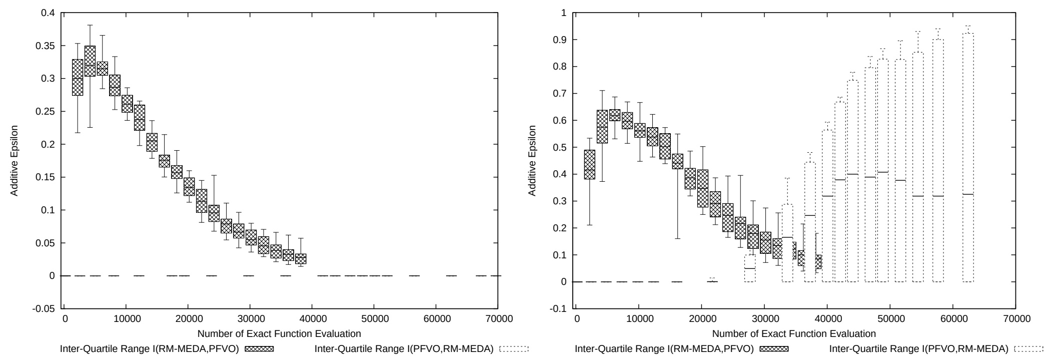
(d) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



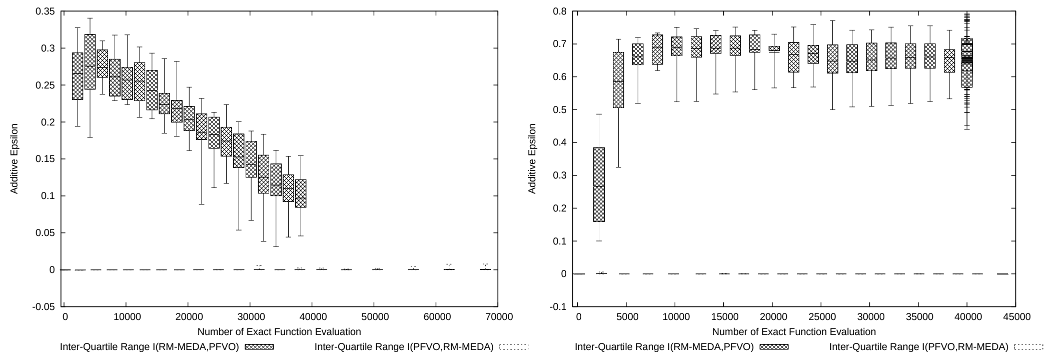
(e) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



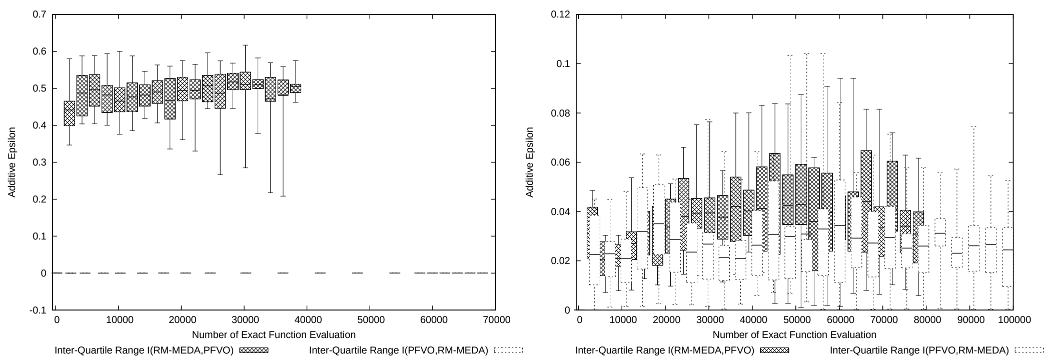
(f) I_H(RM-MEDA) and I_H(RM-MEDA + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)



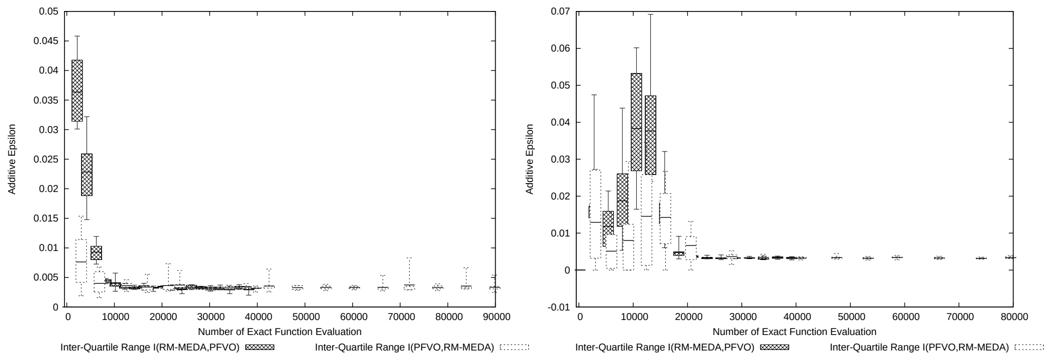
(g) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT1 (Left) and ZDT2 (Right)



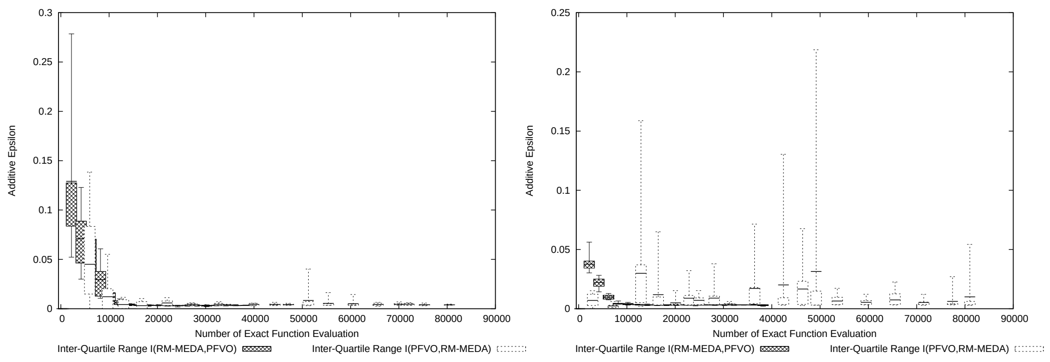
(h) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT3 (Left) and ZDT4 (Right)



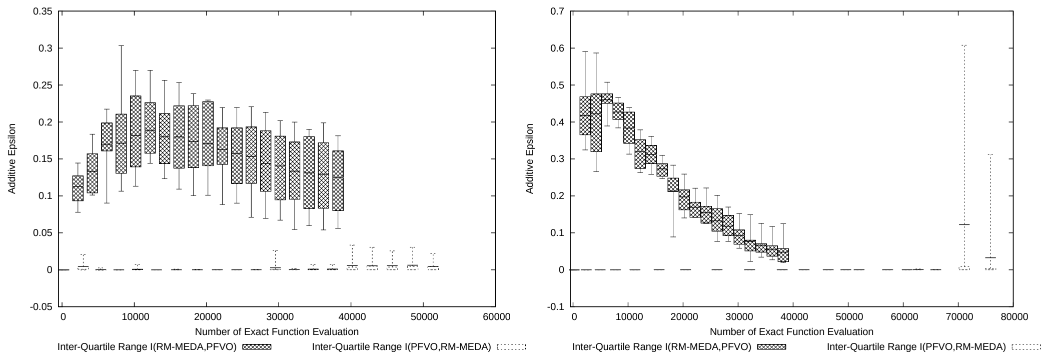
(i) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on ZDT6 (Left) and DTLZ1 (Right)



(j) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ2 (Left) and DTLZ3 (Right)



(k) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ4 (Left) and DTLZ5 (Right)



(l) $I_{\epsilon+}$(RM-MEDA) and $I_{\epsilon+}$(RM-MEDA + PFVO) on DTLZ6 (Left) and DTLZ7 (Right)

A.4 Statistical Analysis: Kruskal-Wallis Test Results

For statistical analysis, we have executed each pair of algorithms on every problems for 30 times and the I_H and $I_{\epsilon+}$ values were collected for every generation. From these values, we have conducted the Kruskal-Wallis [162] test by setting α values at 0.05 (i.e. 95% significance). Due to the space constraints, we are not able to provide the results for every generations. So, to maintain the consistency with the generational snapshots, we present only the p-values at the function evaluations (FE) same as the generational snapshots (i.e. at only three generational steps). Please note that the two algorithms (original MOEA and PFVO enhanced MOEA) having the same performance is considered as the null-hypothesis (H_0). When H_0 is rejected, we find a p-value, otherwise not. Details can be found in `http://khaled.ahsan.talukder.googlepages.com`.

A.4.1 Kruskal-Wallis Test on I_H Indicator Values

Problem	Function Evaluation	p-value (NSGA-II)	p-value (NSGA-II+PFVO)	Remark
ZDT1	800	1.0	4.55e-13	NSGA-II+PFVO wins
	2600	1.0	2.58e-13	NSGA-II+PFVO wins
	5200	1.0	2.58e-13	NSGA-II+PFVO wins
ZDT2	400	1.0	1.22e-10	NSGA-II+PFVO wins
	5200	1.0	2.58e-13	NSGA-II+PFVO wins
	9600	1.0	2.58e-13	NSGA-II+PFVO wins
ZDT3	600	0.999865	0.00013	NSGA-II+PFVO wins
	4800	1.0	2.58e-13	NSGA-II+PFVO wins
	6600	1.0	2.58e-13	NSGA-II+PFVO wins
ZDT4	800	1.0	3.75e-07	NSGA-II+PFVO wins
	22600	1.0	3.75e-07	NSGA-II+PFVO wins
	24800	1.0	3.75e-07	NSGA-II+PFVO wins
ZDT6	1280	1.0	1.53e-09	NSGA-II+PFVO wins
	8448	1.0	2.58e-13	NSGA-II+PFVO wins
	20992	1.0	2.58e-13	NSGA-II+PFVO wins
DTLZ1	650	–	–	Incomparable
	9320	–	–	Incomparable
	11514	0.998371	0.001629	NSGA-II+PFVO wins
DTLZ2	912	1.0	2.58e-13	NSGA-II+PFVO wins
	2058	1.0	2.58e-13	NSGA-II+PFVO wins
	3996	1.0	2.58e-13	NSGA-II+PFVO wins
DTLZ3	1304	0.999997	2.65e-06	NSGA-II+PFVO wins
	10188	1.0	3.67e-07	NSGA-II+PFVO wins
	18322	1.0	4.52e-08	NSGA-II+PFVO wins
DTLZ4	1574	–	–	Incomparable
	3114	–	–	Incomparable
	4052	–	–	Incomparable
DTLZ5	914	1.0	3.75e-07	NSGA-II+PFVO wins
	1966	1.0	3.75e-07	NSGA-II+PFVO wins
	4122	1.0	3.75e-07	NSGA-II+PFVO wins
DTLZ6	470	1.0	3.75e-07	NSGA-II+PFVO wins
	4746	1.0	3.75e-07	NSGA-II+PFVO wins
	42446	0.999987	1.26e-05	NSGA-II+PFVO wins
DTLZ7	1728	1.0	1.49e-11	NSGA-II+PFVO wins
	5534	1.0	2.58e-13	NSGA-II+PFVO wins
	11548	1.0	2.58e-13	NSGA-II+PFVO wins

Table A.1: p-values of the I_H indicator results for NSGA-II and NSGA-II+PFVO

Problem	Function Evaluation	p-value (SPEA-II)	p-value (SPEA-II+PFVO)	Remark
ZDT1	544	1.0	2.56e-13	SPEA-II+PFVO wins
	2752	1.0	2.56e-13	SPEA-II+PFVO wins
	6668	1.0	2.56e-13	SPEA-II+PFVO wins
ZDT2	534	1.0	2.74e-16	SPEA-II+PFVO wins
	3146	0.999958	4.17e-05	SPEA-II+PFVO wins
	6344	–	–	SPEA-II wins
ZDT3	532	1.0	2.56e-13	SPEA-II+PFVO wins
	2684	1.0	2.56e-13	SPEA-II+PFVO wins
	6560	1.0	2.56e-13	SPEA-II+PFVO wins
ZDT4	636	1.0	2.28e-09	SPEA-II+PFVO wins
	1774	1.0	2.43e-07	SPEA-II+PFVO wins
	39208	1.0	2.43e-07	SPEA-II+PFVO wins
ZDT6	910	1.0	2.56e-13	SPEA-II+PFVO wins
	2676	1.0	2.56e-13	SPEA-II+PFVO wins
	7438	1.0	2.56e-13	SPEA-II+PFVO wins
DTLZ1	234	1.0	2.76e-07	SPEA-II+PFVO wins
	1270	1.0	2.56e-07	SPEA-II+PFVO wins
	3080	1.0	2.56e-07	SPEA-II+PFVO wins
DTLZ2	100	–	–	Incomparable
	2686	–	–	Incomparable
	13676	3.75e-07	1.0	SPEA-II wins
DTLZ3	674	–	–	Incomparable
	2968	–	–	Incomparable
	6104	–	–	Incomparable
DTLZ4	1380	–	–	Incomparable
	2832	–	–	Incomparable
	6974	–	–	Incomparable
DTLZ5	230	–	–	Incomparable
	5086	0.000446753	0.999553	SPEA-II wins
	17358	1.06e-06	0.999999	SPEA-II wins
DTLZ6	490	1.0	3.75e-07	SPEA-II+PFVO wins
	5574	1.0	3.75e-07	SPEA-II+PFVO wins
	44548	1.0	3.75e-07	SPEA-II+PFVO wins
DTLZ7	644	1.0	3.75e-07	SPEA-II+PFVO wins
	1330	1.0	3.75e-07	SPEA-II+PFVO wins
	4196	1.0	3.75e-07	SPEA-II+PFVO wins

Table A.2: p-values of the I_H indicator results for SPEA-II and SPEA-II+PFVO

Problem	Function Evaluation	p-value (RM-MEDA)	p-value (RM-MEDA+PFVO)	Remark
ZDT1	2088	1.0	2.58e-13	RM-MEDA+PFVO wins
	6424	1.0	2.58e-13	RM-MEDA+PFVO wins
	14836	1.0	2.58e-13	RM-MEDA+PFVO wins
ZDT2	11612	1.0	2.58e-13	RM-MEDA+PFVO wins
	14388	1.0	2.58e-13	RM-MEDA+PFVO wins
	29496	1.0	2.58e-13	RM-MEDA+PFVO wins
ZDT3	5908	1.0	2.58e-13	RM-MEDA+PFVO wins
	30542	1.0	2.58e-13	RM-MEDA+PFVO wins
	72144	1.0	2.58e-13	RM-MEDA+PFVO wins
ZDT4	4784	1.0	3.75e-07	RM-MEDA+PFVO wins
	6990	1.0	3.75e-07	RM-MEDA+PFVO wins
	43668	1.0	3.75e-07	RM-MEDA+PFVO wins
ZDT6	3320	1.0	2.58e-13	RM-MEDA+PFVO wins
	11578	1.0	2.58e-13	RM-MEDA+PFVO wins
	69758	1.0	2.58e-13	RM-MEDA+PFVO wins
DTLZ1	200	–	–	Incomparable
	20000	–	–	Incomparable
	58600	–	–	Incomparable
DTLZ2	468	0.999212	0.000788392	RM-MEDA wins
	3036	1.0	3.75157e-07	RM-MEDA+PFVO wins
	7342	1.0	3.75157e-07	RM-MEDA+PFVO wins
DTLZ3	644	–	–	Incomparable
	11940	–	–	Incomparable
	15938	–	–	Incomparable
DTLZ4	7720	1.0	3.75e-07	RM-MEDA+PFVO wins
	10618	–	–	Incomparable
	14748	–	–	Incomparable
DTLZ5	710	1.0	1.26e-05	RM-MEDA+PFVO wins
	3242	1.0	3.75e-07	RM-MEDA+PFVO wins
	6154	0.000346721	0.999653	RM-MEDA wins
DTLZ6	3182	0.999997	2.65e-06	RM-MEDA+PFVO wins
	9146	0.999987	1.26e-05	RM-MEDA+PFVO wins
	15554	0.999987	1.26e-05	RM-MEDA+PFVO wins
DTLZ7	1066	1.0	3.75e-07	RM-MEDA+PFVO wins
	15220	1.0	3.75e-07	RM-MEDA+PFVO wins
	51978	1.0	3.75e-07	RM-MEDA+PFVO wins

Table A.3: p-values of the I_H indicator results for RM-MEDA and RM-MEDA+PFVO

A.4.2 Kruskal-Wallis Test on $I_{\epsilon+}$ Indicator Values

Problem	Function Evaluation	p-value (NSGA-II)	p-value (NSGA-II+PFVO)	Remark
ZDT1	800	1.0	2.58e-13	NSGA-II+PFVO wins
	2600	1.0	2.58e-13	NSGA-II+PFVO wins
	5200	1.0	2.58e-13	NSGA-II+PFVO wins
ZDT2	400	1.0	4.55e-13	NSGA-II+PFVO wins
	5200	1.0	2.58e-13	NSGA-II+PFVO wins
	9600	1.0	2.57e-13	NSGA-II+PFVO wins
ZDT3	600	1.0	4.55e-13	NSGA-II+PFVO wins
	4800	1.0	2.58e-13	NSGA-II+PFVO wins
	6600	1.0	2.58e-13	NSGA-II+PFVO wins
ZDT4	800	1.0	3.75e-07	NSGA-II+PFVO wins
	22600	1.0	2.28e-09	NSGA-II+PFVO wins
	24800	1.0	2.28e-09	NSGA-II+PFVO wins
ZDT6	1280	1.0	2.23e-12	NSGA-II+PFVO wins
	8448	1.0	5.82e-18	NSGA-II+PFVO wins
	20992	1.0	5.82e-18	NSGA-II+PFVO wins
DTLZ1	650	–	–	Incomparable
	9320	0.99978	0.000220324	NSGA-II+PFVO wins
	11514	0.999473	0.000526611	NSGA-II+PFVO wins
DTLZ2	912	1.0	2.58e-13	NSGA-II+PFVO wins
	2058	1.0	2.58e-13	NSGA-II+PFVO wins
	3996	1.0	2.58e-13	NSGA-II+PFVO wins
DTLZ3	1304	0.999999	1.06e-06	NSGA-II+PFVO wins
	10188	1.0	2.43e-07	NSGA-II+PFVO wins
	18322	1.0	3.45e-07	NSGA-II+PFVO wins
DTLZ4	1574	–	–	Incomparable
	3114	–	–	Incomparable
	4052	–	–	Incomparable
DTLZ5	914	1.0	3.75e-07	NSGA-II+PFVO wins
	1966	1.0	3.75e-07	NSGA-II+PFVO wins
	4122	1.0	3.75e-07	NSGA-II+PFVO wins
DTLZ6	470	0.999987	1.26e-05	NSGA-II+PFVO wins
	4746	1.0	3.75e-07	NSGA-II+PFVO wins
	42446	0.999987	1.26e-05	NSGA-II+PFVO wins
DTLZ7	1728	1.0	2.58e-13	NSGA-II+PFVO wins
	5534	1.0	2.58e-13	NSGA-II+PFVO wins
	11548	1.0	2.58e-13	NSGA-II+PFVO wins

Table A.4: p-values of the $I_{\epsilon+}$ indicator results for NSGA-II and NSGA-II+PFVO

Problem	Function Evaluation	p-value (SPEA-II)	p-value (SPEA-II+PFVO)	Remark
ZDT1	544	1.0	2.56e-13	SPEA-II+PFVO wins
	2752	1.0	2.40e-13	SPEA-II+PFVO wins
	6668	1.0	2.56e-13	SPEA-II+PFVO wins
ZDT2	534	1.0	2.56e-13	SPEA-II+PFVO wins
	3146	1.0	5.18e-15	SPEA-II+PFVO wins
	6344	0.999718	0.000281941	SPEA-II+PFVO wins
ZDT3	532	1.0	2.56e-13	SPEA-II+PFVO wins
	2684	1.0	2.49e-13	SPEA-II+PFVO wins
	6560	1.0	2.39e-15	SPEA-II+PFVO wins
ZDT4	636	1.0	3.75e-07	SPEA-II+PFVO wins
	1774	1.0	3.38e-07	SPEA-II+PFVO wins
	39208	1.0	2.28e-09	SPEA-II+PFVO wins
ZDT6	910	1.0	2.56e-13	SPEA-II+PFVO wins
	2676	1.0	2.54e-13	SPEA-II+PFVO wins
	7438	1.0	2.24e-13	SPEA-II+PFVO wins
DTLZ1	234	1.0	2.28e-09	SPEA-II+PFVO wins
	1270	1.0	3.75e-07	SPEA-II+PFVO wins
	3080	1.0	3.75e-07	SPEA-II+PFVO wins
DTLZ2	100	–	–	Incomparable
	2686	3.75e-07	1.0	SPEA-II wins
	13676	3.75e-07	1.0	SPEA-II wins
DTLZ3	674	1.0	3.75e-07	SPEA-II+PFVO wins
	2968	1.0	3.75e-07	SPEA-II+PFVO wins
	6104	1.0	3.75e-07	SPEA-II+PFVO wins
DTLZ4	1380	–	–	Incomparable
	2832	–	–	Incomparable
	6974	–	–	Incomparable
DTLZ5	230	1.0	2.28e-09	Incomparable
	5086	3.04e-07	1.0	SPEA-II wins
	17358	3.75e-07	1.0	SPEA-II wins
DTLZ6	490	1.0	3.75e-07	SPEA-II+PFVO wins
	5574	1.0	3.75e-07	SPEA-II+PFVO wins
	44548	1.0	3.75e-07	SPEA-II+PFVO wins
DTLZ7	644	1.0	3.75e-07	SPEA-II+PFVO wins
	1330	1.0	3.75e-07	SPEA-II+PFVO wins
	4196	1.0	1.44e-08	SPEA-II+PFVO wins

Table A.5: p-values of the $I_{\epsilon+}$ indicator results for SPEA-II and SPEA-II+PFVO

Problem	Function Evaluation	p-value (RM-MEDA)	p-value (RM-MEDA+PFVO)	Remark
ZDT1	2088	1.0	2.58e-13	RM-MEDA+PFVO wins
	6424	1.0	2.58e-13	RM-MEDA+PFVO wins
	14836	1.0	2.58e-13	RM-MEDA+PFVO wins
ZDT2	11612	1.0	2.58e-13	RM-MEDA+PFVO wins
	14388	1.0	2.57e-13	RM-MEDA+PFVO wins
	29496	–	–	Incomparable
ZDT3	5908	1.0	2.58e-13	RM-MEDA+PFVO wins
	30542	1.0	2.58e-13	RM-MEDA+PFVO wins
	72144	1.0	2.58e-13	RM-MEDA+PFVO wins
ZDT4	4784	1.0	3.75e-07	RM-MEDA+PFVO wins
	6990	1.0	3.75e-07	RM-MEDA+PFVO wins
	43668	1.0	3.75e-07	RM-MEDA+PFVO wins
ZDT6	3320	1.0	2.56e-13	RM-MEDA+PFVO wins
	11578	1.0	2.54e-13	RM-MEDA+PFVO wins
	69758	1.0	2.57e-13	RM-MEDA+PFVO wins
DTLZ1	200	–	–	Incomparable
	20000	–	–	Incomparable
	58600	–	–	Incomparable
DTLZ2	468	0.99978	0.000220324	RM-MEDA wins
	3036	1.0	3.75e-07	RM-MEDA+PFVO wins
	7342	0.999987	1.26e-05	RM-MEDA+PFVO wins
DTLZ3	644	0.999212	0.000788392	RM-MEDA+PFVO wins
	11940	0.99992	8.023e-05	RM-MEDA+PFVO wins
	15938	0.998371	0.00162872	RM-MEDA+PFVO wins
DTLZ4	7720	1.0	3.75e-07	RM-MEDA+PFVO wins
	10618	0.99773	0.00227035	RM-MEDA+PFVO wins
	14748	–	–	Incomparable
DTLZ5	710	0.99992	8.02e-05	RM-MEDA+PFVO wins
	3242	1.0	3.75e-07	RM-MEDA+PFVO wins
	6154	–	–	Incomparable
DTLZ6	3182	1.0	3.75e-07	RM-MEDA+PFVO wins
	9146	1.0	3.75e-07	RM-MEDA+PFVO wins
	15554	1.0	3.75e-07	RM-MEDA+PFVO wins
DTLZ7	1066	1.0	3.75e-07	RM-MEDA+PFVO wins
	15220	1.0	3.75e-07	RM-MEDA+PFVO wins
	51978	1.0	3.75e-07	RM-MEDA+PFVO wins

Table A.6: p-values of the I_ϵ indicator results for RM-MEDA and RM-MEDA+PFVO

Bibliography

[1] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. West Sussex, England: John Wiley and Sons, Ltd, 2002.

[2] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. NY, USA: Kluwer Academic/Plenum Publishers, 2002.

[3] R. Sarker, *Evolutionary Optimization*, X. Yao, Ed. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[4] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, *Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems*. Springer, 2004.

[5] N. Queipo, R. Haftka, W. S. and. T. Goel, and R. Vaidyanathan, "Surrogate-based analysis and optimization," *Journal of Progress in Aerospace Sciences*, vol. 41, pp. 1–28, 2005.

[6] H. Kaji and H. Kita, "Acceleration of experiment-based evolutionary multi-objective optimization of internal-combustion engine controllers using fitness estimation," pp. 1777–1784, September 2007.

[7] M. Farina, "A Neural Network Based Generalized Response Surface Multiobjective Evolutionary Algorithm," in *IEEE Congress on Evolutionary Computation (CEC-2002)*, vol. 1. Honolulu, HI: IEEE Press, May 2002, pp. 956–961.

[8] D. Van Veldhuizen, J. Zydallis, and G. Lamont, “Considerations in engineering parallel multiobjective evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 144–173, April 2003.

[9] F. de Toro Negro, J. Ortega, E. Ros, S. Mota, B. Paechter, and J. M. Martín, “PSFGA: Parallel Processing and Evolutionary Computation for Multiobjective Optimisation,” *Parallel Computing*, vol. 30, no. 5–6, pp. 721–739, May – June 2004.

[10] A. J. Nebro, E. Alba, and F. Luna, “Multi-Objective Optimization using Grid Computing,” *Soft Computing*, vol. 11, no. 6, pp. 531–540, 2007.

[11] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, “Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 1, pp. 66–76, January 2007.

[12] Y.-S. Ong, P. B. Nair, and K. Lum, “Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Design,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, August 2006.

[13] Y. Jin, M. Olhofer, and B. Sendhoff, “A Framework for Evolutionary Optimization with Approximate Fitness Functions,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, October 2002.

[14] A. Messac and A. A. Mullur, “A computationally efficient metamodeling approach for expensive multiobjective optimization,” *Optimization and Engineering*, vol. 9, no. 1, pp. 37–67, March 2008.

[15] G. E. P. Box and K. B. Wilson, “On the experimental attainment of optimum conditions,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 13, no. 1, pp. 1–45, 1951.

[16] I. Kampolis, A. Zymaris, V. Asouti, and K. Giannakoglou, “Multilevel optimization strategies based on metamodel-assisted evolutionary algorithms, for computationally expensive problems,” in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC ’07)*, September 2007, pp. 4116–4123.

[17] I. Voutchkov and A. J. Keane, "Multiobjective Optimization using Surrogates," in *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, I. C. Parmee, Ed. Bristol, UK: The Institute for People-centred Computation, April 2006, pp. 167–175.

[18] A. K. A. Talukder, M. Kirley, and R. Buyya, "A pareto following variation operator for fast-converging multiobjective evolutionary algorithms," in *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2008, pp. 721–728.

[19] A. K. A. Talukder and M. Kirley, "A pareto following variation operator for evolutionary dynamic multi-objective optimization," *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2270–2277, June 2008.

[20] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford, UK: Oxford University Press, 1996.

[21] A. Osyczka, "Multicriteria Optimization for Engineering Design," in *Design Optimization*, J. S. Gero, Ed. Academic Press, pp. 193 – 227.

[22] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Genetic Algorithms: Proceedings of the Fifth International Conference*. Morgan Kaufmann, 1993, pp. 416–423.

[23] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 82–87 vol.1.

[24] N. Srinivas and K. Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[25] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,” *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[26] C. A. C. Coello and G. B. Lamont, *Applications of Multi-Ojective Evolutionary Algorithms*, ser. Advances In Natural Computation. World Scientific Publishing, Singapore, 2004, vol. 1.

[27] I. Das and J. Dennis, “Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems,” *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[28] A. Messac, A. I. Yahaya, and C. A. Mattson, “The Normalized Normal Constraint Method for Generating the Pareto Frontier,” *Structural and Multidisciplinary Optimization, Journal of the International Society of Structural and Multidisciplinary Optimization (ISSMO)*, vol. 25, no. 2, pp. 86 – 98, 2003.

[29] A. Messac and C. A. Mattson, “Normal Constraint Method with Guarantee of Even Representation of Complete Pareto Frontier,” *AIAA Journal*, vol. 42, no. 10, pp. 2101 – 2111, October 2004.

[30] H. P. Benson, *Multicriteria Optimization*. Springer.

[31] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. Kluwer’s International Series in Operations Research and Management Science. Boston, USA: Kluwer Academic Publishers, 1999, vol. 12.

[32] C. A. C. Coello, “Recent Trends in Evolutionary Multiobjective Optimization,” in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, A. A. L. J. Lakhmi Jain, Xindong Wu and R. Goldberg, Eds. Springer Berlin Heidelberg, March 2006, pp. 7 – 32.

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

[34] D. E. Goldberg, *Genetic Algorihms in Search, Optimization and Machine Learning*, fifth indian reprint ed. Singapore: Pearson Education Pte. Ltd., 2002.

[35] P. Hajela and C. Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural and Multidisciplinary Optimization*, vol. 4, no. 2, pp. 99–107, June 1992.

[36] R. S. Zebulum and M. Vellasco, "A Multi-Objective Optimisation Methodology Applied to the Synthesis of Low-Power Operational Amplifiers," in *In I.J. Chueiri and C.A. dos Reis Filho, editors, Proc. XIII Int. Conf. on Microelectronics and Packaging, volume I*. The Brazilian Microelectronics Society, 1998, pp. 264–271.

[37] P. Hajela and C. Y. Lin, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Structural and Multidisciplinary Optimization*, vol. 14, no. 1, pp. 63 – 69, August 1997.

[38] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 402–412, Aug 2002.

[39] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1985, pp. 93–100.

[40] ——, "Multiple objective optimization with vector evaluated genetic algorithms," Ph.D. dissertation, Vanderbilt University, Nashville, TN, 1984.

[41] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 42–50.

[42] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[43] 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.

[44] J. Knowles and D. Corne, “Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors,” *IEEE Transactions on Evolutionary Computation*, vol. 7, pp. 100–116, April 2003.

[45] J. D. Knowles and D. Corne, “Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy,” *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[46] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[47] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, Trosset, and M. W., “A Rigorous Framework for Optimization of Expensive Functions by Surrogates,” Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, Technical Report TR-98-47, November 1998.

[48] A. A. Giunta and L. T. Watson, “A Comparison Of Approximation Modeling Techniques: Polynomial Versus Interpolating Models,” Multidisciplinary Analysis and Design (MAD), Center for Advanced Vehicles, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Tech. Rep., September 1998.

[49] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree, “Comparison Of Response Surface And Kriging Models For Multidisciplinary Design Optimization,” Tech. Rep., September 1998.

[50] R. Snieder, “The role of nonlinearity in inverse problems,” *Inverse Problems*, vol. 14, no. 18.

[51] D. Chafekar, L. Shi, K. Rasheed, and J. Xuan, "Multiobjective GA Optimization Using Reduced Models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 2, pp. 261–265, May 2005.

[52] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 3, Sept. 2005, pp. 2832–2839.

[53] K. Sastry, C. F. Lima, and D. E. Goldberg, "Evaluation relaxation using substructural information and linear estimation," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006, pp. 419–426.

[54] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and Multidisciplinary Optimization*, vol. 35, no. 6, pp. 561–569, June 2008.

[55] K. Harada, J. Sakuma, and S. Kobayashi, "Local Search for Multiobjective Function Optimization: Pareto Descent Method," in *The 8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*. Seattle, Washington, USA: ACM Press, 2006, pp. 659–666.

[56] P. A. N. Bosman and E. D. de Jong, "Exploiting Gradient Information in Numerical Multi-Objective Evolutionary Optimization," in *The 7th Annual Conference on Genetic and Evolutionary Computation (GECCO-2005)*. Washington DC, USA: ACM Press, 2005, pp. 755–762.

[57] ——, "Combining Gradient Techniques for Numerical Multi-Objective Evolutionary Optimization," in *The 8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*. Seattle, Washington, USA: ACM Press, 2006, pp. 627–634.

[58] M. Brown and R. E. Smith, "Effective Use of Directional Information in Multiobjective Evolutionary Computation," in *The 5th Annual Conference on Genetic and*

Evolutionary Computation (GECCO-2003), ser. LNCS, vol. 2723/2003. Chicago, Illinois, USA: Springer, 2003, pp. 778–789.

[59] ——, “Directed Multiobjective Optimization,” *International Journal of Computers, Systems and Signals*, vol. 6, no. 1, pp. 3–17, 2005.

[60] Y. S. Ong, P. B. Nair, and A. J. Keane, “Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling,” *American Institute of Aeronautics and Astronautics Journal*, vol. 41, no. 4, pp. 689–696, 2003.

[61] T. Ray and W. Smith, “Surrogate assisted evolutionary algorithm for multiobjective optimization,” in *Proceedings of The 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2006, pp. 1–8.

[62] L. Santana-Quintero, V. Serrano-Hernandez, C. C. Coello, A. Hernandez-Diaz, and J. Molina, “Use of Radial Basis Functions and Rough Sets for Evolutionary Multi-Objective Optimization,” in *Proceedings of The IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, April 2007, pp. 107–114.

[63] S. F. Adra, I. Griffin, and P. J. Fleming, “An Informed Convergence Accelerator for Evolutionary Multiobjective Optimiser,” in *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*. London, England: ACM Press, NY, USA, 2007, pp. 734–740.

[64] H. Soh, Y. S. Ong, M. Salahuddin, T. Hung, and B. S. Lee, “Playing in the Objective Space: Coupled Approximators for Multi-Objective Optimization,” in *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (ICDM-2007)*. Honolulu, HI: IEEE Press, April 2007, pp. 325–332.

[65] Y.-S. Hong, H. Lee, and M.-J. Tahk, “Acceleration Of The Convergence Speed Of Evolutionary Algorithms Using Multi-Layer Neural Networks,” *Engineering Optimization*, vol. 35, no. 1, pp. p91 –, 2003.

[66] D. Fonseca, D. Navaresse, and G. Moynihan, “Simulation metamodeling through artificial neural networks,” *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7, pp. 177–183, April 2003.

[67] P. K. Nain and K. Deb, “Computationally Effective Search and Optimization Procedure Using Coarse to Fine Approximations,” in *IEEE Congress on Evolutionary Computation (CEC-2003)*, vol. 3. Canberra, NSW, Australia: IEEE Press, December 2003, pp. 2081–2088.

[68] Z. Zhou, Y. S. Ong, and P. Nair, “Hierarchical surrogate-assisted evolutionary optimization framework,” in *Proceedings of The IEEE Congress on Evolutionary Computation. (CEC ’04)*, vol. 2, June 2004, pp. 1586–1593.

[69] K. S. Won and T. Ray, “Performance of kriging and cokriging based surrogate models within the unified framework for surrogate assisted optimization,” *Proceedings of The IEEE Congress on Evolutionary Computation (CEC ’04)*, vol. 2, pp. 1577–1585, June 2004.

[70] D. Buche, N. Schraudolph, and P. Koumoutsakos, “Accelerating Evolutionary Algorithms with Gaussian Process Fitness Function Models,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 2, pp. 183–194, May 2005.

[71] 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, no. 4, pp. 421–439, August 2006.

[72] W. Liu, Q. Zhang, E. P. K. Tsang, C. Liu, and B. Virginas, “On the Performance of Metamodel Assisted MOEA/D.” in *ISICA-2007*, ser. LNCS, L. Kang, Y. Liu, and S. Y. Zeng, Eds., vol. 4683. Wuhan, China: Springer, August 2007, pp. 547–557.

[73] J. Knowles, “ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, February 2006.

[74] T. Goel, R. Vaidyanathan, R. T. Haftka, W. Shyy, N. Queipo, and K. V. Tucker, "Response surface approximation of Pareto optimal front in multi-objective optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 4-6, pp. 879 – 893, January 2007.

[75] Q. Zhou and Y. Li, "Directed Variation in Evolution Strategies," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 356–361, August 2003.

[76] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 2, pp. 192–200, April 2005.

[77] S. Jun and K. Shigenobu, "Extrapolation-Directed Crossover for Real-coded GA: Overcoming Deceptive Phenomena by Extrapolative Search," in *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*. Seoul, Korea: IEEE Press, 27-30 May 2001, pp. 655–662.

[78] A. Zhou, Q. Zhang, Y. Jin, E. Tsang, and T. Okabe, "A Model-Based Evolutionary Algorithm for Bi-objective Optimization," in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 3. Edinburgh, Scotland: IEEE Press, September 2005, pp. 2568–2575.

[79] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, February 2008.

[80] J. Branke, T. Kaußler, and H. Schmeck, "Guiding Multi-Objective Evolutionary Algorithms Towards Interesting Regions," in *Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000), Poster Proceedings*, I. C. Parmee, Ed. Plymouth, UK: Plymouth Engineering Design Centre, University of Plymouth, 2000, pp. 1–4.

[81] J. Branke, T. Kaußler, and H. Schmeck, "Guidance in Evolutionary Multi-Objective Optimization," *Advances in Engineering Software*, vol. 32, pp. 499–507, 2001.

[82] V. Torczon and M. W. Trosset, “Using approximations to accelerate engineering design optimization,” Institute for Computer Applications in Science and Engineering, Technical Report TR-98-33, August 1998.

[83] L. Bull, “On model-based evolutionary computation,” *Soft Comput.*, vol. 3, no. 2, pp. 76–82, 1999.

[84] Y. Jin, “A Comprehensive Survey of Fitness Approximation in Evolutionary Computation,” *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 9, pp. 3–12, January 2005.

[85] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff, “A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation,” in *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*. London, England: ACM Press, 2007, pp. 1288–1295.

[86] M. A. El-Beltagy, P. B. Nair, and A. J. Keane, “Metamodeling Techniques for Evolutionary Optimization of Computationally Expensive Problems: Promises and Limitations,” in *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. San Francisco, CA: Morgan Kaufmann, 1999.

[87] M.-N. Terrasse, “A Metamodeling Approach to Evolution,” *Lecture Notes in Computer Science*, vol. 2065, p. 202, 2001.

[88] Y. Jin, M. Olhofer, and B. Sendhoff, “On evolutionary optimization with approximate fitness functions,” in *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2000, pp. 786–792.

[89] A. Ratle, “Accelerating the Convergence of Evolutionary Algorithms by Fitness Landscape Approximation,” in *Parallel Problem Solving from Nature - PPSN V*, ser. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer Berlin/Heidelberg, 1998, pp. 87–96.

[90] J. F. Rodrigues, J. E. Renaud, and L. T. Watsen, “Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data,” *Structural and Multidisciplinary Optimization*, vol. 15, no. 3-4, pp. 141–156, June 1998.

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

[92] M. K. Karakasis, A. P. Giotis, and K. C. Giannakoglou, “Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization,” *International Journal for Numerical Methods in Fluids*, vol. 43, no. 10-11, pp. 1149–1166, 2003.

[93] I. C. Kampolis, Giannakoglou, and K. C., “A multilevel approach to single- and multiobjective aerodynamic optimization,” *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33-40, pp. 2963–2975, June 2008.

[94] H. seog Chung and J. J. Alonso, “Using Gradients to Construct Response Surface Models for High-Dimensional Design Optimization Problems,” in *39th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 2001.

[95] M. Butz, P. Lanzi, and S. Wilson, “Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction,” *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 3, pp. 355–376, June 2008.

[96] K. Rasheed and H. Hirsh, “Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Las Vegas, Nevada, USA: Morgan Kaufmann, 10-12 2000, pp. 628–635.

[97] A. W. Iorio and X. Li, “Incorporating Directional Information within a Differential Evolution Algorithm for Multi-objective Optimization,” in *Proceedings of The 8th Annual ACM Genetic and Evolutionary Computation Conference (GECCO ’06)*, M. K.

et al., Ed., vol. 1. Seattle, Washington, USA: ACM Press, NY, July 2006, pp. 691–697.

[98] J. Moody and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[99] A. A. Mullur and A. Messac, "Metamodeling using extended radial basis functions: a comparative approach," *Engineering with Computers*, vol. 21, no. 3, pp. 203–217, 2006.

[100] Y. S. Ong, K. Y. Lum, and P. B. Nair, "Hybrid Evolutionary Algorithm with Hermite Radial Basis Function Interpolants for Computationally Expensive Adjoint Solvers," *Computational Optimization and Applications*, vol. 39, pp. 97–119, January 2008.

[101] A. Isaacs, T. Ray, and W. Smith, "An Evolutionary Algorithm with Spatially Distributed Surrogates for Multiobjective Optimization," in *ACAL*, ser. Lecture Notes in Computer Science, M. Randall, H. A. Abbass, and J. Wiles, Eds., vol. 4828. Springer, 2007, pp. 257–268.

[102] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, Mar 1990.

[103] E. Rios-Patron and R. Braatz, "On the "Identification and control of dynamical systems using neural networks"," *Neural Networks, IEEE Transactions on*, vol. 8, no. 2, pp. 452–, March 1997.

[104] H. Pierreval, "A Metamodeling Approach Based on Neural Networks," *Int. Journal in Computer Simulation*, vol. 6, no. 3, p. 365, 1996.

[105] W. Liu and S. M. Batill, "Gradient-enhanced neural network response surface approximations," Tech. Rep. AIAA-2000-4923, 2000.

[106] X. Han, “Approximate Interpolation by Neural Networks with the Inverse Multiquadric Functions,” in *ISICA*, ser. Lecture Notes in Computer Science, L. Kang, Y. Liu, and S. Y. Zeng, Eds., vol. 4683. Springer, 2007, pp. 296–304.

[107] J. Lee, H. Jeong, and S. Kang, “Derivative and GA-based methods in metamodeling of back-propagation neural networks for constrained approximate optimization,” *Structural and Multidisciplinary Optimization*, vol. 35, no. 1, pp. 29–40, 2007.

[108] W. Carpenter and J.-F. Barthelemy, “Common misconceptions about neural networks as approximators,” *ASCE Journal of Computing in Civil Engineering*, vol. 8, no. 3, pp. 345–358, 1994.

[109] ——, “A comparison of polynomial approximation and artificial neural nets as response surface,” AIAA, Tech. Rep. 92-2247, 1992.

[110] 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, 2005.

[111] P. K. Nain and K. Deb, “A Multi-Objective Optimization Procedure with Successive Approximate Models,” Kanpur Genetic Algorithm Laboratory (KanGAL), Indian Institute Of Technology, Kanpur, India, Tech. Rep. KanGAL Report Number 2005002, 2005.

[112] A. Gaspar-Cunha and A. Vieira, “A Multiobjective Evolutionary Algorithm Using Neural Networks to Approximate Fitness Evaluations,” *International Journal of Computers, Systems and Signals*, vol. 6, no. 1, pp. 18–36, 2005.

[113] S. Adra, A. Hamody, I. Griffin, and P. J. Fleming, “A Hybrid Multi-objective Evolutionary Algorithm Using an Inverse Neural Network for Aircraft Control System Design,” in *IEEE Congress on Evolutionary Computation (CEC-2005)*, vol. 1. Edinburgh, UK: IEEE Pres, September 2005, pp. 1–8.

[114] T. Ray and W. Smith, “A Surrogate Assisted Parallel Multiobjective Evolutionary Algorithm for Robust Engineering Design,” *Engineering Optimization*, vol. 38, no. 8, pp. 997–1011, 2006.

[115] M. Greig, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[116] J. Sacks, S. B. Schiller, and W. J. Welch, "Design of Computer Experiments," *Technometrics*, vol. 31, no. 1, pp. 41–47, February 1989.

[117] M. D. Morris, "The design and analysis of computer experiments. thomas j. santner , brian j. williams , and william i. notz," *Journal of the American Statistical Association*, vol. 99, no. 2, December 2004.

[118] J. P. Kleijnen, "Kriging metamodeling in simulation: A review," *European Journal of Operational Research*.

[119] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 15, no. 1, pp. 37 – 49, 2001.

[120] J. J. Alonso and H. seog Chung, "Using Gradients To Construct Cokriging Approximation Models for High-Dimensional Design Optimization Problems," in *40th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 2002.

[121] P. N. Koch, D. Mavris, and F. Mistree, "Multi-Level, Partitioned Response Surfaces for Modeling Complex Systems," March 1998.

[122] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems 8*. MIT press, 1996, pp. 514–520.

[123] D. J. C. Mackay, *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, June 2002.

[124] ——, "Gaussian Processes - A Replacement for Supervised Neural Networks?" Lecture notes for a tutorial at Advances in Neural Information Processing Systems (NIPS), 1997.

[125] A. J. Keane, A. Choudhury, and P. B. Nair, "A data parallel approach for large-scale gaussian process modeling," in *Proceedings of the Second SIAM International Conference on Data Mining*, Arlington, VA, 2002.

[126] C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3410. Springer, 2005.

[127] B. Wilson, D. Cappelleri, T. W. Simpson, and M. Frecker, "Efficient Pareto Frontier Exploration using Surrogate Approximations," *Optimization and Engineering*, vol. 2, no. 1, pp. 31–50, March 2001.

[128] P. Larraanaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[129] J. A. Lozano, P. Larraaga, I. Inza, and E. Bengoetxea, Eds., *Towards a New Evolutionary Computation*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2006, vol. 192.

[130] F. Lobo, M. Pelikan, M. Pelikan, and D. E. Goldberg, "A survey of optimization by building and using probabilistic models," Computational Optimization and Applications, Illinois Genetic Algorithm Laboratory, Tech. Rep., 1999.

[131] O. Takahashi and S. Kobayashi, "An Angular Distance Dependent Alternation Model for Real-Coded Genetic Algorithms," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 20-23 June 2004, pp. 2159–2165.

[132] I. Ono, S. Kobayashi, and K. Yoshida, "Optimal lens design by real-coded genetic algorithms using UNDX," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 483–497, 2000.

[133] K. Y. I. Ono, S. Kobayashi, "Optimal lens design by real-coded genetic algorithms using UNDX," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 483–497, October 2000.

[134] K. Deb, A. Anand, and D. Joshi, "A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization," *Evolutionary Computation*.

[135] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, October 2000.

[136] K. Deb and R. B. Agrawal, "Simulated Binary Crossover For Continuous Search Space," Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, Tech. Rep. IITK/ME/SMD-94027, November 1994.

[137] S. Tsutsui and D. Goldberg, "Simplex Crossover And Linkage Identification: Single-Stage Evolution vs. Multi-Stage Evolution," in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC 02)*. Honolulu, HI: IEEE Press, Piscataway, NJ, May 2002, pp. 974–979 vol. 1.

[138] K. S. Anderson and Y. Hsu, "Genetic crossover strategy using an approximation concept," in *Proceedings of The IEEE Congress On Evolutionary Computation 1999 (CEC - 1999)*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 1, Washington D.C., USA, June - September 1999, pp. –533 Vol. 1.

[139] P. D. Stroud, "Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations," *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 66–77, February 2001.

[140] S. Shan and G. Wang, "Space exploration and global optimization for computationally intensive design problems: a rough set based approach," *Structural and Multidisciplinary Optimization*, vol. 28, no. 15, pp. 427–441, December 2004.

[141] Z. Zhou, Y.-S. Ong, M.-H. Lim, and B.-S. Lee, "Memetic algorithm using multi-surrogates for computationally expensive optimization problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 10, pp. 957–971, 2007.

[142] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, December 2007.

[143] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research and Management Science. Kluwer Academic Publishers, Dordrecht, 1999, vol. 12.

[144] M. Gallagher, I. Wood, J. Keith, and G. Sofronov, "Bayesian inference in estimation of distribution algorithms," in *Proceedings of The IEEE Congress on Evolutionary Computation (CEC '07)*, September 2007, pp. 127–133.

[145] F. Guimaraes, F. Campelo, H. Igarashi, D. Lowther, and J. Ramirez, "Optimization of Cost Functions Using Evolutionary Algorithms With Local Learning and Local Search," *Magnetics, IEEE Transactions on*, vol. 43, no. 4, pp. 1641–1644, April 2007.

[146] C. Gomes da Silva, J. ao Clímaco, and J. Figueira, "A scatter search method for bi-criteria {0,1}-knapsack problems," *European Journal of Operational Research*, vol. 169, no. 2, pp. 373–391, March 2006.

[147] V. Aggarwal and U.-M. O'Reilly, "COSMO: A Correlation Sensitive Mutation Operator for Multi-Objective Optimization," in *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*. London, England: ACM Press, 2007, pp. 741–748.

[148] Y. Wang, Z. Cai, G. Guo, and Z. Zhou, "Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems," *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, vol. 37, no. 3, pp. 560–575, June 2007.

[149] L. Ljung, *System Identification: Theory for The User*. NJ, USA: Prentice-Hall Inc., 1999.

[150] L. Ljung and T. Glad, *Modelling of Dynamic Systems*. NJ, USA: Prentice-Hall Inc., 1994.

[151] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C (2nd ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992.

[152] D. E. Stewart and Z. Leyk, "Meschach: Matrix Computations in C," in *Centre for Mathematics and Its Applications*, vol. 32, Australian National University, Canberra, Australia, 1994, available at: http://www.netlib.org/c/meschach/ and http://www.math.uiowa.edu/~dstewart/meschach/.

[153] O. Schtze, S. Mostaghim, M. Dellnitz, and J. Teich, *Covering Pareto Sets by Multilevel Evolutionary Subdivision Techniques*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, January 2003, vol. 2632/2003, p. 10.

[154] K. Deb, L. Thiele, M. Laumanns, and E. Ziztler, "Scalable Test Problems for Evolutionary Multi-objective Optimization," India, Tech. Rep. KanGAL Report No 2001001, August 2001.

[155] S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit," *IEEE Transaction on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, October 2006.

[156] K. Deb, D. Joshi, and A. Anand, "Real-coded evolutionary algorithms with parent-centric recombination," vol. 1, May 2002, pp. 61–66.

[157] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[158] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, April 2003.

[159] J. Knowles, L. Thiele, and E. Zitzler, "A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, Tech. Rep. 214, February 2006, revised Version.

[160] C. M. Fonseca and P. J. Fleming, "On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers," in *Parallel Problem Solving from*

Nature–PPSN IV, ser. Lecture Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, September 1996, pp. 583 – 594.

[161] V. G. da Fonseca, C. M. Fonseca, and A. O. Hall, "Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function," in *First International Conference on Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, Eds. Springer-Verlag, 2001, pp. 213 – 225.

[162] W. Connover, *Practical Nonparametric Statistics*, 3rd ed. New York, NY: John Wiley and Sons, 1999.

[163] K. Deb, U. V. Rao, and S. Karthik, "Dynamic Multi-objective Optimization and Decision Making Using Modified NSGA-II: A Case Study on Hydro-Thermal Power Scheduling," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, May 2007, vol. 4403/2007, pp. 803–817.

[164] A. Zhou, Y. Jin, Q. Zhang, B. Sendoff, and E. Tsang, "Prediction-based Population Re-initialization for Evolutionary Dynamic Multi-objective Optimization," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, May 2007, vol. 4403/2007, pp. 832–846.

[165] S. Zeng, G. Chen, L. Zheng, H. Shi, H. de Garis, L. Ding, and L. Kang, "A Dynamic Multi-Objective Evolutionary Algorithm Based on an Orthogonal Design," in *Proceedings of the IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*. Vancouver, Canada: IEEE Press, Piscataway, NJ, July 2006, pp. 573–580.

[166] I. Hatzakis and D. Wallace, "Dynamic Multi-objective Optimization with Evolutionary Algorithms: A Forward-Looking Approach," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO 2006)*. Seattle, Washington, USA: ACM Press, NY, USA, July 2006, pp. 1201–1208.

[167] E. Khor, K. Tan, and T. Lee, “Learning the Search Range for Evolutionary Optimization in Dynamic Environments,” *Knowledge and Information Systems*, vol. 4, no. 2, pp. 228–255, April 2002.

[168] X. Li, J. Branke, and M. Kirley, “On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems,” in *Proceedings of The IEEE Congress on Evolutionary Computation 2007 (CEC 2007)*. Singapore: IEEE Press, Piscataway, NJ, September 2007, pp. 1635–1643.

[169] J. Teich, “Pareto-front Exploration with Uncertain Objectives,” in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-1993)*, vol. 1993. Zurich, Switzerland: Springer, March 2001, pp. 314–328.

[170] H. G. Cobb, “An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments,” Naval Research Lab, Washington, D.C., Tech. Rep. 6760 (NLR Memorandum), 1990.

[171] E. J. Hughes, “Evolutionary Multi-objective Ranking with Uncertainty and Noise,” in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-1993)*, vol. 1993. Zurich, Switzerland: Springer, March 2001, pp. 329–343.

[172] J. Branke, “Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC - 1999)*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3. Washington D.C., USA: IEEE Press, June-September 1999, pp. 1875–1882.

[173] C. Goh and K. Tan, “An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 354–381, June 2007.

[174] S. Yang, “Explicit Memory Schemes for Evolutionary Algorithms in Dynamic Environments,” in *Evolutionary Computation in Dynamic and Uncertain Environments*, ser.

Studies in Computational Intelligence, S. Yang, Y.-S. Ong, and Y. Jin, Eds. Springer Berlin/Heidelberg, April 2007, vol. 51/2007, ch. Optimum Tracking in Dynamic Environments, pp. 3–28.

[175] J. Branke, *Evolutionary Optimization in Dynamic Environments*, ser. Genetic Algorithms and Evolutionary Computation. Boston, MA: Kluwer Academic Publishers, 2002, vol. 3.

[176] Y. Jin and B. Sendoff, “Constructing Dynamic Optimization Test Problems Using the Multi-objective Optimization Concept,” in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, March 2004, vol. 3005/2004, pp. 525–536.

[177] M. Farina, K. Deb, and P. Amato, “Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, October 2004.

[178] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. NY, USA: Macmillan Publishing Company, 1992.

[179] S. E. Eklund, “A massively parallel architecture for distributed genetic algorithms,” *Parallel Computing*, vol. 30, no. 5-6, pp. 647 – 676, 2004.

[180] W. Kus, “Grid-enabled evolutionary algorithm application in the mechanical optimization problems,” *Engineering Applications of Artificial Intelligence*, vol. 20, no. 5, pp. 629 – 636, 8 2007.

[181] H.-K. Ng, D. Lim, Y.-S. Ong, B.-S. Lee, L. Freund, S. Parvez, and B. Sendhoff, *Advances in Natural Computation*. Springer Berlin/Heidelberg, 2005, ch. A Multi-cluster Grid Enabled Evolution Framework for Aerodynamic Airfoil Design Optimization, pp. 1112 – 1121.

[182] J. Herrera, E. Huedo, R. S.Montero, and I. M. Llorente, “A Grid-Oriented Genetic Algorithm,” in *Advances in Grid Computing - EGC 2005, European Grid Conference, 2005, Revised Selected Papers*, ser. Lecture Notes in Computer Science (LNCS),

P. M. A. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, Eds., vol. 3470. Amsterdam, The Netherlands: Springer, February 14-16 2005, pp. 315–322.

[183] R. G. Regis and C. A. Shoemaker, "Parallel radial basis function methods for the global optimization of expensive functions," *European Journal of Operational Research*, vol. 182, no. 2, pp. 514–535, October 2007.

[184] *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, California: Morgan Kaufmann Publishers, 1998.

[185] D. Abramson, T. Peachey, and A. Lewis, *Computational Science ICCS 2006*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2006, ch. Model Optimization and Parameter Estimation with Nimrod/O, pp. 720 – 727.

[186] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid," *High-Performance Computing in the Asia-Pacific Region, International Conference on*, vol. 1, p. 283, 2000.

[187] A. Liefooghe, M. Basseur, L. Jourdan, and E.-G. Talbi, *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2007, ch. ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization, pp. 386 – 400.

[188] M. G. Arenas, P. Collet, A. E. Eiben, M. Jelasity, J. J. Merelo, M. Preu, and M. Schoenauer, "A framework for distributed evolutionary algorithms," in *In Proceedings of PPSN 2002*. Springer-Verlag, 2002, pp. 665–675.

[189] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms. (Cover story)," *Nature*, vol. 406, no. 6799, pp. p974 –, August 2000.

[190] L. Yaeger, "Computational genetics, physiology, metabolism, neural systems, learning, vision and behavior or polyworld: Life in a new context," in *proceedings of the Artificial Life III*, C. Langton, Ed., vol. XVII. Santa Fe Institute Studies in the Sciences of Complexity: Addison-Wesley, 1994, pp. 263–298.

[191] J. Mehnen, T. Michelitsch, K. Schmitt, and T. Kohlen, "pMOHypEA: Parallel Evolutionary Multiobjective Optimization using Hypergraphs," Reihe Computational Intelligence Collaborative Research Center 531, University of Dortmund, Dept. of Computer Science, Dortmund, Germany, Tech. Rep. CI-189/04, December 2004.

[192] M. Mezmaz, N. Melab, and E.-G. Talbi, "Towards a Coordination Model for Parallel Cooperative P2P Multi-objective Optimization," in *Advances in Grid Computing - EGC 2005*, 2005, pp. 305–314.

[193] F. Streichert, H. Ulmer, and A. Zell, "Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms," in *EMO*, ser. Lecture Notes in Computer Science, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410. Springer, 2005, pp. 92–107.

[194] S. Gustafson and E. K. Burke, "The speciating island model: An alternative parallel evolutionary algorithm," *Journal of Parallel and Distributed Computing*, vol. 66, no. 8, pp. 1025 – 1036, August 2006.

[195] N. Xiao and M. P. Armstrong, "A Specialized Island Model and Its Application in Multiobjective Optimization," in *GECCO*, 2003, pp. 1530–1540.

[196] M. Kirley and R. Stewart, "An Analysis of The Effects of Population Structure on Scalable Multiobjective Optimization Problems," in *The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007)*. London, England: ACM Press, 2007, pp. 845–852.

[197] X. Li and M. Kirley, "The effects of varying population density in a fine-grained parallel genetic algorithm," *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, pp. 1709–1714, 2002.

[198] M. Kirley, "MEA: a metapopulation evolutionary algorithm for multi-objective optimisation problems," in *Evolutionary Computation, 2001.Proceedings of the 2001 Congress on*, vol. 2, 2001, pp. 949–956.

[199] E. Alba and B. Dorronsoro, “The exploration/exploitation tradeoff in dynamic cellular genetic algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 2, pp. 126–142, April 2005.

[200] M. Giacobini, M. Tomassini, A. G. B. Tettamanzi, and E. Alba, “Selection intensity in cellular evolutionary algorithms for regular lattices,” *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 5, pp. 489–505, October 2005.

[201] M. Kirley, “A Cellular Genetic Algorithm with Disturbances: Optimisation Using Dynamic Spatial Interactions,” *Journal of Heuristics*, vol. 8, no. 3, pp. 321 – 342, May 2002.

[202] K. Deb, M. Mohan, and S. Mishra, “A Fast Multi-Objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions,” Kanpur Genetic Algorithm Laboratory (KanGAL), Indian Institute of Technology, Kanpur, PIN 208016, India, Tech. Rep. 2003002, 2003.

[203] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, and R. Buyya, “Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications,” in *E-SCIENCE ’07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 151–159.

[204] B. Hayes, “Cloud computing,” *Commun. ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[205] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in *OSDI’04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[206] F. Luna, A. J. Nebro, and E. Alba, “Observations in using grid-enabled technologies for solving multi-objective optimization problems,” *Parallel Computing*, vol. 32, no. 5-6, pp. 377 – 393, June 2006.

[207] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997.

[208] C. Germain, V. Nri, G. Fedak, and F. Cappello, "XtremWeb: Building an Experimental Platform for Global Computing," in *in Grid Computing GRID 2000*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2000, pp. 107 – 129.

[209] D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B.-S. Lee, "Efficient Hierarchical Parallel Genetic Algorithms using Grid Computing," *Future Generation Computer Systems*, vol. 23, no. 4, pp. 658–670, May 2007.