DYNAMIC MULTIOBJECTIVE OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

ARRCHANA MURUGANANTHAM

NATIONAL UNIVERSITY OF SINGAPORE

2017

DYNAMIC MULTIOBJECTIVE OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

ARRCHANA MURUGANANTHAM

(*B.Eng. (Hons), NUS*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2017

Supervisors:

Professor Tan Kay Chen

Associate Professor Prahlad Vadakkepat

Examiners:

Associate Professor Abdullah Al Mamun

Dr Lin Feng

Professor Zhang Mengjie, Victoria University of Wellington

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Arrchana Muruganantham

May 3, 2017

Name	:	Arrchana Muruganantham
Degree	:	Doctor of Philosophy
Supervisor(s)	:	Professor Tan Kay Chen, Associate Professor Prahlad Vadakkepat
Department	:	Department of Electrical & Computer Engineering
Thesis Title	:	Dynamic Multiobjective Optimization using Evolutionary Algorithms

Abstract

This thesis focuses on solving Dynamic Multiobjective Optimization Problems(DMOPs) using Evolutionary Algorithms in unconstrained and constrained environments.

Multiobjective Optimization involves the optimization of two or more conflicting objectives simultaneously. There is no single solution to such problems, but multiple trade-off solutions. Evolutionary Algorithms are a good candidate to solve such problems as they can obtain multiple solutions in a single run. When the optimal solutions change with time, it results in a Dynamic Multiobjective Optimization problem. Many real-world problems involve multiple objectives which maybe conflicting, are dynamic in nature and affected by constraints. In this thesis, the issues of dynamicity and presence of constraints are addressed by providing some possible solutions.

Evolutionary Algorithms have shown remarkable performance in solving static Multiobjective Optimization Problems. In general, they take

significant time to converge, which becomes an impediment in dynamic environments. The optimization algorithm must be able to track the moving optima efficiently. A prediction model can learn the patterns from past experience and predict future changes. To address dynamicity of problems, prediction techniques are implemented to work in tandem with Evolutionary Algorithms.

Firstly, a Dynamic Multiobjective Evolutionary Algorithm based on MOEA/D-DE (Multiobjective Evolutionary Algorithm based on Decomposition with Differential Evolution) using Kalman Filter predictions in decision space is proposed to solve DMOPs. The predictions help to guide the search towards the changed optima, thereby accelerating convergence. A scoring scheme is devised to hybridize the Kalman Filter prediction with a random reinitialization method. The proposed algorithm is tested on a number of benchmark problems and it shows significantly improved performances over a number of test benchmark problems. The Kalman Filter based prediction mechanism does not require any learning time and provides predictions of the changing Pareto Optimal Solutions reasonably well right from the start of the run.

MOEA/D-DE assisted by a non-linear prediction method using Support Vector Regression predictions is also proposed to solve DMOPs. Support Vector Machines have traditionally been used in the context of classification and regression. In this method, a time series is formed by the near-optimal solutions obtained by the Evolutionary Algorithm during previous changes. Support Vector Machines, which are data-driven are used in tandem with the Evolutionary Algorithm to predict new solutions for future generations from the time series, when a change in the environment is detected. Results of testing the proposed algorithm on several benchmark problems show that the Support Vector assisted MOEA/D-DE performs significantly better

than the other algorithms in the more complicated problems as Support Vector Regression does not make any assumptions about the underlying process or structure of the changing Pareto Optimal Fronts.

Dynamic Multiobjective Optimization in constrained environments has not been explored much in the literature. Only a handful of algorithms have been proposed to solve constrained dynamic optimization problems. However, a number of strategies have been proposed for constraint handling alone (in single objective optimization or static multiobjective optimization). To address the second issue of handling constraints, the Kalman Filter based prediction mechanism is combined with an adaptive threshold based constraint handling method to ensure solution feasibility while simultaneously tracking the time varying solutions. An existing static constrained Multiobjective Optimization benchmark problem set is modified to incorporate dynamicity. The proposed algorithm is tested on these benchmark problems and the performance compared against Dynamic NSGA-II algorithm variants is encouraging as the proposed algorithm performs significantly better in many of the problems. Further, the proposed algorithms are more robust and tend to perform even better with increasing problem difficulty.

Keywords : dynamic, evolutionary multi-objective optimization, prediction, kalman filter, support vector machines, constraint handling

Acknowledgment

My PhD journey would not have been possible without the support and guidance of a multitude of people. I would like to thank:

My Supervisor, Associate Professor Tan Kay Chen, for his unwavering support and guidance throughout these years.

My Co-Supervisor, Associate Professor Prahlad Vadakkepat, for his philosophical guidance, not just in research, but in life in general as well, which has made me a much better person.

ECE Dept and NUS, for providing me with the NUS Research Scholarship, without which this journey would not have been possible.

My lab colleagues, Willson, Senbong, Hu Jun, Yu Qiang, Bharath and Arun, who were my guiding seniors in this PhD journey. This journey would not have been as eventful and fun without my partner-in-crime in the lab, Qiu Xin. Lim Pin, Zhang Chong, Sim Kuan, Ruoxu and Stella for their help in various circumstances. Sivam and Raj, helped me cope with the stress of the last stretch in this journey with their fun-filled personalities. I would also like to thank lab officers, Ms. Sara and Mr. Zhang Hengwei for their continued assistance in various tasks.

My friends, Arun, Divya, Kritika, Vanchi, Sunethra, and Shweta who have lent their shoulders for support whenever I needed them without any hesitation. Lalitha Aunty and Raghu Uncle who have been a family away from home in Singapore.

My family - Dad, Mom and brother, Vishnu for believing in me always and immensely supporting me in all my endeavors.

May 3, 2017

Publications Resulting from this Thesis

Journal Papers

1. A. Muruganantham, K. C. Tan and P. Vadakkepat, "Evolutionary Dynamic Multiobjective Optimization Via Kalman Filter Prediction," in IEEE Transactions on Cybernetics, vol. 46, no. 12, pp. 2862-2873, Dec. 2016.
2. A. Muruganantham, K. C. Tan and Vadakkepat, P. Data-driven Accelerated Convergence in Evolutionary Dynamic Multiobjective Optimization, *IEEE Transactions on Cybernetics*(Under Review), 2016
3. A. Muruganantham, K. C. Tan and Vadakkepat, P. Constraint Handling Approaches for Dynamic Multiobjective Optimization Problems with Constraints (Under Preparation)

Conference Papers

1. A. Muruganantham, Yang Zhao, Sen Bong Gee, Xin Qiu, Kay Chen Tan. Dynamic Multiobjective Optimization Using Evolutionary Algorithm with Kalman Filter, *Procedia Computer Science*, Volume 24, 2013
2. A. Muruganantham, Tan, K. C. and Vadakkepat, P. Solving the IEEE CEC 2015 Dynamic Benchmark Problems Using Kalman Filter Based Dynamic Multiobjective Evolutionary Algorithm, *Proceedings of the 19th Asia Pacific Symposium on Intelligent and Evolutionary Systems 2015*, November 2015

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

$\mathbf{f}, \mathbf{f}(\cdot)$	Objective vector or functions
$\mathbf{x}, \mathbf{y}$	Decision vector
$\mathbf{z}^*$	Ideal vector
F	Objective space
Ω	Decision space
$g(\cdot), h(\cdot)$	Inequality, equality constraint function
m	Number of objective functions
n	Number of decision variables
p, q	Number of inequality, equality contraints
$[\cdot]^T, [\cdot]^{-1}$	Transpose, inverse operation
$\mathbb{E}[\cdot]$	Expectation operation
$\mathbb{R}$	Real space
$\mathcal{N}$	Normal distribution
$\prec, \preceq, \sim$	Strongly, weakly dominates and incomparable
$\mathcal{I}(\cdot)$	Performance indicator function

List of Abbreviations

MOP Multi-objective Optimization Problem

SOP Single-objective Optimization Problem

DMOP Dynamic Multi-objective Optimization Problem

EA Evolutionary Algorithm

GA Genetic Algorithm

EDO Evolutionary Dynamic Optimization

EMO Evolutionary Multi-objective Optimization

EDMO Evolutionary Dynamic Multi-objective Optimization

MOEA Multi-Objective Evolutionary Algorithm

DMOEA Dynamic Multi-Objective Evolutionary Algorithm

POF Pareto Optimal Front

POS Pareto Optimal Set

EDA Estimation of Distribution Algorithm

MOEA/D Decomposition-based Multi-Objective Evolutionary Algorithm

MOEA/D-DE Decomposition-based Multi-Objective Evolutionary Algorithm with Differential Evolution

NSGA-II Non-dominated Sorting Genetic Algorithm II

dCOEA Dynamic Competitive-Cooperative Coevolutionary Algorithm

D-QMOO Dynamic Queuing Multi-Objective Optimizer

DOMOEA Dynamic Orthogonal Multi-objective Evolutionary Algorithm

SBX Simulated Binary Crossover

DE Differential Evolution

GD Generational Distance

IGD Inverted Generational Distance

HD Hausdorff Distance

HPV Hypervolume

GS Generalized Spread

FDA Farina-Deb-Amato dynamic multi-objective benchmark

CDT Change Detection Test

RMSE Root Mean Square Error

MO Multiobjective

MOO Multiobjective Optimization

DMO Dynamic Multiobjective Optimization

SOO Single-Objective Optimization

RND Random Reinitialization

MOEA/D-KF Dynamic Multiobjective Evolutionary Algorithm based on Kalman Filter Prediction

HYP Hypermutation

PPS Population Prediction Strategy

RM-MEDA Regularity Model-based Multiobjective Estimation of Distribution Algorithm

DNSGA-II-A Dynamic NSGA-II with 20% random reinitialization

DNSGA-II-B Dynamic NSGA-II with 20% hypermutation

SVM Support Vector Machines

SVR Support Vector Regression

MOEA/D-SVR Dynamic Multiobjective Evolutionary Algorithm based on Support Vector Regression Prediction

DCMOP Dynamic Constrained Multiobjective Optimization Problem

CMOEA Constrained Multiobjective Evolutionary Algorithm

DCMOEA Dynamic Constrained Multiobjective Evolutionary Algorithm

GENOCOP Genetic Algorithm for Numerical Optimization of Constrained Problems

ENORA Evolutionary Algorithm of Nondominated Sorting with Radial Slots

CMOEA/D-RND Constraint Handling embedded MOEA/D with Random Reinitialization

CMOEA/D-HYP Constraint Handling embedded MOEA/D with Hypermutation

CMOEA/D-KF Constraint Handling embedded MOEA/D with 3by3 Kalman Filter

CMOEA/D-KFSC Constraint Handling embedded MOEA/D with 3by3 Kalman Filter using Scoring Scheme

Chapter 1

Introduction

Optimization problems are aplenty and are found in various fields such as science, engineering, economics, finance, management, scheduling, planning, design, control, etc. The list is ever growing, and scientists and industrialists alike are in the lookout for better and more efficient techniques to solve their problems. Optimization in general refers to the process of finding one or more feasible solutions which correspond to extreme values of one or more objectives. Many researchers have tend to focus on optimization problems which consider a single objective, although most real-world search and optimization problems involve more than one objective. Further, the presence of conflict in the multiple objectives makes these optimization problems (commonly termed as Multiobjective Optimization (MOO) problems) more interesting and challenging to solve. Since no single solution can satisfy the multiple conflicting objectives simultaneously, the solution to a MOO problem is a set of trade-off optimal solutions. Classical optimization methods such as hill climbing, simulated annealing can at best find one solution in a simulation run, thereby deeming these methods inefficient to solve MOO problems.

Evolutionary algorithms are inspired from biological evolution and

mimic nature's evolutionary principles to drive the search towards optimal solution(s). These algorithms use a population of solutions in each iteration, consequently making them ideal candidates for solving MOO problems. Numerous Evolutionary Algorithms(EAs) have been developed in the past few decades to solve MOO problems such as NSGA-II, MOEA/D, MOEA/D-DE, to name a few. The advances of Evolutionary Multiobjective Optimization research has been drastic and has resulted in many new paradigms to be developed such as the Estimation of Distribution Algorithms(EDAs), decomposition based algorithms, and so on. However, there has only been lukewarm interest in applying Evolutionary Algorithms to solve dynamic optimization problems, where the optimum(or optima) changes with time. Furthermore, most of the EA researchers in this area have tend to focus on dynamic Single-Objective Optimization (SOO) problems, while most real-world problems are Dynamic Multiobjective Optimization (DMO) problems.

Using Evolutionary Algorithms to solve DMO problems has started gaining attention over the past few years. Nevertheless, there is large scope for contribution and improvement in this field. In DMO problems the fitness landscape is changing over time. Preliminary research in solving proposed benchmark problems involved applying Multiobjective (MO) Evolutionary Algorithm(MOEA) directly to solve them. However, the inherent characteristic of an MOEA is that it takes significant amount of time to converge to the Pareto Optimal Front(POF). This is an important issue in DMO where the POF and/or the Pareto Optimal Solution(s) (POS) are continuously changing with time. In the current literature, various approaches have been proposed to solve DMO problems.

1.1 Basic Definitions

This section provides the basic definitions used in the evolutionary MO community together with some key concepts which are essential for understanding the work described in a scientific manner.

1.1.1 Multiobjective Optimization Problem

A MO problem can be expressed in its general form mathematically as

$$\begin{aligned} \text{Minimize/Maximize } & f_m(x), && m = 1, 2, \ldots, M; \\ \text{subject to } & g_j(x) \geq 0, && j = 1, 2, \ldots, J; \\ & h_k(x) = 0, && k = 1, 2, \ldots, K; \\ & x_i^L \leq x_i \leq x_i^U, && i = 1, 2, \ldots, n. \end{aligned}$$

where f_i is the i-th objective function and M is the number of objectives. $f(x) = [f_1(x)f_2(x)...f_m(x)]^T$ forms the objective vector, $f(x) \in \mathbb{R}^M$. A solution x is a vector of n decision variables: $x = [x_1x_2...x_n]^T$. The above general problem is associated with J inequality and K equality constraints. The last set of constraints are called *variable bounds*, restricting each decision variable x_i to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound. These variable bounds constitute the *decision variable space* $\Omega \in \mathbb{R}^n$, or simply the decision space.

In the presence of constraints g_j and h_k, the entire decision variable space Ω may not be feasible. The feasible region S is the set of all feasible solutions in the context of optimization. The feasible search space can be divided into 2 sets of solutions - pareto optimal and non pareto otpimal set. To define pareto optimality, first we need to look into the concept of domination.

1.1.2 Concept of Domination

There are M objective functions in a MO problem. Say, we have 2 solutions, i and j. $i < j$ implies i is better than j *or* i dominates j. A solution x^1 is said to dominate another solution x^2, if both the following conditions are true.

1. The solution x^1 is as good as x^2 in all objectives, *i.e.* $f_m(x^1) \leq f_m(x^2)$ for all $m = 1, 2, ..., M$, assuming a minimization problem.

2. The solution x^1 is strictly better than x^2 in at least one objective, *i.e.* $f_m(x^1) < f_m(x^2)$ in atleast one objective.

1.1.3 Pareto Optimality

Among a set of solutions P, the non-dominated solutions, P^* are those that are not dominated by any member of the set P. When the set P comprises the entire search space, the resulting non-dominated set P^* is the *Pareto Optimal Set*(POS in the decision space). Pareto optimal solutions joined together as a curve form the *Pareto Optimal Front*(POF in the objective space). The front lies in the bottom-left corner of the search space for problems where all objectives are to be minimized.

1.1.4 Goals of an MOEA

The working principle for an ideal MO procedure consists of finding multiple trade-off optimal solutions with a wide range of values for the objectives, and later choosing one of the obtained solutions using higher level information. In such a case it is difficult to prefer one solution over the other without any further information about the problem. If higher level information is satisfactorily available, this can be used to make a biased search. However,

in the absence of any such information, all pareto optimal solutions are equally important. Therefore, there are 2 goals:

1. To find a set of solutions as close as possible to the POF, i.e. *Convergence*

2. To find a set of solutions as diverse as possible, i.e. *Diversity*

For each of the M conflicting objectives, there exists one different optimal solution. An objective vector constructed with these individual optimal objective values constitutes the ideal objective vector, z^*, which in general lies in the infeasible space. For more detailed discussion of the concepts on MOO , please refer to [1].

1.1.5 Dynamic Multiobjective Optimization Problem

The various concepts discussed for MO are still essential in DMO together with some additional issues and goal(s). In general, in a DMO problem(DMOP), the optimum changes with time. Mathematically, a DMOP can be described as

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{f}(\mathbf{x},t) = [f_1(\mathbf{x},t)\ f_2(\mathbf{x},t)\ \ldots\ f_m(\mathbf{x},t)]^T \\ \text{subject to} \quad & \mathbf{x} \in \Omega \end{aligned} \tag{1.1}$$

where t represents time index, $x \in \mathbb{R}^n$ represents the decision vector, n is the number of decision variables and $\Omega \subset \mathbb{R}^n$ represents the decision space. m is the number of objectives, $\mathbb{R}^m$ is the objective space and $f(\mathbf{x},t)$ consists of m real-valued objective functions, each of which is continuous with respect to x over Ω. Thus, the *POF* and/or *POS* may change over time.

[2] have classified DMO problems based on the possible ways a problem can demonstrate a time varying change.

Type I POS changes, but POF does not change

Type II Both POS and POF change

Type III POS does not change, POF changes

Type IV Both POS and POF do not change, although the problem can change

These four cases are summarized in the Table 1.1. There are other possible ways of classifying DMOPs as well such as based on severity of change, predictability, etc [3].

Table 1.1: Four Different Types of DMOP

	POS	
POF	No Change	Change
No Change	Type IV	Type I
Change	Type III	Type II

1.2 Goals and Scope of the Thesis

Evolutionary Multiobjective Optimization has been a very active research field and numerous works have shown excellent results in solving static Multiobjective Optimization Problems. Even dynamic single objective optimization has been relatively well explored. However, the challenging area of Evolutionary Dynamic Multiobjective Optimization has been attempted only sparsely.

Summary of the limitations of existing work in the literature are enumerated as follows:

- A number of different approaches have been proposed to tackle dynamic single objective optimization problems. Evolutionary Algorithms have strongly established their strength in solving static Multiobjective Optimization problems. However, they take significant

amount of time to converge to the optima. This becomes a very important issue in Multiobjective Optimization problems in dynamic environments, as the continuous tracking of the time-varying optima becomes a crucial goal in addition to the MO objectives of convergence and diversity. Therefore, the Evolutionary Algorithms need to be strengthened in order to attain fast convergence to enable continuous tracking.

- Prediction based approaches have a good potential to contribute in this context as they can learn from the movement of the time-varying optima and work in tandem with the Evolutionary Algorithms to predict the subsequent location of the pareto optimal solutions. This is only recently beginning to gain traction in the Evolutionary Dynamic Multiobjective Optimization literature.

- The few prediction based algorithms seen in the literature require significant amount of time to learn the patterns exhibited in the changing optima before they can start predicting for future changes.

- Further, nonlinear prediction mechanisms have not been explored in the literature to solve Dynamic Multiobjective Optimization problems using Evolutionary Algorithms. Many real-world problems inherently have non-linear characteristics and employing only linear prediction methods may not be fully fruitful.

- Multiobjective Optimization in the presence of constraints has been well explored in static environments. However, only unconstrained or boundary constrained (limits on the decision variables range) have been considered widely and Dynamic Multiobjective Optimization in the presence of constraints has not been explored well.

This thesis aims to bridge the research gap in DMO by exploring the aforementioned issues and proposing possible solutions, as follows.

- Firstly, a Dynamic Multiobjective Optimization Evolutionary Algorithm assisted by an efficient linear prediction method should be explored. It is important to look at using as little time for training/learning the patters before the DMOEA can begin to provide reasonably good results, as this would pose a significant advantage compared to the linear prediction methods available in the literature.

- Notwithstanding the linear prediction methods usually considered, a non-linear prediction method should also be explored. A number of non-linear prediction methods are generally available. However, important conditions such as less number of free parameters to tune, fast convergence, etc need to be carefully considered in the selection of the method to be employed.

- Constraints in dynamic environments pose a significant challenge, as they add a further goal of finding feasible solutions apart from the goals of convergence, diversity and tracking of the time-varying optima in Dynamic Multiobjective Optimization. An adaptive constraint handling method which can be easily combined with the proposed dynamic optimization techniques is a possible solution to be evaluated to address the various goals of DMO in constrained environments using Evolutionary Algorithms.

1.3 Major Contributions

The major contributions of this thesis are listed as follows:

1. A linear Kalman Filter based prediction method is combined with

MOEA/D-DE (Multiobjective Evolutionary Algorithm based on Decomposition with Differential Evolution) to predict the location of subsequent optima in the decision variable space of time-varying Dynamic Multiobjective Optimization problems. The Kalman Filter operates in real-time and does not require any learning time, which results in a significant advantage when combined with the Evolutionary Algorithm. A scoring scheme mechanism has been proposed to hybridize the Kalman Filter prediction mechanism with a random reinitialization method. Performance comparison with the current state-of-the-art algorithms shows that the proposed algorithm predicts the time-varying solutions in the decision space efficiently right from the beginning while the second best algorithm, *Population Prediction Strategy*, takes a significant amount of training time before providing reasonable results. While the proposed algorithm's performance may not be superior when the linear dynamical assumption is violated, the Kalman Filter does not make any assumptions about the shape of the pareto fronts, unlike the existing methods and could therefore be considered a more generalized solution for Evolutionary Dynamic Multiobjective Optimization.

2. A non-linear prediction method using Support Vector Machines is explored. A time series formed by near-optimal solutions obtained by the Evolutionary Algorithm during previous changes is formulated as training data. Algorithm design encompasses data formulation, preprocessing and selection of parameters of the Support Vector Machines. Support Vector Regression, a data-driven method, learns from the training data to provide predictions for subsequent changes. The Support Vector Regression based algorithm performs comparably with the Kalman Filter assisted DMOEA in some of the problems.

Nevertheless, the Support Vector Regression based prediction model does not make any assumptions about the underlying process (linear dynamical or otherwise) or similarity in shape of consecutive Pareto Optimal Fronts, thereby leading to its better performance in more complicated problems with sharp and irregular change environments. The visualization of parameter selection for the Support Vector Regression model showed interesting patterns which may prove useful to get insights on problems with unknown pareto optimal characteristics or linkages between decision variables.

3. In Constrained Dynamic Multiobjective Optimization, feasibility of solutions also need to be ensured while simultaneously tracking the time-varying optima. An adaptive threshold based constraint handling mechanism is combined with dynamic optimization techniques such as the Kalman Filter prediction method, random reinitialization and hypermutation to solve these problems. The performance of the proposed algorithms are quite encouraging in the performance comparison results. While there is scope for improvement in terms of increasing the selection pressure, better diversity maintenance especially for problems with highly disconnected pareto optimal fronts, the proposed algorithms tend to perform better in most of the problems with both increasing frequency and severity of change.

1.4 Organization

The rest of the thesis is organized as follows:

Chapter 2 provides some background in Evolutionary Computation in general, and Dynamic Multiobjective Optimization in particular. The chapter also reviews some existing state-of-the-art in Dynamic Optimization,

both single and multiobjective.

Chapter 3 gives a brief introduction on the background of the Kalman Filter. The algorithm design involving the Kalman Filter prediction technique, the manner in which it is combined with MOEA/D-DE and the scoring scheme is described in detail. This chapter also covers the empirical study entailing performance comparison of the proposed algorithm with existing state-of-the-art algorithms.

Chapter 4 outlines related work in which Support Vector Regression is used to predict for time-series. A brief introduction on Support Vector Regression is also provided. The algorithm design comprising data formulation, data preprocessing and parameter selection of the SVM are provided in detail. The proposed algorithm is compared with the linear Kalman Filter prediction technique as well as random reinitialization.

Chapter 5 gives a detailed review of various constraint handling methods in both single and multiobjective optimization. It highlights the lack of contributions in dynamic constrained multiobjective optimization. An adaptive threshold based constraint handling mechanism is combined with dynamic optimization techniques to evaluate their performance in DCMOPs. Comparative studies are performed to show the improvements provided by the proposed algorithms.

Chapter 6 concludes this thesis and possible directions for future research are also discussed.

Chapter 2

Background & Literature Review

Despite the widespread victory of research on evolutionary multiobjective optimization, there is a growing need to apply evolutionary algorithms on DMO problems in the past two decades only. Efficient algorithms, benchmark problems, as well as appropriate performance metrics are needed to further the research in this field. Some preliminary research includes applying static MOEA directly to solve dynamic problems [4]. However, in a DMO problem, the fitness landscape is changing over time. Due to the inherent characteristics of evolutionary algorithm, MOEA generally takes a significant amount of time to converge to POF. Thus, explicit strategies are required to solve these time varying optimization problems efficiently.

In stationary single-objective optimization, the goal is to find the optimum as quickly as possible. In the case of MOO , diversity also becomes important. In DMO , the goal also encompasses tracking the changing optimum apart from convergence and diversity. The general assumption is that the problem after a change is in some way related to the problem before the change, and therefore it would be sensible for

an optimization algorithm to learn as much from the past experience to advance the search more effectively in solving the future problems after a change. Many methods have been proposed and investigated in the past, not restricted to evolutionary DMO . The main approaches suggested include

1. diversity introduction after a change
2. diversity maintenance throughout the run
3. memory approaches
4. prediction approaches
5. self-adaptive methods
6. multi-population approaches

2.1 Diversity Introduction

While convergence is sought after in static optimization, it can be detrimental in DMO. Intuitively, a simple solution is to increase the diversity in the population of an EA after a change so that the newly introduced solutions can aim to discover regions of the search space where the new optimum might be which was not available to the EA previously. Pioneering studies employing this strategy are hyper-mutation [5] and variable local search (VLS) [6]. Hyper-mutation is an adaptive mutation operator whose mutation rate is a product of the normal mutation rate and a hyper-mutation factor, which is invoked after a change in the problem is detected. In the VLS algorithm, as the name suggests, the mutation size is determined by a variable local search range. Hyper-mutation, though is one of the oldest methods, it is still traditionally used in many of the recent algorithms [7].

2.2 Diversity Maintenance

While diversity introduction has its advantages of maintaining the focus on search and need not waste their efforts on maintaining diversity throughout the time, it has some weaknesses resulting from dependence on change detection, difficulty in identifying the correct mutation size and little retention of information from previous search experience. Using diversity maintenance strategy, the algorithms continuously maintain population diversity throughout the search process to avoid the whole population converging to one place, and hence unable to track either the moving optimum or detect a new competing optimum. In this strategy, the algorithms need not detect the change in the problem explicitly as they rely on their diversity to cope with the changes adaptively. Classic example of this approach are Random Immigrants [8], fitness sharing [9], Thermo-Dynamical GA [10], Population-Based Incremental Learning [11]. In the Random Immigrants method, in every generation a number of randomly generated individuals are added to the population to maintain diversity. Cobb [12] has combined the usage of hyper-mutation and Random Immigrants leading to the development of a single mechanism that can work well in both stationary and nonstationary environments. Though the performance of algorithms following this approach is better than with the EA alone, continuously focusing on diversity maintenance can slow down, or even distract the optimization performance and may also perform poorly in problems where the changes are small, as totally stochastic individuals are introduced without making use of any historical information.

2.3 Memory approaches

Memory approaches have proved to be very useful in the case of periodical or recurrent changes, wherein it would be useful to reuse previously found solutions by adding memory components to the EAs. This helps in saving computational time and to bias the search process.The memory component can be implicit (as in redundant coding using diploid genomes) or explicit (as in archiving previous good solutions and environmental information). However, drawbacks of this approach lies in the fact that it is effective only for problems with cyclic environments and further, only in problems that return to the same optimum as before. Furthermore, it also results in diversity loss and redundancy coding may not be good for cases where the number of oscillating states is large.

2.4 Prediction approaches

While memory approaches can perform very well in problems with periodic or recurrent changes, they do not perform well with other types of changes which may also exhibit some patterns. In such dynamic environments, it would be sensible to learn from such patterns that are predictable and predict changes in the future. Memory approaches can indeed be considered as a subset of prediction approaches. In such approaches, a learning model is built to estimate the current change and use the knowledge to predict for the next change, to generate new individuals that best match the estimation. A detailed overview of various works following this approach is given, as prediction strategy is the proposed mechanism for solving DMO problems in this thesis.

[13] have combined a time series forecasting technique, autoregressive model with an EA to predict the movement of the moving optima. The

concept used here is to maintain two anchor points at each end of the pareto front of a 2 objective problem. The forecasting technique is used to track the movement of these 2 points and predict for the future when a change is detected. A more advanced strategy is used in the proposed methodology in the paper to account for 3 objective problems and not restricted to 2 objective problems. Further, problems with complicated pareto sets could also be solved using the proposed methodology, which will be discussed in detail in the following chapters. In [14], the focus has been on utilizing history information to guide future search by firstly, predicting the new location of individuals and secondly, by perturbing the current population with a Gaussian noise whose variance is estimated according to previous changes. A simple linear model is used as the prediction model to estimate the new location of solutions in the decision space.

Some of the earlier works have focused on combinatorial optimization problems as well. In [15], the authors have investigated the usage of 2 prediction mechanisms in using EAs for dynamic environments. Linear regression is used to predict the generation when a change in the environment will occur, and Markov chains are used to predict to which state (or states) the environment may change. This strategy combines the usage of prediction and memory approaches and was successfully applied to several instances of the dynamic bit matching problem. The same authors have also stressed on the importance of better anticipating the changes in the landscape and maximizing the EA's adaptability and have implemented nonlinear regression mechanisms to evaluate the predictor's accuracy [16]. Apart from the other works using this approach, [17] have proposed a predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. The predicted direction and magnitude of the next change, known as the predictive gradient, is estimated based on

the history of previously discovered solutions using a weighted average approach.

Solving DMO problems has not been restricted only to genetic algorithms. Other evolutionary computation algorithms such as Artificial life [18], Particle Swarm Optimization (PSO) [19], Differential Evolution (DE) [20] algorithms have also been implemented to tackle dynamic environments. More recently, [21] have proposed usage of an Estimation of Distribution Algorithm (EDA), RM-MEDA [22] together with autoregressive time series forecasting technique to solve DMO problems. In this work, they have focused on a population prediction strategy which utilizes the properties of continuous DMOPs by modeling the pareto set as a center point and manifold. The movement of centres is learnt by maintaining a sequence of center points by the time series model and the previous manifolds are used to estimate the next manifold. They have systematically compared the proposed strategy PPS with 2 other strategies on a variety of test instances with linear or nonlinear correlation between design variables and the statistical results have shown that PPS is promising for dealing with dynamic environments.

2.5 Self-adaptive methods

The basic notion behind this approach is to make use of the self-adaptive characteristics of EAs to cope with the changes. Some researchers have adaptively tuned the different parameters such as mutation rate, crossover probability, selection ratio, etc by encoding them in the genomes [23] [24]. These methods still do not give better performance than hypermutation and the difficulty level is high when trying to solve complex dynamic optimization problem where the velocity of the moving peaks is not constant (in the case

of the moving peaks benchmark problem [25]). Here also, some researchers have used other EAs such as Evolution Strategy (ES) and Evolutionary Programming (EP).

2.6 Multi-population approaches

This approach can be considered as a combination of diversity introduction/maintenance, memory and adaptation by maintaining multiple sub-populations concurrently which may take in different search areas or even different tasks. Algorithms using this approach are numerous in the literature. While most of them have focused on dynamic single objective optimization, more recently, [26] have implemented a Competitive-Cooperative Coevolutionary paradigm to solve dynamic multiobjective optimization problems. This algorithm is considered as one of the state-of-the-art algorithms in DMO currently.

Although various approaches have been in vogue in evolutionary dynamic optimization, there is still a growing need for more efficient algorithm to solve different kinds of DMO problems. Most of the previous works have focused on a particular kind of problems and their performance can still be improved upon. In the following sections, a novel Kalman filter based EA for dynamic MO is designed and its performance is analysed compared to other state-of-the-art algorithms.

Chapter 3

Kalman Filter Prediction based Evolutionary Dynamic Multiobjective Optimization

3.1 Introduction

Multiobjective Optimization (MOO) involves finding a set of trade-off solutions by optimizing conflicting objectives simultaneously. Most real-world problems are multiobjective in nature. Evolutionary algorithms (EAs) which evolve multiple solutions in a single run are good candidates to solve MO problems compared to classical methods such as gradient descent and simulated annealing [1]. Various Multiobjective Evolutionary Algorithms (MOEAs) exist that are capable of attaining the MO goals of convergence and diversity with high efficacy [27–29]. Many-objective optimization is one of the recent topics that has gathered Evolutionary Computation researchers' attention [30–36]. In addition to multiobjectivity and many-objectivity, real-world optimization problems have various uncertainties [37] and dynamics which need to be handled effectively [38]. These uncertainties

and dynamics can occur in the form of fluctuating stock prices in financial markets, new jobs or breakdown of machines in a production line, and inflation of component costs in a design scenario [39] [40]. Instances of dynamic problems in literature are dynamic job shop scheduling [41], hydro thermal power scheduling [7], war resource allocation [42] and UAV online path planning [43], to name a few.

While static MOO and dynamic single-objective optimization problems are addressed using EAs, little attention was given to dynamic MO problems until recently. The objective functions, constraints, and decision variables may vary with time in Dynamic Multiobjective Optimization Problems (DMOPs). EAs are useful in solving DMOPs as they are inspired by natural evolution which is a continuous process of adaptation [44]. Traditional EAs once converged, cannot adapt quickly to environmental changes [45]. Speed of convergence is considered an important issue in dynamic multiobjective optimization [46] [44] [17]. An ideal dynamic Multiobjective Evolutionary Algorithm (DMOEA) must possess fast convergence capabilities to track the varying optimum solutions effectively either through inherent design or by incorporating additional dynamic handling techniques [17]. Changes in dynamic environments may exhibit some patterns that are predictable. Consequently, from the past optimum solutions, subsequent changes based on the patterns exhibited can be predicted. In this work, a novel prediction mechanism using Kalman Filter is proposed for solving DMOPs.

The number of state estimation and tracking applications in which Kalman filter has been applied cannot be overstated [47] [48]. Genetic algorithms have been traditionally used to tune the parameters of Kalman Filter in a variety of applications [49] [50]. Kalman Filter has also been used with evolutionary algorithms in noisy objective evaluations [51] in a multiobjective optimization context. Hybridization of GA and Kalman

Filter for optimization problems is not uncommon [52] [53]. A vision based tracking robotic application in a dynamic single objective optimization context used genetic algorithms to search for the optimum together with Kalman filter to incorporate motion information [54].

Kalman Filter is an algorithm that uses a series of measurements observed over time, containing noise and other inaccuracies, and produces statistically optimal estimates of the underlying system state [55]. The algorithm works in a two-step process involving a prediction step and a measurement step. In the prediction step, the Kalman filter estimates the current state *a priori*. Once the subsequent measurement is obtained, the *a priori* estimates from the Kalman filter are updated to obtain the *a posteriori* estimates. The Kalman filter can run in real-time, thereby making it a good candidate for the prediction model in solving DMOPs. In the proposed algorithm, the Linear Discrete Time Kalman filter [56] is applied to the whole population to direct the search for Pareto Optimal Solutions (POS) in the decision space after a change has occurred. Although nonlinear formulations of the Kalman filter are available by means of the Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF) [57], the state transition and observation matrices which are required in the prediction and update steps are formed by Jacobian(matrix containing partial derivatives of f with respect to x) of the nonlinear functions, which are not directly available to us. Though particle filters may give better results for non-linear conditions, it would come at the price of large computational cost [58]. Thus, considering the optimal state estimation performance of Kalman Filter in linear conditions and its low computational cost, we use the Kalman filter over other methods for predicting the pareto optimal sets in our work.

The rest of the chapter is organized as follows. Section 3.2 provides background on related work. Section 3.3 presents the proposed algorithm.

The underlying MOEA, MOEA/D-DE and the Kalman Filter based prediction method are also elaborated. Section 3.4 describes the experiment results and performance comparisons. Section 3.5 provides the discussion and analysis. Section 3.6 concludes the work and potential future research directions are highlighted.

3.2 Related Work

Many methods have been proposed and investigated in the past, not restricted to evolutionary DMO . The main approaches suggested include,

1. diversity introduction after a change [5] [6] [59],
2. diversity maintenance throughout the run [10] [11],
3. memory approaches [25],
4. prediction approaches [13] [14] [17],
5. self-adaptive methods [24] [25] [60], and,
6. multi-population approaches [26] [61] [62].

The above approaches have been well investigated in evolutionary dynamic single objective optimization, but only a few strategies have been proposed to solve DMOPs. Farina et al. [2] proposed a direction-based hybrid algorithm based on evolutionary strategies and deterministic local search to increase the convergence speed by searching for nadir points, Utopia points and payoff matrix. This is followed by searching for uniformly distributed solutions between the Utopia points. The optimization process is restarted when change is detected. Deb et al. [7] extended the NSGA-II to Evolutionary DMO by proposing to solve a hydro-thermal power

scheduling problem. 10% of the population is re-evaluated at the start of each generation to check for changes. Adaptation to dynamic environments is achieved either by following the random immigrants strategy [8] or by hypermutation [5].

Orthogonal design methodology has been incorporated in Dynamic Orthogonal Multi-objective Evolutionary Algorithm (DOMOEA) [63] to improve convergence. Dynamic Queuing Multi-objective Optimizer [13] exploits past information by employing an autoregressive model to estimate the location of the changed pareto optimal set and the predicted individuals are used to seed the new population after a landscape change is detected. In contrast, changing dimension of objective space is also accounted for in the development of Multiobjective Optimization Immune Algorithm [18] where a novel algorithm suitable for DMO problems is proposed based on Pareto dominance and immune functions of the germinal center in the immune system. A new coevolutionary paradigm is proposed by Goh et al. [26] that hybridizes competitive and cooperative mechanisms observed in nature to solve multiobjective optimization problems and to track the Pareto front in a dynamic environment. The aforementioned approach falls under the multi-population category, wherein a number of subpopulations compete with each other to find the best solutions for their subcomponent. Finally, they cooperate to evolve for better solutions thereby enabling the algorithm to adapt and emerge better solutions in dynamic environments. DMOPs have also been solved by Particle Swarm Optimization methods [19] [64].

A few algorithms based on the prediction strategy seemed to have become increasingly popular. Koo et al. [17] proposed a prediction strategy, wherein a predictive gradient (predicted direction and magnitude of the next change) is estimated based on the history of previously discovered solutions using a weighted average approach. A new memory technique is introduced to

exploit any periodicity in the dynamic problem. The importance of utilizing history information to guide future search is further highlighted by Aimin et al. in their Feed-forward prediction strategy (FPS) [14] and Population prediction strategy (PPS) [21] algorithm proposals. FPS uses a simple linear prediction model to predict the location of the individual for the next change. Four types of reinitialization procedures are experimented with FPS - Random reinitialization, Variation (perturbing the individuals by an amount estimated from previous changes), Prediction and a Hybrid of the previous two methods. In PPS, an Estimation of Distribution Algorithm together with autoregressive time series forecasting technique is proposed. This algorithm uses the properties of DMOPs by modeling the changing pareto optimal set as a center point and manifold. The movement of centers is learnt by maintaining a sequence of center points by the time series model and the previous two manifolds are used to estimate the subsequent manifold.

While research on dynamic single objective optimization has been extensive [65] [66] [67] [68] [3], algorithms proposed to solve DMOPs are only a handful. Further, usage of powerful prediction techniques to assist EAs in solving DMOPs has not been fully explored yet. Many of the prediction techniques require a training or learning time before which they can be used for tracking the changing POF/POS. Kalman Filter can estimate the state of a process without any such learning time. It can predict from the start and correct itself based on subsequently made measurements. In this thesis chapter, we propose a novel DMOEA algorithm based on Kalman filter predictions in the decision space [69] [78]. Further, while employing this prediction approach as the tracking technique, other approaches such as diversity maintenance / introduction, and memory, can also be used simultaneously to build more efficient algorithms.

3.3 Algorithm Design

In this section, the underlying EA for the proposed DMOEA is outlined, followed by a brief introduction to the Kalman Filter. The Kalman Filter prediction model, change detection function and scoring scheme used are described in detail.

3.3.1 Multiobjective Evolutionary Algorithm with Decomposition based on Differential Evolution

The prediction model of Kalman filter proposed is built on the structure of Multiobjective Evolutionary Algorithm with Decomposition based on Differential Evolution (MOEA/D-DE) [29]. MOEA/D-DE has received significant attention due to its good optimization performance in solving continuous multiobjective optimization problems with relatively fast convergence and diverse spread. The algorithm decomposes a problem into several subproblems and simultaneously optimizes them using neighborhood relations. The neighborhood relations are defined based on the distances among their weight vectors. The decomposition is performed using classical approaches, such as the Tchebycheff approach or the weighted sum approach. The Tchebycheff approach is used in this work due to its simplicity and decent optimization performance.

3.3.2 Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient computational means to estimate the state of a process, in a way that minimizes the mean of the squared error [56]. The filter is very powerful in several aspects as it provides the past, present and future estimates even when the precise nature of the modeled system is unknown.

The Kalman filter addresses the general problem of estimating the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation,

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{3.1}$$

with a measurement $z \in \mathbb{R}^m$ that is given by,

$$z_k = Hx_k + v_k, \tag{3.2}$$

where, A is the *state transition matrix* that relates the state at the previous time step, $k-1$ to the state at the current step k. B, the *control input matrix* relates the optional control input, $u \in \mathbb{R}^l$ to the state x. H in the measurement equation (4) relates the state to the measurement, z_k and is known as the *measurement matrix*. The random variables w_k and v_k represent the process and measurement noise respectively. They are assumed to be independent of each other, white and with normal probability distributions$(5, 6)$.

$$p(w) \sim N(0, Q) \tag{3.3}$$

$$p(v) \sim N(0, R) \tag{3.4}$$

The matrices Q and R are the *process noise* and *measurement noise covariance* matrices respectively. These matrices are assumed as diagonal matrices with equal elements which can lead to saving computational effort in estimating the exact values [51]. The various matrices in equations 3.1, 3.2, 3.3 and 3.4 can be time varying, however, for simplicity they are assumed to be invariant.

3.3.3 Kalman Filter Prediction Model

The Kalman filter estimates a process by using a form of feedback control. The filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. The discrete Kalman Filter is employed in this work. The equations for the discrete Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations, which can be thought of as *predictor* equations, are responsible for projecting forward in time the current state and the error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement equations can be thought of as *corrector* equations and are responsible for the feedback wherein they incorporate a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The Linear Discrete Time Kalman Filter is used to predict the location(s) of the Pareto Optimal Set after a change is detected in the problem. A simple state transition matrix is assumed to represent the process : $x_k = x_{k-1} + \dot{x}_{k-1}\Delta t$. In matrix formulation, the above equation becomes

$$\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix}. \tag{3.5}$$

This one dimensional example is extended to n dimensions, to obtain the state transition matrix for the n-dimensional state (decision variables) and their estimated velocities.

The equations [56] for the time update and measurement update steps

are presented in (8) and (9). The equations for the time update step are,

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1}, \\ P_k^- &= AP_{k-1}A^T + Q, \end{aligned} \tag{3.6}$$

and, the equations for the measurement update step are,

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1}, \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-), \\ P_k &= (I - K_k H)P_k^-, \end{aligned} \tag{3.7}$$

where x is the state vector to be estimated by the Kalman Filter, A denotes the state transition matrix, u is the optional control input to the state x, B is the control input matrix, P is the error covariance estimate. z denotes the measurement of the state vector, H is the observation matrix and the process and measurement noise covariance matrices are Q and R respectively. K is the Kalman filter gain.

As shown, the current estimates are made using only the previous predictions and the current observation. There are two variants of Kalman Filters designed for prediction, a two-dimensional Kalman Filter (2by2KF) and a three-dimensional Kalman filter (3by3KF). In both the variants, the observation matrix is the identity matrix, since the decision variables can be directly measured from the EA. Further, there are no control inputs in the system. The process and observation noise are Gaussian noise of zero mean and assumed variance. The corresponding covariance matrices Q and R can be calculated.

2by2 Kalman Filter (2by2KF)

The state vector is

$$X = \begin{bmatrix} x & v \end{bmatrix}^T, \tag{3.8}$$

where x is the vector for the decision variables and v is the vector of the first order change in the decision variables. The state transition model used is

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{3.9}$$

and the covariance of state vectors is initialized as

$$P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.10}$$

In this case, the Kalman Filter is a first order linear model perturbed by Gaussian noise. The initial covariance P_0 suggests some uncertainty in the initial state vectors, which is adaptive and will be updated as time proceeds. Noise cannot be modelled exactly in this context and assumed to be a constant Gaussian noise.

3by3 Kalman Filter (3by3KF)

The state vector in this case is

$$X = \begin{bmatrix} x & v & a \end{bmatrix}^T, \tag{3.11}$$

where x, v are the same as in 2by2KF, and a is the vector of the second order change in the decision variables. The state transition model used is

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.12}$$

and the covariance of state vectors is initialized as

$$P_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

In this case, the Kalman Filter is a second order linear model perturbed by Gaussian noise. According to the state transition model, the decision variables are updated by previous state and first order rate of change only. The second order vector a, is controlled by the Gaussian noise and is only used to estimate the first order change term of the state vector, as can be deducted from the state transition model. It is designed in this way to include more historical information but not greatly rely on it. Both the variants are discrete and linear Kalman Filters. Higher order change is ignored in trade-off for speed and computational resources.

3.3.4 Change Detection Function

The relationship of MOEA/D-DE with the Kalman Filter (KF) prediction model is shown in Figure 3.1.

When there is no change detected, MOEA/D-DE takes control and the population evolves accordingly. Otherwise, the Kalman Filter prediction model directs the search for Pareto optimal solutions in the decision space.

A change detection function is needed to combine the prediction model with the MOEA/D-DE algorithm. Assuming that there is no noise in objective functions evaluation, some individuals are randomly selected as detectors and their objective values are stored in the system. At the beginning of each generation, the detectors' objective values are recalculated and compared with the previously stored values. A mismatch in the objective values suggests that a change in the problem has occurred caused by moving

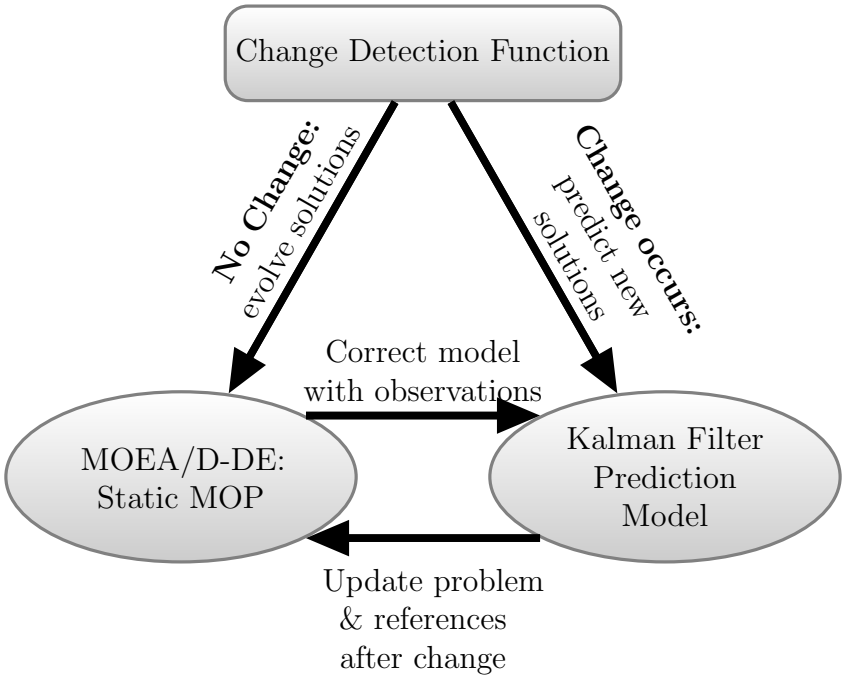


Figure 3.1: Relationship of EA with Kalman Filter model

POS or POF landscape.

3.3.5 Scoring Scheme

Since Kalman Filter prediction makes the assumption of linear dynamic model of the system, it may cause some problems when the system violates the assumption. To circumvent such a situation, random re-initialization method (RND) is introduced into the algorithm. A scoring scheme (SC) is proposed to hybridize Kalman Filter prediction and random re-initialization method. The diagram illustrating the relationship of the scoring scheme with the proposed model is shown in Figure 3.2.

In order to allocate resources efficiently, the scoring scheme computes a score or proportion of using random re-initialization method against Kalman filter prediction. A random number can then be generated and if it is smaller than this score, random re-initialization is used; otherwise, Kalman Filter prediction is used. The procedure following by the scoring scheme based

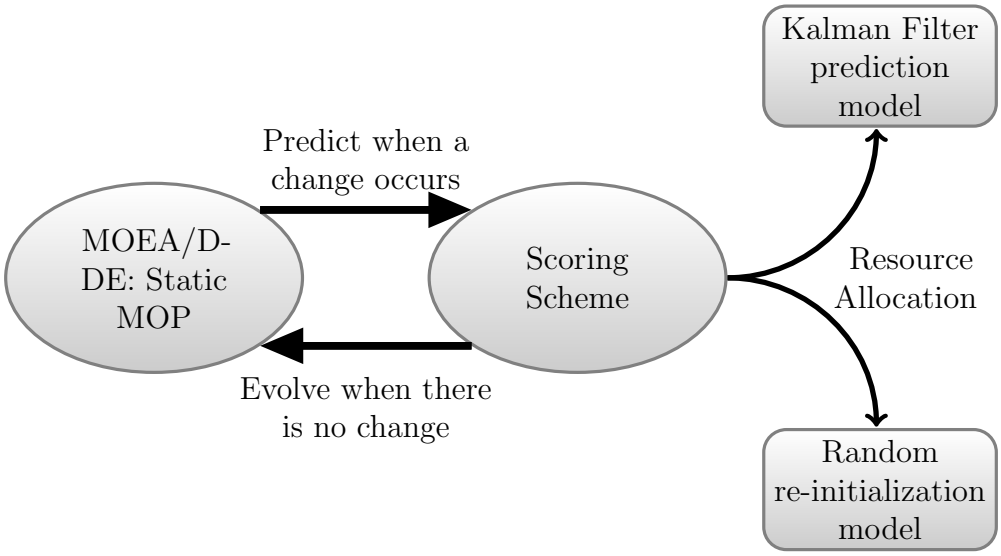


Figure 3.2: Relationship diagram for scoring scheme.

model is given in Algorithm 3.1.

To start with, the chance of producing one solution for the next generation is 50-50 between random re-initialization method and Kalman Filter prediction. The method used to produce each child solution is stored as an attribute of the individual. After a change is detected, the Euclidean distances from the solutions just before the change to the solutions after the previous change are computed and the average is taken. A smaller-than-average distance implies that the improvements made over generations by MOEA/D-DE are small. Therefore, the corresponding prediction method in use is likely to produce solutions closer to the POS in the current setting. The scores of both methods are recorded and normalized by the total number of solutions produced by each method. The overall score of random re-initialization to Kalman Filter prediction is then calculated by dividing the re-initialization score to the sum of both scores.

The scoring scheme dynamically alters the probability of implementing either one of the methods, with the higher probability favoring the method that performs better in the previous prediction. It is to be noted that even if the random re-initialization model is used, the measurement update step

Algorithm 3.1 Scoring Scheme based prediction model

Require:

N: Population size

rnd_{count}, kf_{count}: Number of individuals that are reinitialized/predicted using RND and Kalman Filter(KF) method respectively

rnd_{score}, kf_{score}: Proportion of individuals that RND and KF reinitialize/predict with less than average distance respectively

Loop through Steps 1 to 4 for every change detected

Step 1 ➢Initialization:

1. Scores and counts are initialized to zero, i.e.

$$rnd_{score},\ kf_{score},\ rnd_{count},\ kf_{count} = 0$$

Step 2 ➢Average distance calculation:

1. For each individual i in the population of size N, Euclidean distance($dist_i$) between solution, i at t_{k-1}^{+} to t_k^{-} is computed. k is a time instant.
2. Average of the distances is calculated.

$$dist_{average} = \frac{1}{N}\sum_{i=1}^{N} dist_i$$

Step 3 ➢Score Computation: For each individual i in the population,

1. If i used RND during previous change, increment rnd_{count} by 1. Else increment kf_{count} by 1.
2. Increment corresponding score by 1, if $dist_i < dist_{average}$

Normalize scores:

$$rnd_{score} = \frac{rnd_{score}}{rnd_{count}},$$

$$kf_{score} = \frac{kf_{score}}{kf_{count}},$$

$$overall_{score} = \frac{rnd_{score}}{(rnd_{score} + kf_{score})}$$

Step 4 ➢Method Selection: For each individual i in the population,

1. Perform Kalman filter measurement update
2. Generate random number, r between 0 to 1
3. if $r < overall_{score}$ use random re-initialization method, else use Kalman filter prediction method

of the Kalman Filter model is still performed. This is to keep track of the changes in the system and update the model, in case that the Kalman Filter prediction model is used later for an individual.

The various steps of the proposed model to solve DMOP are shown in Algorithm 3.2 for clarity.

3.4 Empirical Study

3.4.1 Benchmark problems

The proposed algorithm is tested on problems from 3 test benchmark suites - FDA [2], dMOP [26], and F [21]. The FDA benchmark suite is commonly used in the performance evaluation of DMO algorithms. It consists of 5 different problems pertaining to the different types of DMOP. Two of the problems (FDA3 and FDA5) have time varying density distribution of solutions along the pareto front. The dMOP benchmark problems are an extension of the FDA benchmark suite to test further performance characteristics of DMO algorithms such as learning that the POS/POF does not change. The problem suite proposed in [21] is very recent and consists of 10 problems which are partly adopted from the above 2 benchmark suites. Nevertheless, they have also proposed 6 new test instances in which non-linear linkages between the decision variables are considered and problems with sharp and irregular environments are also constructed.

3.4.2 Parameter Settings

The proposed Kalman filter prediction model is implemented in MOEA/D-DE, which is referred as MOEA/D-KF for simplicity. The parameter settings for the experiments of the various test benchmark suites are tabulated in

Algorithm 3.2 MOEA/D-DE with Kalman Filter prediction for Dynamic Multiobjective Optimization

Require:

MOP

A stopping criterion

N: Population size

P: the number of subproblems considered in MOEA/D

A uniform spread of N weight vectors: $\lambda^1, \lambda^2, \ldots, \lambda^P$

T: neighbourhood size

Kalman Filter Parameters

Ensure:

Approximated POF $\{f^1, \ldots, f^N\}$

Approximated POS $\{x^1, \ldots, x^N\}$

Step 1 ➢ Initialization:

1. Generate evenly spread weight vectors. Initialize the neighbourhood of each vector by finding its T closest weight vectors in terms of Euclidean distance.
2. Generate an initial population, $\mathbf{x}^1, \ldots, \mathbf{x}^N$ by uniform random initialization within the decision space. Evaluate objective function values of each solution and set $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i)$.
3. Initialize Kalman Filter matrices and vectors for each solution. Initial population decision variables are set as the initial state of the Kalman Filter.
4. Initialize ideal vector by setting $z_k = \min_{j=1,\ldots,N} f_k^j$ where $k = 1. \ldots, m$
5. Randomly initialize a set of detector individuals within the decision space for change detection.

Step 2 ➢ Update:

1. **Change Detection:** Evaluate the objective function values of the detector individuals and check whether they have changed to indicate a change in the dynamic multiobjective problem. If change is detected go to **Step 2.2**. Otherwise, go to **Step 2.4**
2. If scoring scheme based model, iterate through the steps in Algorithm 1
3. If Kalman Filter prediction, perform
   (a) Measurement Update
   (b) Time Update

   Else, perform random reinitialization.
4. Reproduction: Mating selection, Differential Evolution, update neighbourhood and the ideal vector

Step 3 ➢ Stopping Criteria: If stopping criteria is satisfied, then stop and output the final population and their corresponding values in the objective space. Otherwise, go to **Step 2**.

Table 3.1. The number of decision variables is set as 20 for all the test problems. FDA4, FDA5, and F8 are 3-objective problems and are assigned a population size of 200 for better search capability, while the rest of the problems which are 2-objective are assigned a population size of 100. The various parameters for MOEA/D-DE are implemented as guided in [29]. 10 detector individuals are utilized for change detection purpose.

A random reinitialization method (RND) is implemented for baseline performance comparison of MOEA/D-KF. In this algorithm, instead of the Kalman filter prediction model, 20% of the population is randomly reinitialized after a change is detected.

The matrices of Kalman Filter are initialized according to the description given in Section 3.3.3. The Q and R diagonal matrices are with equal element values q and r, respectively 0.04 and 0.01. These numbers are obtained by tuning the values of q and r for FDA1 resulting in best performance. The change frequency (τ_T) and severity (n_t) setting of the problems determine their difficulty. A reasonable setting of $\tau_T = 30$, and $n_t = 10$ is used. Each algorithm is run 30 times for each test instance independently for 161 environmental changes in each run.

3.4.3 Performance Metrics

A number of metrics are in use for performance assessment of static MOEAs which evaluate convergence and diversity quite effectively. These metrics have been modified for evaluating DMOEAs. The Inverted Generational Distance (IGD) is a unary performance indicator which provides a quantitative measurement for the proximity and diversity goal of MOO [70]. It is

Table 3.1: Experiment Settings

Number of decision variables, n	20 for all test problems
Population size	100 for 2 objective problems, 200 for 3 objective problems.
Neighborhood	Size: 20.
Probability that parents are selected from the neighborhood	0.9
Decomposition method	Tchebycheff
Differential Evolution	CR = 1.0 and F = 0.5
Polynomial Mutation	$\eta = 20$, $p_m = 1/n$.
Number of detectors	10
Percentage for RND model	20%
KF model process noise	Gaussian of N(0, 0.04)
KF model observation noise	Gaussian of N(0, 0.01)
Dynamic Setting	Frequency of change τ_T: 30, Severity of change n_t: 10
Number of changes	161
Number of generations	4835
Number of runs	30

mathematically given by,

$$IGD(P^{t*}, P^t) = \frac{\sum_{v \in P^{t*}} d(v, P^t)}{|P^{t*}|}, \tag{3.14}$$

where, P^{t*} is a set of uniformly distributed Pareto optimal solutions in the POF at time t (POF^t) and P^t is an approximation of the POF obtained by the algorithm in consideration. d is a distance measure between P^t and P^{t*}, given by,

$$d(v, P^t) = \min_{u \in P^t} \|F(v) - F(u)\|. \tag{3.15}$$

A lower value of IGD implies that the algorithm has better optimization performance. To obtain a low value of IGD, it can be seen from the equations (3.14 and 3.15) that, P^t must be very close to POF^t and cannot miss any part of POF^t, thus measuring both convergence and diversity.

To adapt the IGD metric for DMO [21], the mean of the IGD values in some time steps over a run is taken as the performance metric, given by,

$$MIGD = \frac{1}{|T|} \sum_{t \in T} IGD(P^{t*}, P^t), \tag{3.16}$$

where, T is a set of discrete time points (immediately before the change occurs) in a run and $|T|$ is the cardinality of T. A lower value of the MIGD metric would assist in evaluating the tracking ability, as the approximated pareto front obtained from the algorithm with the changing pareto optimal front is measured before every change. 1000 and 2500 equidistant points along the POF, P^{t*} are chosen for computing the IGD metrics for bi-objective and tri-objective problems, respectively.

Another performance metric that is proposed recently to evaluate the performance of MOEAs is the Averaged Hausdorff Distance proposed in [71]. Shortcomings of the previously widely used indicators, GD (Generational Distance) and IGD have been identified and a new indicator to measure the Hausdorff distance to the pareto front is proposed from the corrected GD (GD_p) and IGD (IGD_p) metrics. The Averaged Hausdorff distance

$\Delta_p(X, Y)$ is defined as

$$\begin{aligned}\Delta_p(X, Y) &= \max\left(GD_p(X, Y), IGD_p(X, Y)\right) \\ &= \max\left(\left(\frac{1}{N}\sum_{i=1}^{N} dist(x_i, Y)^p\right)^{1/p},\right. \\ &\left.\left(\frac{1}{M}\sum_{i=1}^{M} dist(y_i, X)^p\right)^{1/p}\right) \qquad (3.17)\end{aligned}$$

where X and Y are finite sets in the objective space [71]. This metric is also modified similar to IGD to act as a performance metric for evaluating DMOEAs.

3.4.4 Results

The statistical results of MIGD values for the various test benchmark problems are tabulated in Table 3.2.

The performance of the Kalman Filter based models are definitely much better than the performance of the simplistic RND method. This shows that the Kalman Filter prediction model substantially enhances the MOEA adapted for dynamic optimization compared to the static MOEA with restart (partial) mechanism. Kalman3by3SC algorithm performs significantly better than the other Kalman Filter based models and RND in 6 out of the 13 problems.

The benchmark problems' characteristics are such that the changing pareto optimal solutions do not move in the decision space or objective space uniformly. In the proposed Kalman Filter models combined with MOEA/D-DE, the 2by2 variants only account for first order change, while the 3by3 variants take second order change into account as well. As a result, the 3by3 variants tend to perform better in many of the problems. From Table 3.2, it can be observed that the performance of Kalman3by3

Table 3.2: Experiment Results of MOEA/D-KF and RND

Problems	Kalman2by2	Kalman2by2SC	Kalman3by3	Kalman3by3SC	RND
FDA1	0.00836 ± 0.00491(+)	0.00802 ± 0.00489(+)	0.00786 ± 0.00497(+)	**0.00736 ± 0.00495**	0.0192 ± 0.00966(+)
FDA2	0.00622 ± 0.0086(-)	0.00581 ± 0.00867(-)	0.00741 ± 0.00739(+)	**0.00572 ± 0.00753**	0.00615 ± 0.00864(-)
FDA3	0.0307 ± 0.02032(+)	0.0278 ± 0.01731(+)	0.0585 ± 0.07018(+)	**0.0263 ± 0.01626**	0.0388 ± 0.02128(+)
FDA4	0.104 ± 0.04578(+)	0.103 ± 0.04577(+)	0.0926 ± 0.02881(+)	**0.0892 ± 0.02829**	0.123 ± 0.05481(+)
FDA5	0.194 ± 0.07244(+)	0.197 ± 0.07438(+)	0.19 ± 0.06149(+)	**0.167 ± 0.04438**	0.316 ± 0.11351(+)
dMOP1	0.0084 ± 0.02968(+)	0.00725 ± 0.02972(-)	0.0123 ± 0.03098(+)	0.00773 ± 0.03136(+)	**0.00707 ± 0.02977**
dMOP2	0.00977 ± 0.00769(+)	0.00923 ± 0.00768(+)	0.00878 ± 0.00773(+)	**0.00822 ± 0.00768**	0.026 ± 0.01509(+)
F5	0.044 ± 0.03687(+)	0.042 ± 0.03988(+)	0.0292 ± 0.02902(-)	**0.0287 ± 0.03099**	0.149 ± 0.15027(+)
F6	0.0575 ± 0.17951(-)	0.0564 ± 0.18286(+)	**0.0553 ± 0.18959**	0.0566 ± 0.1919(-)	0.0868 ± 0.18803(+)
F7	0.0345 ± 0.10026(+)	0.0329 ± 0.1028(-)	0.0325 ± 0.09218(-)	**0.0318 ± 0.09381**	0.0512 ± 0.10295(+)
F8	0.108 ± 0.03687(+)	**0.101 ± 0.03814**	0.137 ± 0.03798(+)	0.105 ± 0.038(+)	0.111 ± 0.03875(+)
F9	0.243 ± 0.46665(+)	**0.116 ± 0.23707**	0.24 ± 0.50978(+)	0.152 ± 0.39725(+)	0.205 ± 0.16993(+)
F10	0.153 ± 0.10423(+)	0.155 ± 0.14985(+)	0.0793 ± 0.0292(+)	**0.0408 ± 0.0359**	0.351 ± 0.20895(+)

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

and Kalman3by3SC are not significantly different in problems F5-F7, while Kalman3by3SC tends to perform significantly better than Kalman3by3 in most of the other problems. The scoring scheme aims to hybridize the Random Reinitialization (RND) method with the Kalman Filter based prediction method to get the benefit of both the methods. Problems in which the Pareto Optimal Solutions do not change with time, such as FDA2 and dMOP1, if the converged population is not altered it would result in good performance. Thus, in such scenarios, it becomes beneficial to

hybridize the Kalman Filter based prediction method with RND, as RND retains majority of the population to be as is.

In general, scoring scheme based method is better compared to standalone Kalman Filter based methods. The 3by3 variant in particular may be recommended, as second order change is also taken into account, which may lead to better predictions. For more performance analysis on each problem, kindly refer to section 3.5.

The Kalman filter prediction model predicts in the decision space and assists the evolutionary algorithm in the tracking of the Pareto Optimal Solutions (POS) in dynamic environments. The prediction performance for the FDA1 problem is shown in Figure 3.3.

The left half of Figure 3.3 depicts the objective space and the right half depicts the decision space. The Pareto Optimal Front in FDA1 problem remains fixed, while its Pareto Optimal Set changes with time making it a Type I DMOP [2]. In the right half, the solid line represents the POS for the current time instant. In the following time instant, the POS would be shifted to the dotted line. The dotted and solid circles represent the current measurement and prediction estimate of the Kalman filter respectively. It can be seen that the prediction estimate is quite close to the new POS while the solutions obtained by MOEA/D-DE without the Kalman Filter prediction method remain close to the previous POS. Further, the MOEA/D's [29] weight vectors further assist the algorithm in getting a widely distributed set of solutions covering the entire front. Similar performance is observed in most of the other benchmark problems as well.

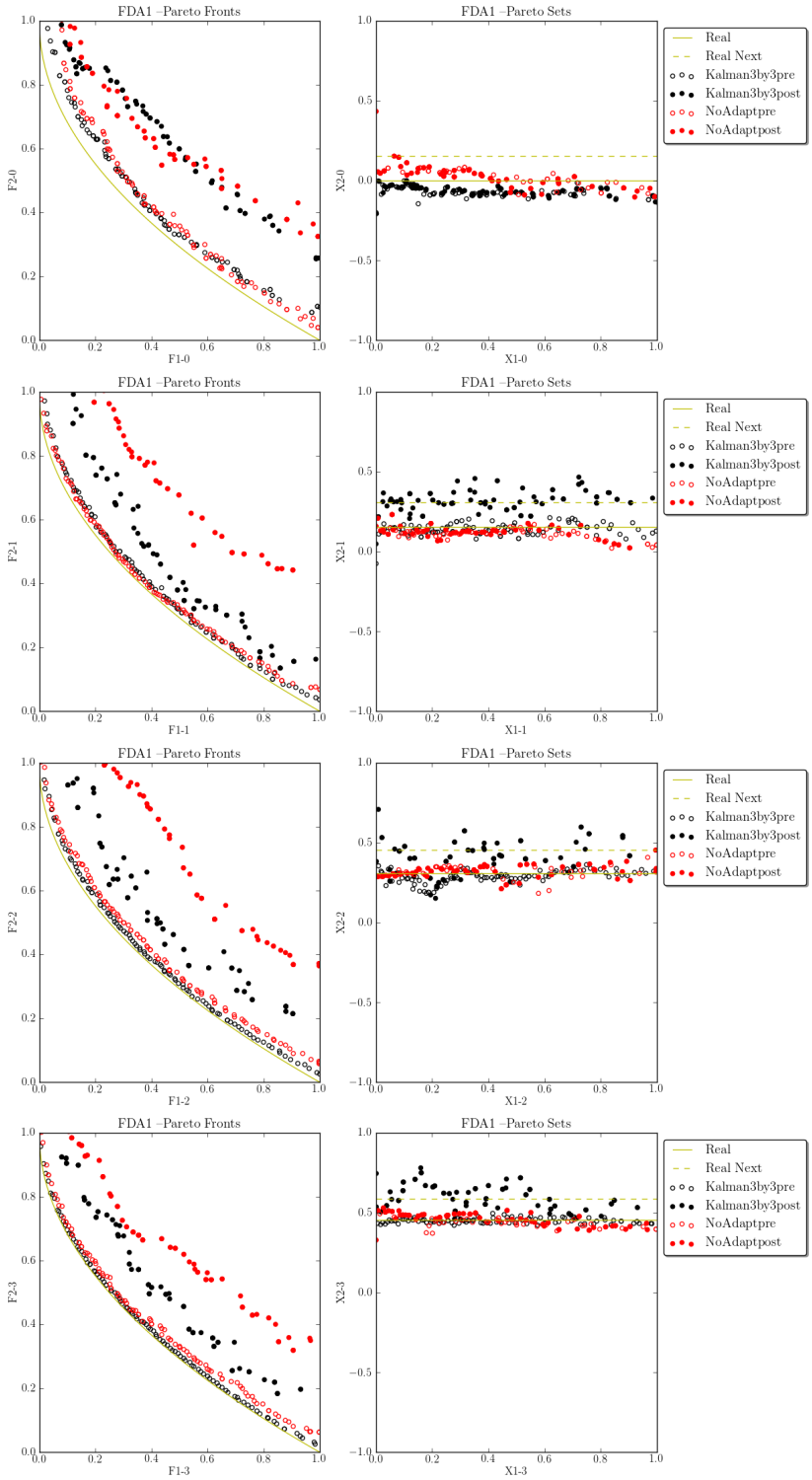


Figure 3.3: Visualization of Kalman Filter prediction performance in FDA1. The time steps associated with the figures (referred in the order - top left, top right, bottom left, bottom right) are the first 4 changes which occur in the problem at generation numbers 30, 60, 90 and 120 respectively. The rings indicate the values before the change, while the circles represent the values after the change. The POF front is indicated in yellow color. The full line represents the current POF while the dotted line represents the POF after the change.

3.4.5 Performance comparison with other DMOEAs

Population Prediction Strategy

The Population Prediction Strategy (PPS) formulated in [21], takes into consideration the properties of continuous DMO problems and is used to predict a whole population. The Pareto set is divided into two parts, a center point and a manifold. A sequence of center points is maintained to predict the subsequent center, and the previous manifolds are used to estimate the subsequent manifold. The univariate autoregression model, a time series prediction method is applied to forecast the next location of the center. The next manifold is obtained by studying the similarity between previous manifolds and estimating the variance which is subsequently used in solution generation. The source code for the PPS algorithm is obtained from the authors of [21] and the parameter settings are maintained in the simulations.

Dynamic NSGA-II

The original Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [27] is modified to handle DMO problems in [7]. 10% of the population members are picked randomly from the parent population to act as change detector individuals similar to the change detection mechanism proposed in the current work. Two versions of dynamic NSGA-II are proposed. In the first version, DNSGA-II-A, a ζ% of the population is replaced with new random solutions whenever there is a change in the problem. In the second version, DNSGA-II-B, a ζ% of the population is replaced with mutated solutions of randomly chosen existing solutions, similar in principle to hypermutation based GAs for single objective optimization [5]. DNSGA-II-A is expected to perform better in problems undergoing a large change, while DNSGA-II-B

may perform better in problems undergoing a small change. The source code of static NSGA-II is obtained from [72] and the 2 dynamic versions are implemented according to [7]. The ζ parameters for the 2 versions, which determine the portion of population reinitialized and the hypermutation rate for the second version are tuned for the benchmark problems. The values used are $\zeta\%$ of 20%, and hypermutation rate η_h of 0.5 which is much higher than the probability of mutation in normal use.

The statistical values of MIGD for the 3by3SC Kalman filter formulation in comparison with PPS, DNSGA-IIA and DNSGA-II-B are presented in Table 3.3.

Statistical test of T-test for independent samples is performed for the MIGD statistical values. The null hypothesis is that the proposed algorithm of Kalman prediction based MOEA, MOEA/D-KF does not perform significantly better than the compared algorithms at the 95% significance level. However, the results tabulated show that the proposed algorithm performs significantly better than the compared algorithms in 7 out of the 13 benchmark problems.

The averaged Hausdorff distance statistics for the proposed algorithm, MOEA/D-KF with the Kalman third order formulation, and the comparison strategies of RND and PPS are provided in Table 3.4. T-test is performed on these statistical values at the 95% significant level as well. It is observed that the MOEA/D-KF algorithm performs significantly better than the compared strategies in 9 out of the 13 problems.

3.5 Discussion

In this section, the results on various benchmark problems are elaborated and the analyses on parameter settings and influence of frequency of change

Table 3.3: Performance Comparison with other DMOEAs

Problems	MOEA/D-KF	PPS	DNSGA-II-A	DNSGA-II-B	RND
FDA1	**0.00736 ± 0.00495**	0.0136 ± 0.0429(+)	0.0405 ± 0.10008(+)	0.0721 ± 0.09713(+)	0.0192 ± 0.00966(+)
FDA2	**0.00572 ± 0.00753**	0.00877 ± 0.01404(+)	0.0806 ± 0.13219(+)	0.144 ± 0.12528(+)	0.00615 ± 0.00864(-)
FDA3	**0.0263 ± 0.01626**	0.238 ± 0.28847(+)	0.147 ± 0.16913(+)	0.216 ± 0.09985(+)	0.0388 ± 0.02128(+)
FDA4	**0.0892 ± 0.02829**	0.148 ± 0.05673(+)	0.398 ± 0.07428(+)	0.238 ± 0.10781(+)	0.123 ± 0.05481(+)
FDA5	**0.167 ± 0.04438**	0.201 ± 0.06615(+)	0.324 ± 0.10786(+)	0.354 ± 0.13003(+)	0.316 ± 0.11351(+)
dMOP1	0.00773 ± 0.03136(+)	0.0281 ± 0.0958(+)	0.198 ± 0.7372(+)	0.245 ± 0.69538(+)	**0.00707 ± 0.02977**
dMOP2	**0.00822 ± 0.00768**	0.0183 ± 0.0655(+)	0.088 ± 0.17409(+)	0.27 ± 0.15227(+)	0.026 ± 0.01509(+)
F5	**0.0287 ± 0.03099**	0.0344 ± 0.09251(-)	0.132 ± 0.15775(+)	0.577 ± 0.14982(+)	0.149 ± 0.15027(+)
F6	**0.0566 ± 0.1919**	0.0584 ± 0.25025(-)	0.0836 ± 0.15129(+)	0.153 ± 0.15904(+)	0.0868 ± 0.18803(+)
F7	0.0318 ± 0.09381(+)	**0.0268 ± 0.13148**	0.0802 ± 0.11834(+)	0.149 ± 0.14522(+)	0.0512 ± 0.10295(+)
F8	**0.105 ± 0.038**	0.408 ± 0.18856(+)	0.425 ± 0.35938(+)	0.245 ± 0.36647(+)	0.111 ± 0.03875(+)
F9	0.152 ± 0.39725(+)	**0.106 ± 0.15577**	0.124 ± 0.1606(-)	0.437 ± 0.13881(+)	0.205 ± 0.16993(+)
F10	**0.0408 ± 0.0359**	0.106 ± 0.10662(+)	0.119 ± 0.09741(+)	0.271 ± 0.12189(+)	0.351 ± 0.20895(+)

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

are discussed.

3.5.1 Results on FDA1-FDA5, dMOP1 and dMOP2

From Table 3.3 and Table 3.4, it is observed that the proposed Kalman Filter prediction based dynamic MOEA, MOEA/D-KF, performs significantly

Table 3.4: Averaged Hausdorff distance statistics

Problem	MOEA/D-KF	PPS	RND
FDA1	**0.0337 ± 0.7831**	0.04013 ± 0.11846(-)	0.06816 ± 0.03545(+)
FDA2	**0.02111 ± 0.3294**	0.03052 ± 0.05248(+)	0.02139 ± 0.0318(-)
FDA3	**0.78926 ± 0.00001**	99.9425 ± 1260.895(+)	0.82426 ± 1.21056(-)
FDA4	**0.47897 ± 0.00001**	0.7622 ± 0.28473(+)	0.68908 ± 0.31593(+)
FDA5	**0.92068 ± 0.00001**	1.06425 ± 0.36334(+)	1.82485 ± 0.6473(+)
dMOP1	0.03653 ± 0.00001(-)	0.12332 ± 0.36318(+)	**0.03304 ± 0.14687**
dMOP2	**0.0308 ± 0.00001**	0.0653 ± 0.23301(+)	0.08739 ± 0.05148(+)
F5	**0.15191 ± 0.00001**	0.22393 ± 0.67515(+)	1.51016 ± 1.91353(+)
F6	**0.23105 ± 0.00001**	0.39948 ± 1.82883(+)	0.45312 ± 0.69313(+)
F7	**0.12907 ± 0.00001**	0.19043 ± 1.05816(+)	0.25202 ± 0.35812(+)
F8	**0.83853 ± 0**	2.19536 ± 1.0138(+)	1.79629 ± 1.74289(+)
F9	**0.57029 ± 0.00001**	0.81257 ± 0.96851(+)	1.61727 ± 1.74865(+)
F10	**0.17825 ± 0**	1.12794 ± 1.44951(+)	1.9724 ± 1.09394(+)

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

better than the other algorithms. MOEA/D-KF predicts the solution set much more efficiently than the second-best algorithm, PPS in almost all of the problems right from the beginning. PPS takes a significant amount of time (23 changes, which is the training time of the autoregressive filter prediction model in PPS) before providing reasonable results. This can be observed in the peaks in IGD values of PPS till just after 20 changes in Figures 3.4 and 3.5. The Kalman filter does not require any learning time and starts to provide reasonable predictions from the start of the run. The changing pareto optimal set of FDA1 and dMOP2 follow a sinusoidal

pattern. The simple linear filter model, implemented as the prediction method in MOEA/D-KF, is able to provide reasonable performance except for those instants when the sinusoid changes direction and causes the IGD trend to follow patterns as shown in Figure 3.4 and Figure 3.5.

Both in FDA2 and dMOP1, the optimal values of a number of or all of the decision variables are observed to remain the same throughout a run. MOEA/D-KF and PPS produce stable results throughout the entire run where less fluctuations in IGD values are observed. FDA4 and FDA5 are 3-objective problems. The absolute values of MIGD obtained for FDA4 and FDA5 are slightly lower than those obtained for 2-objective problems. In FDA3 and FDA5, the density of solutions along the pareto front changes with time, thereby making the problems challenging. MOEA/D-KF has MOEA/D as the underlying MOEA, which produces an evenly distributed set of solutions throughout the evolution, and tends to perform more consistently than PPS.

3.5.2 Results on F5-F8

F5, F6 and F7 are Type II DMOPs and have non-linear linkages among the decision variables. It is observed that PPS performs significantly better than MOEA/D-KF. In spite of MOEA/D-KF's immediate results in the start, PPS is able to outperform MOEA/D-KF in later stages. It was shown in [21] that the underlying MOEA plays a significant role in the dynamic optimization performance of the algorithm. The underlying algorithm of PPS, RM-MEDA [22] is an estimation of distribution algorithm (EDA) which is able to learn linkages among decision variables. MOEA/D-KF's performance in problems with non-linear linkages may be improved by hybridizing with state-of-the-art EDAs [73], wherein the superior optimization performance of decomposition based approach in MOEA/D-DE and linkage learning

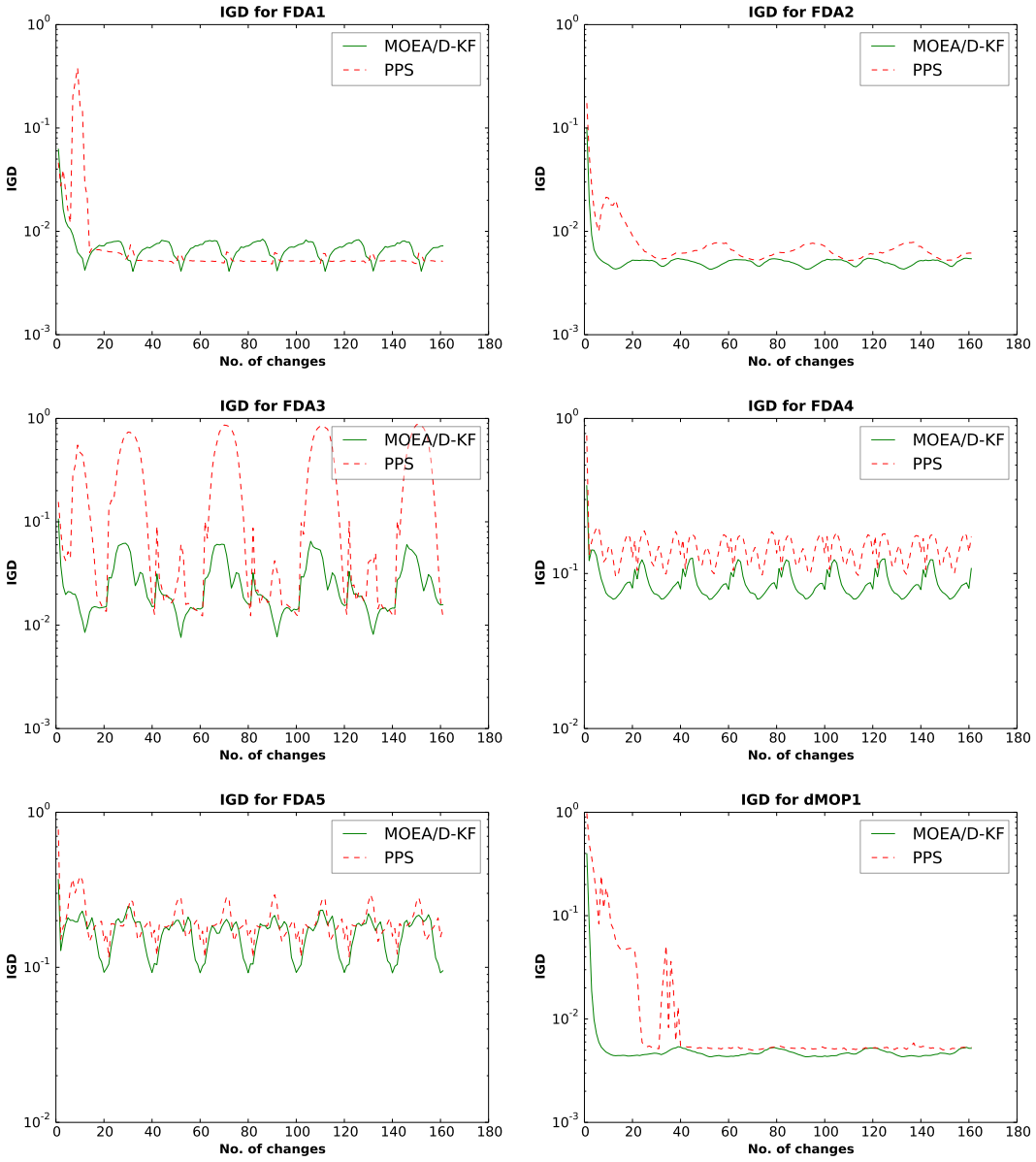


Figure 3.4: IGD Trend comparison of MOEA/D-KF and PPS algorithms over number of changes for 30 runs : FDA1 - FDA5, dMOP1

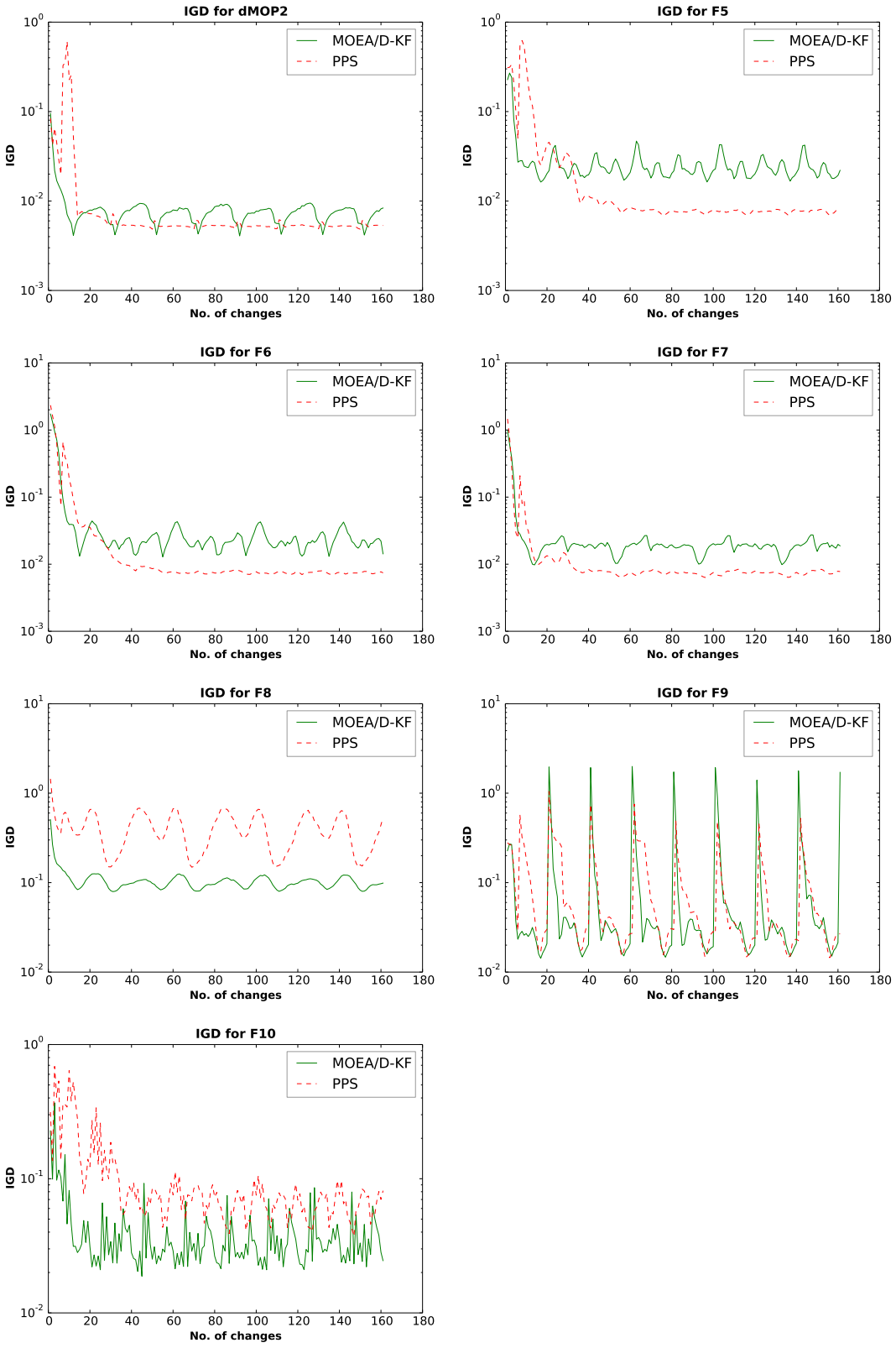


Figure 3.5: IGD Trend comparison of MOEA/D-KF and PPS algorithms over number of changes for 30 runs : dMOP2, F5 - F10

in EDAs may be utilized simultaneously. The Autoregressive time series used in PPS is able to provide more stable performance than the Kalman Filter in MOEA/D-KF. F8 is a 3 objective problem whose Pareto Optimal Front remains the same. For F8, MOEA/D-KF is able to outperform PPS throughout the 160 changes and the MIGD value obtained is significantly better (Table 3.3).

3.5.3 Results on F9 and F10

In all the previously discussed problems, the environment changes smoothly from one time instant to the next and the geometric shapes of two consecutive Pareto Optimal fronts/sets are similar to each other. Two more complicated problems, F9 and F10, are proposed in [21], to further test the performance of dynamic MOEAs. In F9, the pareto set jumps from one area to another occasionally. The geometric shapes of consecutive POFs are completely different from each other in F10. The proposed strategy, MOEA/D-KF makes the assumption of a linear dynamical process in the Kalman filter formulation which is violated in F9. Therefore, it is not surprising that the performance of MOEA/D-KF is not on par with that of PPS. Nevertheless, it must be noted that the IGD values of PPS also follows a pattern similar to that of MOEA/D-KF, but are less affected by the jumps. In F10, MOEA/D-KF seems to perform better than PPS. This may be due to the fact that the Kalman Filter does not make any assumptions about the shape of the pareto fronts, while the PPS does so in the estimation of successive manifolds.

3.5.4 Parameter Sensitivity

The model and parameters of Kalman filter may have a significant impact on MOEA/D-KF algorithm's performance. The linear dynamic model affects the tracking performance of the Kalman filter when there is a deviation

from such an assumption. The Q and R matrices may have substantial effect on the prediction performance and thereby affecting the IGD values obtainable by MOEA/D-KF.

Q is defined as the process noise covariance matrix and quantitatively denotes the noise present in the 'process' that the Kalman filter tries to estimate. Similarly, R is the measurement noise covariance matrix and denotes the amount of noise present in the 'measurements' that are passed to the Kalman filter. The two matrices are assigned as diagonal matrices having diagonal element values of q and r, respectively. The results presented in Sections 3.4.4 and 3.4.5 are based on a single fixed pair of $(q, r) = (0.04, 0.01)$ for all the test benchmark problems. In this section, the q and r values are varied from 0.01 to 0.1 in steps of 0.01, resulting in $100(10 \times 10)$ combinations of values.

Table 3.5: Tuning of Q and R matrices of Kalman Filter

Problem	**lowest MIGD**	**highest MIGD**	**Range of MIGD**
FDA1	0.39	0.5	0.11
FDA2	0.37	0.4	0.03
FDA3	1.08	3.64	2.56
FDA4	3.77	4.87	1.1
FDA5	5.94	12.42	6.48
dMOP1	0.63	0.76	0.13
dMOP2	0.45	0.68	0.23
F5	2.19	7.75	5.56
F6	4.43	7.8	3.37
F7	3.73	5.25	1.52
F8	4.81	5.93	1.12
F9	4.94	31.97	27.03
F10	5.19	21.73	16.54

Table 3.5 shows the lowest and highest MIGD values obtained by utilizing the 100 combinations of q and r values. Range of MIGD is given to

demonstrate the impact of values on the performance. For less complicated problems the range of MIGD is quite low. In these problems, the linear dynamical assumption of Kalman filter is not affected such as in FDA1, FDA2, dMOP1, dMOP2. In the case of F9 and F10, the range of MIGD is substantially higher than those of the other problems. In F9, the pareto set jumps from one area to another implying that the 'process' is not as assumed by the Kalman Filter model occasionally and therefore, subsequently causing fluctuations in performance.

3.5.5 Influence of frequency of change

Problem difficulty increases substantially with increase in the occurrence of change as the DMOEA has to frequently adapt the solutions to the moving optima. In this section, the frequency of change parameter is varied in steps of 10 and the box-plots of IGD values are obtained for MOEA/D-KF, PPS and RND methods (Figures 3.6 and 3.7). It is observed that the Kalman Filter predictions result in better IGD values than the autoregressive filter predictions in PPS, even when the problem changes every 10 generations.

RND and MOEA/D-KF have the same underlying MOEA and the difference is in the tracking mechanism. It is observed that the Kalman Filter improves the performance of MOEA/D in most of the problems. In the case of complicated problems such as F5, F6 and F7, MOEA/D-KF's performance is comparable to that of PPS in spite of the advantage of EDA in PPS and is significantly better than that of RND. It is observed that in the case of the more complicated problems of F9 and F10, increasing the frequency of change has resulted in the Kalman Filter assisted MOEA/D outperforming the PPS strategy.

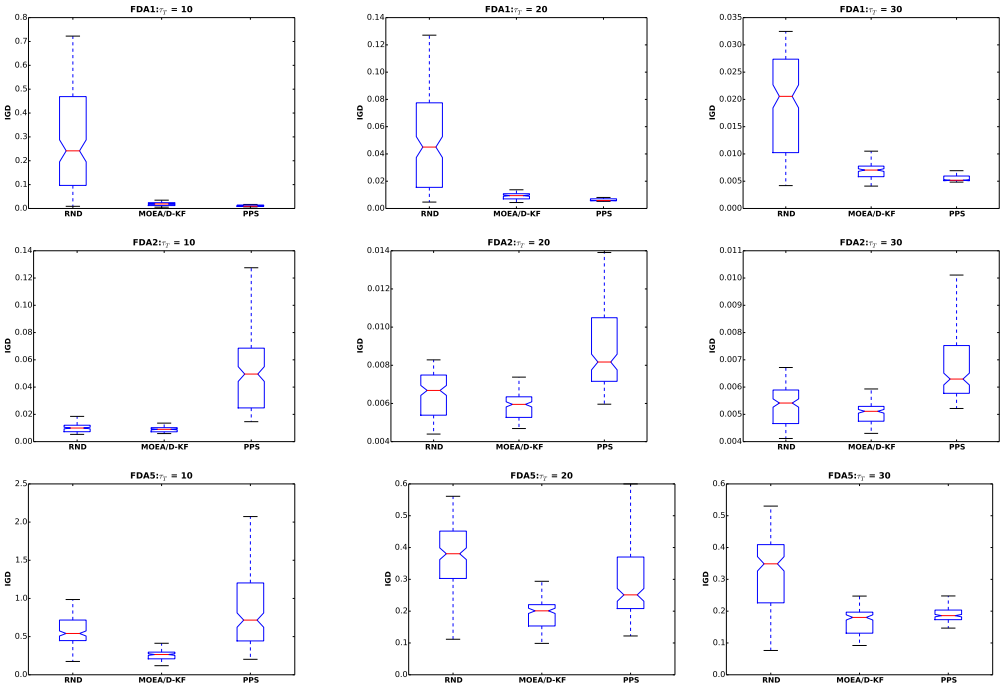


Figure 3.6: Influence of frequency of change on FDA1, FDA2 and FDA5 problems. The figures show the box plot of IGD values for RND, MOEA/D-KF and PPS algorithms for the 3 benchmark problems for $\tau_T = 10$, 20 and 30. Each row is for a particular benchmark problem and τ_T value varies from 10 to 30.

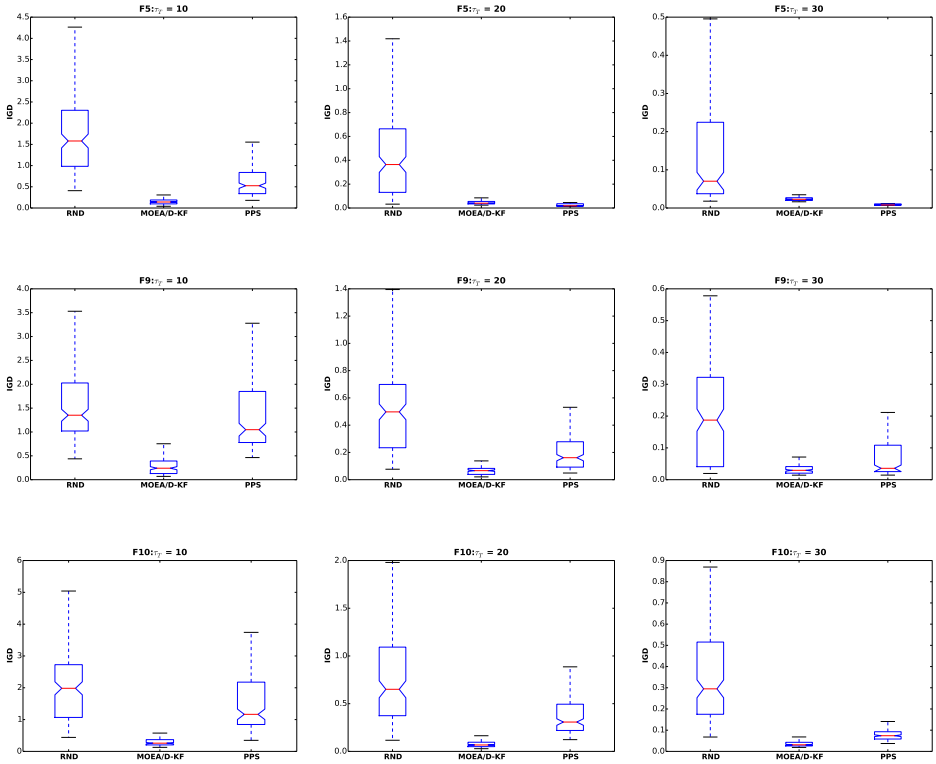


Figure 3.7: Influence of frequency of change on F5, F9 and F10 problems. The figures show the box plot of IGD values for RND, MOEA/D-KF and PPS algorithms for the 3 benchmark problems for $\tau_T = 10$, 20 and 30. Each row is for a particular benchmark problem and τ_T value varies from 10 to 30.

3.6 Chapter Conclusion

A novel Dynamic Multiobjective Evolutionary Algorithm using Kalman Filter predictions in the decision space is proposed. It is built on the MOEA/D framework and the Linear Discrete Time Kalman Filter is used to estimate the subsequent optimal values of decision variables. Change detection is performed through sentry particles and a scoring scheme is devised to hybridize the Kalman Filter with the random reinitialization method. Experimental results demonstrate that the proposed algorithm shows significantly improved performances over a number of test benchmark problems.

The 3by3 variant of the proposed model is particularly recommended as second order change is also taken into account and when hybridized with RND using the scoring scheme, the method shows superior performance over the other Kalman Filter prediction model variants as well as other DMOEAs. The Kalman Filter prediction model does not require any learning time and starts to provide reasonable predictions from the start of the run and corrects itself from subsequent measurements of the decision variables from the Evolutionary Algorithm. In problems with linkages between decision variables, the underlying EA plays a significant role in the optimization and it may be useful to implement the Kalman Filter prediction model in an Estimation of Distribution Algorithm to get even better dynamic optimization performance. Problems in which the linear dynamical process assumption is violated, the proposed algorithm's performance is not optimal. However, the Kalman Filter does not make any assumptions about the shape or structure of consecutive Pareto Optimal Fronts resulting in better performance in such problems. Parameter sensitivity of the model on Q and R matrices is studied and it can be observed that for problems with more non-linear movements the values need to be carefully tuned to get better performance. In spite of the aforementioned shortcomings, with increasing problem difficulty the Kalman Filter based DMOEA is able to outperform the other methods.

In the following chapter, a non-linear prediction method is explored. While MOEA/D-KF and its variants provide many advantages, a non-linear prediction method such as Support Vector Regression may be able to tackle the more complicated problems with better efficacy.

Chapter 4

Data-driven Accelerated Convergence in Evolutionary Dynamic Multiobjective Optimization

4.1 Introduction

The Support Vector algorithm [74] [75] is a nonlinear generalization of the *Generalized Portrait algorithm* developed in Russia in the 1960s [76]. Initially, research on Support Vector Machines (SVM) was focused on Optical Character Recognition (OCR) and object recognition tasks. However, soon after, excellent performances were obtained in regression and time series prediction applications as well [77]. Since then, SVM has been an active field of research. However, little research has been done on using Support Vector Machines with Evolutionary Algorithms for solving Dynamic Multiobjective Optimization Problems.

Evolutionary algorithms have been traditionally used to solve a wide

variety of Multiobjective Optimization Problems (MOPs) [1]. MOPs consist of two or more conflicting objectives that need to be solved simultaneously. As a result, there is no single solution but a number of trade-off solutions. Evolutionary algorithms work with a population of solutions at any single time and this poses a unique advantage for them as they can produce a number of pareto optimal solutions in a single run [27–29]. While there has been extensive research to solve MOPs in the past few decades, another class of challenging problems have received only tepid interest. Most real-world problems are multiobjective in nature, but they are also filled with uncertainties and dynamics [38] [37]. Dynamic Multiobjective Optimization Problems (DMOPs) consist of MOPs which change with time, due to changes in the objective or decision space and constraints [2].

Evolutionary algorithms exhibit competitive performance in solving static MOPs. However, they cannot solve DMOPs in a standalone manner. One of the important drawbacks of Multiobjective Evolutionary Algorithms (MOEAs) is that they require significant amount of time to converge to the optimal solutions [45]. This is an important issue in DMOPs as the problems change with time and tracking the changing solutions takes up a high priority. In this context, algorithms that accelerate convergence of the solutions towards the pareto optimal front would be highly preferred [17, 44, 46]. Various kinds of methods have been proposed to solve dynamic problems in both single-objective and multiobjective scenarios. However, prediction based methods stand out as their contributions can prove to be of high efficacy. Prediction based methods can help identify the pattern exhibited by the time-varying solutions and assist the Evolutionary Algorithm in converging to the Optimal solutions faster than if they were left to work on their own.

A Kalman Filter based Dynamic Multiobjective Evolutionary Algorithm

(DMOEA) was proposed and shown to have competitive dynamic optimization performance compared to other existing algorithms [78]. However, when assumptions of the Linear Discrete Kalman Filter are violated, the performance was not up to the par. Nonlinear formulations of the Kalman filter are available by means of the Extended Kalman Filter(EKF) [58], Unscented Kalman Filter(UKF) [57], and numerous other variants. However, the state transition and observation matrices which are required in the prediction and update step are formed by Jacobian (matrix containing partial derivatives of f with respect to x) of the nonlinear functions, which are not directly available to us. Thus, we need to resort to other prediction techniques.

When considering real-world dynamical systems, the underlying system models are complex and not known *a priori*. Accurate and unbiased estimation of such systems cannot be achieved using linear techniques resulting in the need for more advanced time series prediction algorithms. The machine learning approach of Support Vector Machines(SVM) [79] [80] has been extensively applied and it is found to be able to accurately forecast time series data even when the underlying system processes are not defined a priori [81]. It is not model dependent and can outperform traditional Neural Networks and also has the advantage of small number of free parameters. Unlike the Kalman filter and Autoregressive time series models, SVM is not dependent on linear, stationary processes, can obtain guaranteed convergence and is computationally efficient.

Considering the various advantages of using SVM for prediction, it is proposed to build a SVM based prediction model for solving DMOPs. When using Kalman Filter, it is a state estimation problem to predict the subsequent optimal solutions, while a time series prediction problem is modeled when using Support Vector Regression to assist the Evolutionary

Algorithm in solving the DMOPs. The DMOPs considered in this work change in discrete periods of time followed by a stasis, where there is no change in the optimal solutions. In this method, a time series is formed by the near-optimal solutions obtained by the Evolutionary Algorithm in previous changes. Support Vector Machines are used in tandem with the Evolutionary Algorithm to predict new solutions for future generations from the time series, when a change in the environment is detected. LibSVM [82], an open-source library software for various SVM formulations is utilized to build the prediction model.

The rest of the chapter is organized as follows. Section 4.2 provides background on Support Vector Machines. A review of existing work using SVM for time series prediction in various domains is discussed in Section 4.3. Section 4.4 presents the proposed algorithm. The underlying MOEA, MOEA/D-DE and the Support Vector Regression based prediction method are also elaborated. Section 4.5 describes the empirical study. Section 4.6 provides the results and discussion. Section 4.7 elaborates on the analysis. Section 4.8 concludes the work and potential future research directions are highlighted.

4.2 Background

Support Vector Machine [76], a widely accepted novel artificial intelligence-based method developed from statistical learning theory, is used in this study to predict the pareto optimal solutions after a change in the problem. The SVM, which is based on structural risk minimization (SRM) principle, theoretically minimizes the expected error of a learning machine and thereby reduces the problem of over-fitting [83]. The SVM has been proven to be a robust and competent algorithm for both classification [84] [85] and

regression [86] [87] in many disciplines.

The application of SVMs to general regression analysis case is called Support Vector Regression(SVR). Support Vector Machines and Support Vector Regression are based on statistical learning theory, or VC theory. Support Vector Regression, or SVR, is the methodology by which a function is estimated using observed data which in turn trains the SVM [81]. This is different from traditional time series prediction methodologies as there is no model in the strict sense - the data drives the prediction. SVR uses a kernel function which provides the capability of mapping nonlinear data into feature spaces that are essentially linear. Following this the optimization process can be similar to the linear case.

There are a few formulations of SVR, the most common types are $\epsilon - SVR$ and $\nu - SVR$. Let the given training data be $(x_1, y_1), \ldots, (x_l, y_l) \subset X * \mathbb{R}$, where X denotes the space of the input patterns and y_i is a target output. For nonlinear models, an implicit mapping is done via kernels. Kernel function can be represented by $k(x, x') =< \Phi(x), \Phi(x') >$, where k is the kernel function and Φ is the map.

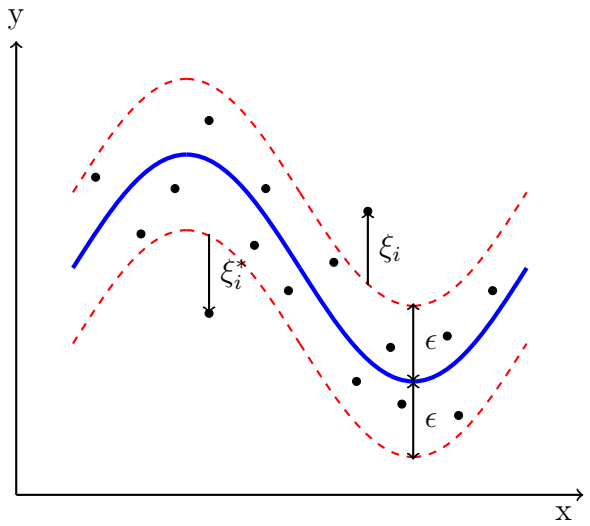


Figure 4.1: Support Vector Regression formulation

ν-Support Vector optimization problem can be stated as,

$$
\begin{aligned}
& min \frac{1}{2} w^T w + C(\nu\epsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)) \\
& (w^T\phi(x_i) + b) - y_i \leq \epsilon + \xi_i, \\
& y_i - (w^T\phi(x_i) + b) \leq \epsilon + \xi_i^*, \\
& \xi_i, \xi_i^* \geq 0, i = 1, ..., l, \epsilon \geq 0. \\
& \text{subject to } \sum_{i=1}^{l}(\alpha_i + \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]
\end{aligned}
\tag{4.1}
$$

In $\nu-$ SVR, the goal is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all the training data. At each point x_i, an error of ϵ is permissible. The ϵ-insensitive loss function (Fig 4.2) means that if $(w^T\phi(x) + b)$ is in the range of $y \pm \epsilon$, no loss is considered. Here, $0 \leq \nu \leq 1$, C is the regularization parameter, and training vectors, x_i are mapped into a higher dimensional space by the function ϕ. Any error above ϵ is captured through the slack variables $\xi_i^{(*)}$, which are penalized in the objective function via the regularization constant C. α_i and α_i^* are multipliers used to solve the Support vector optimization problem using the Lagrangian formulation.

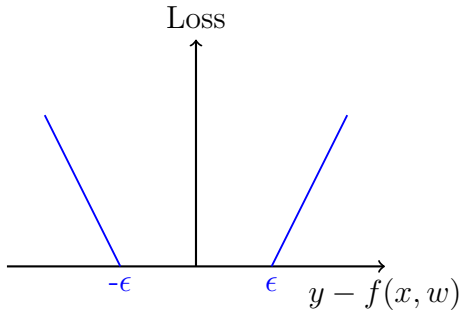


Figure 4.2: ϵ-insensitive loss function

ν-SVR was introduced by [88] since it is difficult to select appropriate ϵ value in $\epsilon - SVR$ as ϵ can range from 0 to ∞. ν can take values only

between 0 and 1, which proves advantageous in parameter selection. ν controls the number of support vectors and training errors. It was proved that ν is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors. Due to ease of choosing appropriate value of ν, $\nu - SVR$ is used in the proposed algorithm.

The key parameters used in SVR model are:

- gamma - defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close by'.
- C - defines balance between training error and the simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at reducing the training error to as close to zero as possible.
- ν - proportion of maximum deviation of the predicted value from the obtained targets.

4.3 Related Work

4.3.1 Time Series Prediction Using Support Vector Machines: A Survey

Support Vector Regression forms a key component of the proposed model. Support Vector Regression is proposed to be used as the prediction model for estimating the time series formed by dynamic multiobjective optimization problems. [81] presents a survey of the applications of Support Vector Machines (SVMs) in time series prediction. It was a good starting point for understanding the existing work in the field of Support Vector Regression.

SVMs provide a method for predicting and forecasting time series for numerous applications including financial market and time series forecasting [89] [90] [91] [80] [86] [87], weather and environmental parameter estimation [92] [93], electrical utility loading prediction [94] [95] [96] [97], machine reliability forecasting [98], various signal processing and control system applications [99] [100], and several other applications. The fundamental reason for considering SVR as an approach for time series prediction in these applications is the nonlinear aspect of the prediction problem. Traditional model-based techniques do not perform as well as SVR in predicting time-series generated from nonlinear systems. The main challenge of using SVR is that the performance of SVR is highly dependent on the design choices made by the designer such as selection of kernel function [101], hyperparameters, etc.

Application of Support Vector Machines in Financial Time Series Forecasting

SVR time series prediction has been extensively researched in the area of financial market prediction. Based on a survey conducted in the year 2009, over 21 research papers had been published in this field [81]. Financial time series are said to be inherently noisy, non-stationary and deterministically chaotic. The complete information about the past behavior of financial market is unavailable, hence making the data noisy. The distribution of financial series changes over time making the problem non-stationary. The term deterministically chaotic means that the financial time series are random in short term but deterministic in the long run.

[102] observes that SVMs forecast significantly better than neural networks trained using backpropagation. This can be accounted to the fact that SVR has comparatively fewer free parameters, guarantees convergence,

and minimizes the upper bound of the generalization error. The study also emphasizes the proper selection of the free parameters. Improper selection of the parameters can cause either over-fitting or under-fitting of the training data. It is hence important to develop a structured way of selecting optimum parameters of SVMs. The proposed algorithm of using SVR in this work is hence in line with the recommendation of [102]. The rationale behind choosing optimum parameters for SVM used in this work is explained in section 4.4.

Load Forecasting Using Support Vector Machines

Electricity load forecasting is one of the predominant areas in which SVR has been used as the prediction model. Forecasting of electrical power consumption demands by consumers is a nonlinear prediction problem. [103] proposed using SVR approach for EUNITE Network Competition which entails the prediction of daily maximal electrical load. The electricity load forms a periodic time series. This is due to the seasonal variation of consumer electricity demand, lesser usage during major holidays, and the impact of weather on electricity demand. The SVR model was built using several attributes such as the day of the week, whether it is a holiday, etc. Based on the SVR model built, the maximum load was predicted. In order to select the proper values of SVR parameters, the training data was divided into two sets. One of the sets was used to train the model while the other was used to evaluate the model. [103] was the winning approach for the EUNITE Network competition.

4.4 Algorithm Design

This section elaborates on the design and functionality of the proposed algorithm. There are 3 main sub-parts to the algorithm which are expanded in the following subsections. The goal of the proposed algorithm is to solve Dynamic Multiobjective Optimization problems wherein the solutions change with time. To accelerate the convergence of Evolutionary Algorithms, MOEA/D-DE (refer to subsection 3.3.1) in this case, we propose to use Support Vector Regression (refer to subsection 4.4.3) as the prediction technique to find the optimal solutions after a change in the problem. To do so, we need a change detection function, which is explained in subsection 4.4.2.

4.4.1 Multiobjective Evolutionary Algorithm with Decomposition based on Differential Evolution

The Support Vector Regression based prediction model proposed is built on the structure of Multiobjective Evolutionary Algorithm with Decomposition based on Differential Evolution (MOEA/D-DE) [29]. MOEA/D-DE has received significant attention due to its good optimization performance in solving continuous multiobjective optimization problems with relatively fast convergence and diverse spread. The algorithm decomposes a problem into several sub-problems and simultaneously optimizes them using neighborhood relations. The neighborhood relations are defined based on the distances among their weight vectors. The decomposition is performed using classical approaches, such as the Tchebycheff approach or the weighted sum approach. The Tchebycheff approach is used in this work due to its simplicity and decent optimization performance.

4.4.2 Change Detection Function

The relationship of MOEA/D-DE with the SVR prediction model is shown in Figure 4.3.

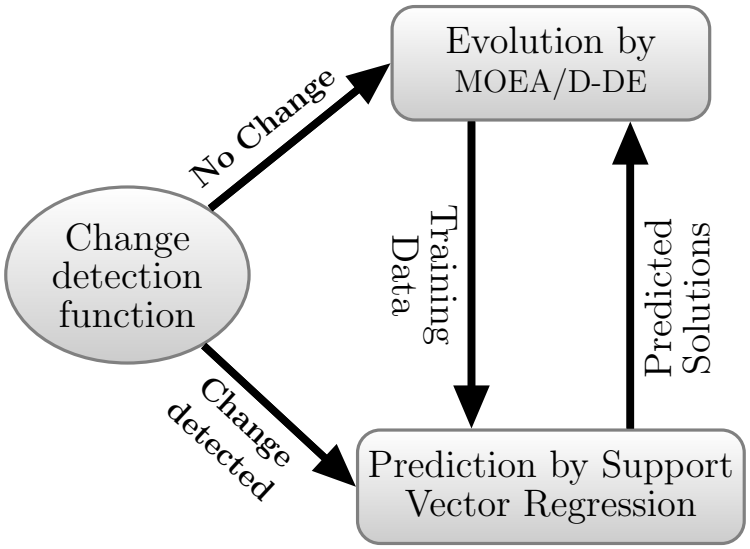


Figure 4.3: Relationship of EA with SVR prediction model

When there is no change detected, MOEA/D-DE takes control and the population evolves accordingly. Otherwise, the SVR Filter prediction model directs the search for Pareto optimal solutions in the decision space. Data for training the SVR Prediction model is obtained from MOEA/D-DE and the solutions predicted by the SVR model are provided to MOEA/D-DE.

A change detection function is needed to combine the prediction model with the MOEA/D-DE algorithm. The DMOPs considered in this work consist of discrete changes in the optimization problem followed by a stasis period when there is no change. Assuming that there is no noise in objective functions evaluation, some individuals are randomly selected as detectors and their objective values are stored in the system. At the beginning of each generation, the detectors' objective values are recalculated and compared with the previously stored values. A mismatch in the objective values suggests that a change in the problem has occurred caused by moving POS

or POF landscape.

4.4.3 Support Vector Regression based Prediction model

The proposed prediction model consists of a number of components which are discussed in detail in this subsection.

Data Formulation

A change in the problem occurs every τ_t number of generations. In order to aid the Evolutionary Algorithm in searching effectively for optimal solutions, SVR predicts the changed optimal values of decision variables. It is assumed that the Evolutionary Algorithm reaches the optima prior to the subsequent change. For instance, in Figure 4.4, the problem changes at t_1, t_2 and t_3. The optimum has to be found by $t_3 - 1$ since the problem changes at t_3. To aid this, SVR needs to predict for $t_3 - 1$ using values of individuals at $t_1 - 1$ and $t_2 - 1$.

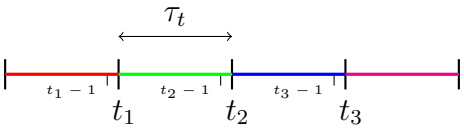


Figure 4.4: Change Occurrence

The SVR model needs to be trained for a number of changes, n, before it can start predicting. This is referred to as the training window size. Initially, the SVR model waits for n number of changes to occur in the problem, before it can come into play. Subsequently, the training window is moved forward in a fashion akin to moving window concept, wherein SVR needs to be trained with values from previous n changes, before it can predict for subsequent change. This is illustrated in Figure 4.5, where SVR is trained with values from t_{i-n} to t_i changes, to predict the value of

decision variable for change t_{i+1}. In this example, the training window size is 5.

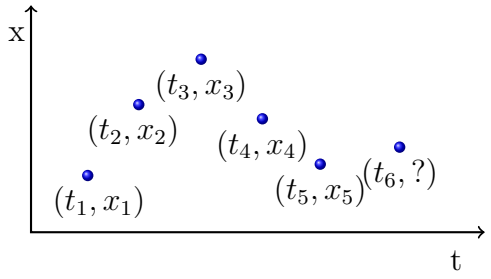


Figure 4.5: Time Series Formulation

Data Preprocessing

Feature scaling or normalization is an important step when using machine learning techniques. It becomes even more so in the case of SVM. Many studies have shown that feature scaling affects the overall performance to a good extent [82] [104]. In this work, the decision variable values are the target. For most of the problems, their boundary values are in the range of -1 to $+1$. However, the input consists of the generation number of the evolutionary algorithm, which ranges from 1 to 4835 in this work. Therefore, the generation number is scaled to a range from 0 to $+8$, obtained through empirical analysis.

Mean Square Error

A mechanism needs to be established for choosing appropriate SVR parameters for good predictions. Mean Square Error of predicted solutions from subsequently reached near optimal solutions can be used as a performance metric to assess the success level of the prediction method. Based on the current and past data about decision variables, SVR predicts the value of decision variables following a change. The Evolutionary Algorithm then searches the neighborhood of the predicted decision variables to find the

new optima. Hence the performance of SVR prediction can be measured by comparing the values predicted by SVR with those that are subsequently obtained by the Evolutionary Algorithm. The mean square error between the solutions predicted by SVR immediately following a change, and the value of decision variables found by the Evolutionary Algorithm just before the subsequent change is a key indicator of the performance of SVR.

The mean square error for each decision variable is calculated using the formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y_i} - Y_i)^2 \tag{4.2}$$

where n is the population size, $\hat{Y_i}$ is the value predicted by SVR and the Y_i is the value obtained from MOEA/D-DE algorithm.

SVR Parameter Selection

- Kernel Type - SVR uses a kernel function to map nonlinear data into feature spaces that are essentially linear. There are three common types of SVR kernels: linear, polynomial and radial basis function (RBF). These can be mathematically represented as,

  Linear: $< x, x' >$

  Polynomial: $(\gamma < x, x' > +r)^d$

  RBF: $exp(-\gamma * |u - v|^2)$

  where x represents a point in the training data, d represents the degree of polynomial, r represents the coefficient, γ represents the influence of a single training example.

  Preliminary testing of SVR was done on simple functions and RBF kernel type was found to be most effective for simple non linear functions as seen in Figure 4.6.

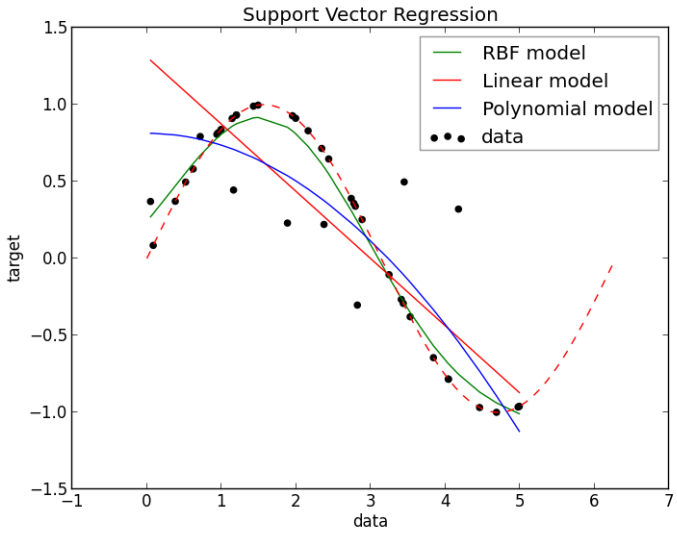


Figure 4.6: Comparison of kernel types

- C and gamma values - Preliminary testing for the SVR prediction model was done on basic functions by varying data to noise ratio, *gamma*, *nu* and C values. Performance of SVR heavily depends on the chosen value of its free parameters, mentioned in section 3.2.2.

Table 4.1: Possible range for C and Gamma

Range	C	Gamma
Minimum	1e-4	1e-4
Maximum	1e4	1e4
Step size	factor of 10	factor of 10

Grid search was performed on a range of C, *gamma* and *nu* values. SVR performance was analyzed for the values mentioned in table 4.1 [82]. The *gamma* parameter can be considered as the inverse of the radius of influence of samples selected by the model as support vectors. If *gamma* is too large, it results in over-fitting and no amount of regularization with C will be able to prevent over-fitting. If *gamma* is too low, the model becomes very simplistic and will not be able to capture the complexity of the data. The resulting model will behave

similar to a linear model. Hence the intermediate values showed promising results. Upon analysis of the SVR results obtained from

Table 4.2: Grid search values

C	1	1	1	10	10	10	100	100	100
Gamma	0.1	1	10	0.1	1	10	0.1	1	10

the possible combinations of C and *gamma* mentioned in Table 4.1, exhaustive grid search was performance on the values mentioned in Table 4.2. The population of the Evolutionary Algorithm was divided equally between each combination of C and *gamma* values. After each change, SVR was used for prediction. Mean square error was calculated as proposed in section 4.4.3. For each decision variable, the SVR parameters used on the population set with the least mean square were chosen. This process is repeated until the Evolutionary Algorithm's stopping criteria is reached. A vast variation is observed in the SVR parameters chosen for each problem. The various steps of the proposed SVR parameter selection mechanism used to solve DMOP are shown in Algorithm 4.1 for clarity.

The above mentioned method performs the mean square error calculation in the decision space by comparing the values of the decision variables. An alternative method of calculating the mean square error on the objective space was implemented for comparison. The parameters where selected based on the mean square error in the objective values of the individuals in the population rather than the decision variables. The performance of SVR in this alternate method was not on par with the proposed method. This is because the dimensionality of the decision space is typically more than that of the objective space. The absolute value of mean square error in decision space is of a higher magnitude as compared to that in the objective

Algorithm 4.1 SVR Parameter Selection

Require:

MOP

A stopping criterion

N: Population size

P: the number of parameter combinations

SVR parameters

T: Training window

Step 1 ➢ Initialization:

1. Select the range and step size of SVR parameters.
2. Generate a grid of size P with the possible combinations of SVR parameters.
3. Divide the individuals in the population equally into P segments.
4. Each segment is assigned a set of parameters

Step 2 ➢ Parameter Selection:

1. Save the solutions associated with each individual before the change.
2. If number of changes is less than training window size T, randomly reinitialize solutions and increment number of changes counter, and go to **Step 2.1**. Else if number of changes is equal to T, go to **Step 2.5**. Otherwise, go to **Step 2.3**.
3. Calculate the mean square error of each population segment according to equation below,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

   where n is the population segment size, $\hat{Y}_i$ is the value predicted by SVR and the Y_i is the value obtained from MOEA/D-DE algorithm.
4. Choose and save the SVR parameters associated with the population segment resulting in the least mean square error.
5. Train SVR using the solutions just before the change and the SVR parameter values associated with each individual.
6. Predict new solutions associated with each individual in the decision space.
7. Save the predicted values associated with each individual after the change.
8. Increment number of changes counter.

Step 3 ➢ Stopping Criteria: If stopping criteria is satisfied, then stop and output the selected SVR parameters.
Otherwise, go to **Step 2**.

space since the mean square error in each dimension gets added. Due to the magnification of error in decision space, mean square error in the decision space is a more accurate indicator of the performance of SVR.

- nu - SVR performance was tested with values of nu ranging from 0.2 to 0.8. It was observed that $nu = 0.5$ gave the best results. This can be accounted for by the fact that a smaller nu results in over-fitting of the SVR model. The training input noise level is quite high. Hence setting a small error margin would adversely affect the performance of SVR.

- Training window size - SVR training window refers to the number of training instances used to build the SVR prediction model. The SVR training window for each problem was tuned based on the performance optimization of SVR. A training window too large would require larger number of changes to be completed before SVR can be used to predict. A training window too small would not be sufficient to capture complex time series formulation.

  The performance of SVR was tested on the following training windows: 10, 15, 20, 25, 30, 35, 40, 45, 50. Subsequently, a training window size of 35 was chosen for use.

Boundary Correction

The SVR prediction model is provided with a time series of the previous near-optimal solutions obtained by the evolutionary algorithm. There are boundary conditions for the decision variables in the search space that need to be adhered to in the optimization process, as defined in 1.1. In case the solutions predicted by SVR violate the boundary condition and lie outside

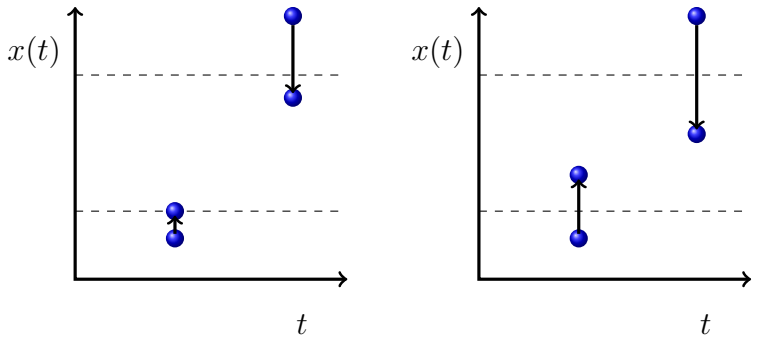
the search space, the new solutions are corrected to fit inside the boundary condition (Refer to Figure 4.7). Two types of boundary corrections were implemented and analyzed [19].

- Clamping approach

  In clamping approach (illustrated in Figure 4.7a), if the predicted solutions violated a specific boundary condition, they are placed on or close to the violated boundary of the search space. It can be mathematically represented as :

$$\begin{aligned} &\text{if } x(t+1) > x_{max} \text{ then } x(t+1) = x_{max} - \epsilon \\ &\text{if } x(t+1) < x_{min} \text{ then } x(t+1) = x_{min} \end{aligned} \tag{4.3}$$

  ϵ is a very small positive number.



(a) Clamping approach (b) Deflection approach

Figure 4.7: Boundary Correction Approaches

- Deflection approach

  In deflection approach (illustrated in Figure 4.7b), if the predicted solution is outside the search range, the difference between the solution

and the boundary value is subtracted from the boundary value.

$$\begin{aligned} &\text{if } x(t+1) > x_{max} \\ &\text{then } x(t+1) = x_{max} - (x(t+1) - x_{max}) \\ &\text{if } x(t+1) < x_{min} \\ &\text{then } x(t+1) = x_{min} + (x_{min} - x(t+1)) \end{aligned} \tag{4.4}$$

where x, x_{min} and x_{max} are any dimension of the decision variable, and its corresponding minimum and maximum boundary constraints respectively.

It was found that the deflection boundary correction was more effective for correcting solutions predicted by SVR.

LibSVM [82], an open source library for various SVM formulations was used to build the prediction model. If there is a change in the detector's objective values, the proposed prediction method comes into play. For a predetermined number of changes, RND method is used and SVR is trained. For the subsequent changes, SVR is used to predict the locations of the solutions in the decision space after the change. The solutions just before the change occurs are considered to be true values (corrupted by Gaussian noise) of *a priori* estimates. They are used to train the prediction model accordingly. Thereafter, new solutions associated with each individual in the decision space are predicted using SVR. The new predicted solutions are checked for boundary conditions and are corrected to fit inside the boundary if required. The reference points and sub-problems are updated with these new solutions and this completes one generation when a change is detected in the system. Pseudo-code of the proposed algorithm is provided in Algorithm 4.2 for clarity.

Algorithm 4.2 MOEA/D-DE with SVR for Dynamic Multiobjective Optimization

Require:

MOP, A stopping criterion
N: Population size
P: the number of subproblems considered in MOEA/D
A uniform spread of N weight vectors: $\lambda^1, \lambda^2, \ldots, \lambda^P$
T: neighbourhood size
SVR Parameters

Ensure:

Approximated POF $\{f^1, \ldots, f^N\}$, Approximated POS $\{x^1, \ldots, x^N\}$

Step 1 ➢ Initialization:

1. Generate evenly spread weight vectors. Initialize the neighborhood of each vector by finding its T closest weight vectors in terms of Euclidean distance.
2. Generate an initial population, $\mathbf{x}^1, \ldots, \mathbf{x}^N$ by uniform random initialization within the decision space. Evaluate objective function values of each solution and set $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i)$.
3. Load SVR parameters obtained from Parameter Selection mechanism described in Algorithm 1.
4. Initialize ideal vector by setting

$$z_k = \min_{j=1,\ldots,N} f_k^j$$

$$\text{where } k = 1. \ldots, m$$

5. Randomly initialize a set of detector individuals within the decision space for change detection.

Step 2 ➢ Update:

1. **Change Detection:** Evaluate the objective function values of the detector individuals and check whether they have changed to indicate a change in the dynamic multiobjective problem. If change is detected go to **Step 2.2**. Otherwise, go to **Step 2.3**.
2. **SVR prediction**
   (a) Save population.
   If number of changes is lesser than training window size, randomly reinitialize the population using RND algorithm and go to **Step 2.1**. Otherwise, go to **Step 2.2.2**.
   (b) Formulate training and testing file to be provided to LibSVM.
   (c) Train SVR using the solutions just before the change and the best SVR parameters obtained from parameter selection mechanism.
   (d) Predict new solutions associated with each individual in the decision space.
   (e) Repair solutions according to deflection approach of boundary correction if necessary.
3. Reproduction: Mating selection, Differential Evolution, update neighborhood and the ideal vector.

Step 3 ➢ Stopping Criteria: If stopping criteria is satisfied, then stop and output the final population and their corresponding values in the objective space. Otherwise, go to **Step 2**.

4.5 Empirical Study

4.5.1 Benchmark problems

The proposed algorithm is tested on problems from 3 test benchmark suites - FDA [2], dMOP [26], and F [21]. The FDA benchmark suite is commonly used in the performance evaluation of DMO algorithms. The dMOP benchmark problems are an extension of the FDA benchmark suite to test further performance characteristics of DMO algorithms such as learning that the POS/POF does not change. The problem suite proposed in [21] is very recent and consists of 10 problems which are partly adopted from the above 2 benchmark suites. Nevertheless, they have also proposed 6 new test instances in which non-linear linkages between the decision variables are considered and benchmark problems with sharp and irregular environments are also constructed.

4.5.2 Parameter Settings

The proposed SVR prediction model is implemented in MOEA/D-DE, which is referred as MOEA/D-SVR for simplicity. The parameter settings for the experiments of the various test benchmark suites are tabulated in Table 4.3. The number of decision variables is set as 20 for all the test problems. All the test problems considered in this study are 2-objective problems, hence a population size of 100 is used. The various parameters for MOEA/D-DE are implemented as guided in [29]. 10 detector individuals are utilized for change detection purpose.

A random reinitialization method (RND) is implemented for baseline performance comparison of MOEA/D-SVR. In this algorithm, instead of the SVR prediction model, 20% of the population is randomly reinitialized after a change is detected.

The SVR prediction model parameters are selected as described in Section 4.4.3.

Table 4.3: Experiment Settings

Number of decision variables, n	20 for all test problems
Population size	100 for 2 objective problems, 200 for 3 objective problems.
Neighborhood	Size: 20.
Probability that parents are selected from the neighborhood	0.9
Decomposition method	Tchebycheff
Differential Evolution	CR = 1.0 and F = 0.5
Polynomial Mutation	$\eta = 20$, $p_m = 1/n$.
Number of detectors	10
Percentage for RND model	20%
Dynamic Setting	Frequency of change τ_T: 30, Severity of change n_t: 5
Number of changes	161
Number of generations	4835
Number of runs	10

4.5.3 Performance Metrics

MIGD, Mean Inverted Generational Distance, elaborated in Chapter 3, Section 3.4.3 is employed as performance metric for performance comparison. A lower value of MIGD indicates better dynamic optimization performance.

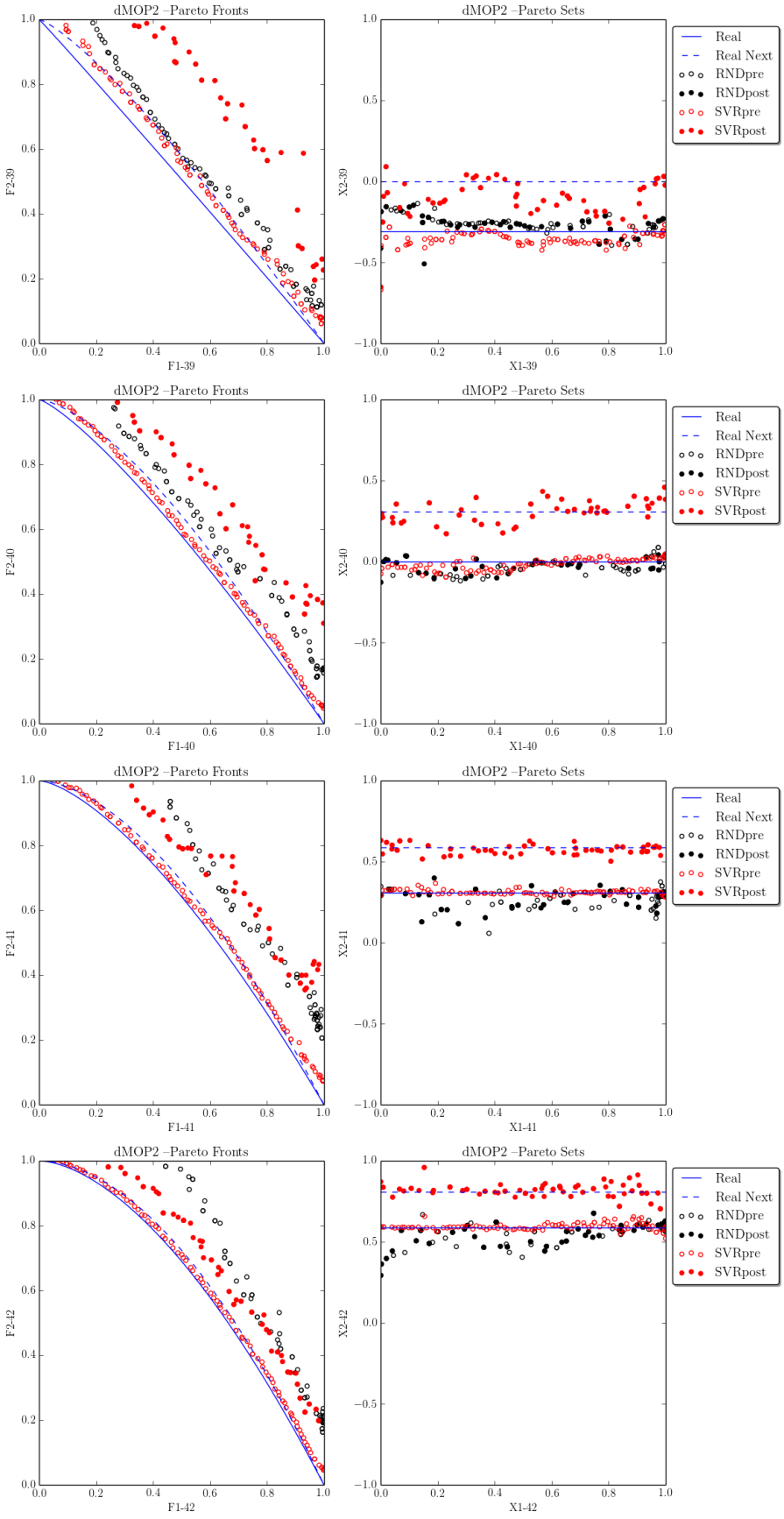


Figure 4.8: Visualization of SVR prediction performance in dMOP2

Table 4.4: Performance Comparison with other DMOEAs

Problem	RND	MOEAD/D-KF	MOEA/D-SVR
FDA1	0.054629 ± 0.041(+)	**0.010603 ± 0.004**	0.015078 ± 0.010(+)
FDA2	0.006180 ± 0.001(+)	**0.005471 ± 0.001**	0.005767 ± 0.001(+)
dMOP1	**0.004392 ± 0.000**	0.004891 ± 0.000(+)	0.004799 ± 0.000(+)
dMOP2	0.079694 ± 0.063(+)	**0.012550 ± 0.005**	0.017388 ± 0.011(+)
F5	0.523506 ± 0.319 (+)	0.210278 ± 0.171(+)	**0.067280 ± 0.027**
F6	0.139782 ± 0.113(+)	**0.085984 ± 0.041**	0.104402 ± 0.139(+)
F7	0.186740 ± 0.179(+)	**0.036362 ± 0.024**	0.037965 ± 0.015(-)
F9	0.453211 ± 0.250(+)	0.484315 ± 0.496(+)	**0.367412 ± 0.531**
F10	0.523506 ± 0.319(+)	0.210278 ± 0.171(+)	**0.065992 ± 0.027**

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

4.6 Results

4.6.1 Performance Comparison with other DMOEAs

MOEA/D-SVR is compared with RND and MOEA/D-KF, Kalman Filter assisted MOEA/D [78]. Statistical test of T-test for independent samples is performed for the MIGD statistical values. The null hypothesis is that the proposed algorithm of SVR prediction based MOEA, MOEA/D-SVR does not perform significantly better than the compared algorithms at the 95% significance level. The results tabulated in Table 4.4 show that the proposed algorithm performs significantly better than the compared algorithms in 3 out of the 9 benchmark problems. Also, MOEA/D-SVR performs comparably with MOEA/D-KF in one problem (F7), where the statistical test shows that their performances are not significantly different.

4.6.2 Discussion

- Type I DMOPs - FDA1: In Type I DMOPs, the POF remains constant, while the POS changes with time. From Table 4.4, it can be seen that MOEA/D-SVR performs much better than RND in FDA1, however it falls behind MOEA/D-KF, albeit only a small amount.

- Type III DMOPs - dMOP1: In these problems, the POS remains constant, while POF changes with time. RND retains 80% of the populations after a change, thereby giving the algorithm an edge in such problems, where no change is made to the existing converged solutions.

- Type II DMOPs: The rest of the problems used in this empirical study, FDA2, dMOP2 and F5-F10, fall under this category, where both the POF and POS change with time. MOEA/D-SVR performs significantly better than the other 2 algorithms in 3 out of 7 of these problems. F5-F7 have nonlinear linkages between their decision variables and their POS/POF shapes have more difficult structure than that of FDA and dMOP problems. MOEA/D-SVR performs relatively close to MOEA/D-KF in the other 4 problems as well.

  F9 and F10 are complicated problems, wherein the environment does not change smoothly from one time instant to the next unlike all the previously discussed problems. Further, their geometric shapes of two consecutive POF/POS are dissimilar to each other. In F9, the pareto set jumps from one area to another occasionally. The geometric shapes of consecutive POFs are completely different from each other in F10. MOEA/D-SVR is able to perform significantly better than the other 2 algorithms in F9 and F10. The proposed Support Vector Regression based prediction model does not make any assumptions

about the underlying process or similarity in shape of consecutive POFs, thereby leading to its better performance.

4.7 Analysis

4.7.1 Prediction visualization

The SVR prediction model predicts in the decision space and assists the evolutionary algorithm in the tracking of the Pareto Optimal Solutions (POS) in dynamic environments. The prediction performance for the dMOP2 problem is shown in Figure 4.8. The left half of sub-figures in Figure 4.8 depicts the objective space and the right half depicts the decision space. Both the Pareto Optimal Front and Pareto Optimal Set in dMOP2 problem changes with time, making it a Type II DMOP [2]. The solid lines represent the POF/POS for the current time instant. In the following time instant, the POF/POS would be shifted to the dotted line. The red rings and solid circles represent the current measurement and prediction estimate of MOEA/D-SVR respectively, while the black ones indicate that of random reinitialization method (RND). It can be seen that the prediction estimate of SVR in MOEA/D-SVR is quite close to the new POS while the solutions obtained by MOEA/D-DE with the RND method remain close to the previous POS. It is to be noted that the MOEA/D's [29] weight vectors further assist the algorithm in getting a widely distributed set of solutions covering the entire front.

4.7.2 Parameter Selection

C and Gamma parameters of the SVR model are selected based on the mechanism described in 4.4.3. Both the parameters have significant impact

on the performance of the SVR prediction model. Based on the benchmark problem characteristics, different values of C and Gamma may get selected according to the problem definition. Further, since the population prediction is performed in a univariate manner, each decision variable has its own set of parameters that give the best performance and are chosen for future use.

Visualization of the parameters selected for various dimensions of the decision variable shows interesting patterns as observed in figures 4.9 and 4.10. The figures are heatmaps in which the individual boxes in a figure indicate the frequency at which the particular item was selected. For instance, Figure 9, top left figure shows the visualization of C parameter for the FDA1 problem. For decision variable 0, it can be seen that C values of 1 and 10 were selected much less frequently compared to the value of 100.

There are a number of observations that can be made from the visualizations, as enumerated below.

1. The heatmaps' patterns are relatively distinctive for different problems.

2. Problems in which some of the decision variables share similar characteristics such as in FDA2, a repetitive pattern is observed.

3. It can be observed that problems with similar characteristics as a whole, such as F6 and F7 have very similar patterns exhibited, as seen in Figure 4.10.

Observations made from such visualizations may prove useful when drawing conclusions about problems with unknown pareto-optimal characteristics. This may prove to be useful in understanding and analyzing real-world problems whose POS/POF are unknown.

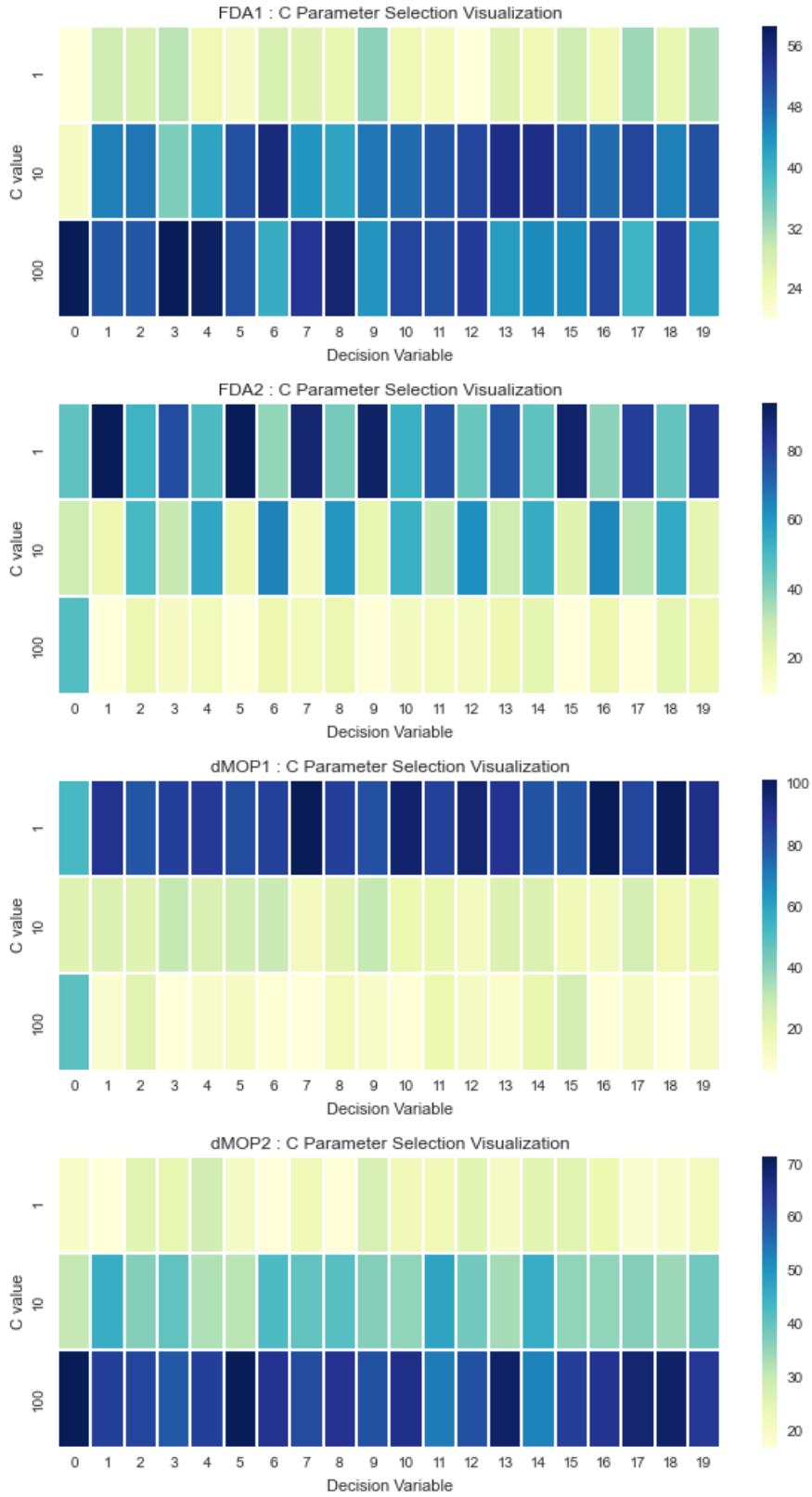


Figure 4.9: C Parameter Selection Visualization based on decision variable number

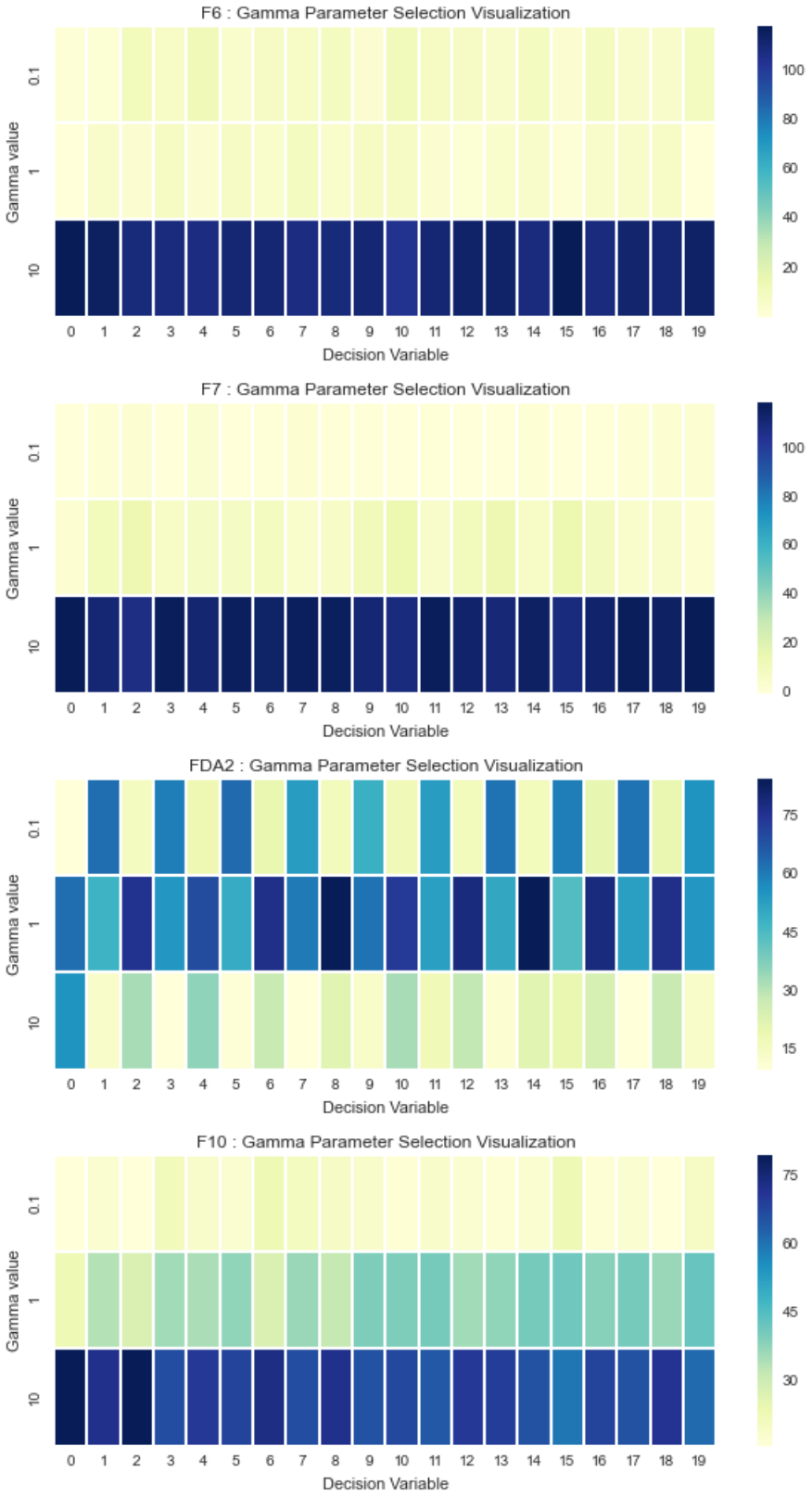


Figure 4.10: Gamma Parameter Selection Visualization based on decision variable number

4.7.3 MOEA/D-SVR Time Series Formulation Visualization

In this subsection, the time series formed in MOEA/D-SVR is visualized for a few problems in Figure 4.11. Each row shows the plots for a particular problem, and different dimensions of the decision variable (nvar) are considered column-wise. A few observations from the figures are enumerated below.

1. The training data contains some noise, as observed for FDA2, nvar = 2 subfigure in Figure 4.11. This results from the fact that the DMOEA may not fully converge to the optimal solution prior to subsequent change.

2. MOEA/D-SVR predictions though not optimal in all cases, are quite close to expectation from visual analysis. Improvements can be attained by further tuning of parameters to search for finer values.

3. Training size was chosen empirically and set as a common value of 35 for all test problems. Problem-specific training size may be needed to improve performance based on the periodicity of POS movement, as observed for $nvar = 0$, in problems F9 and F10.

4.7.4 Influence of severity of Change

n_t determines the severity of change in a problem. A smaller value of n_t means a larger change occurs. The difficulty of the problem increases with decrease in the value of n_t. In this section the severity of the change parameter is varied by steps of 5 and box-plots of IGD values are obtained for MOEA/D-SVR, MOEA/D-KF and RND methods.

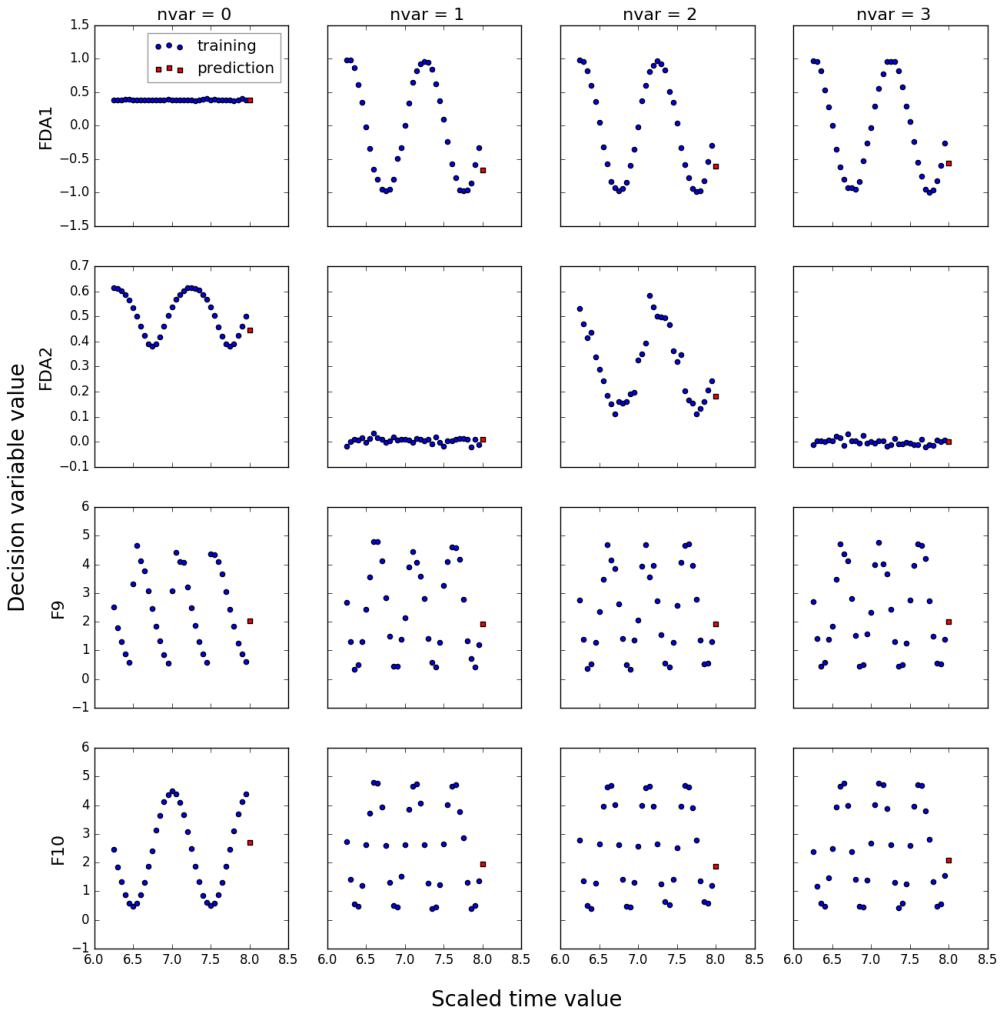
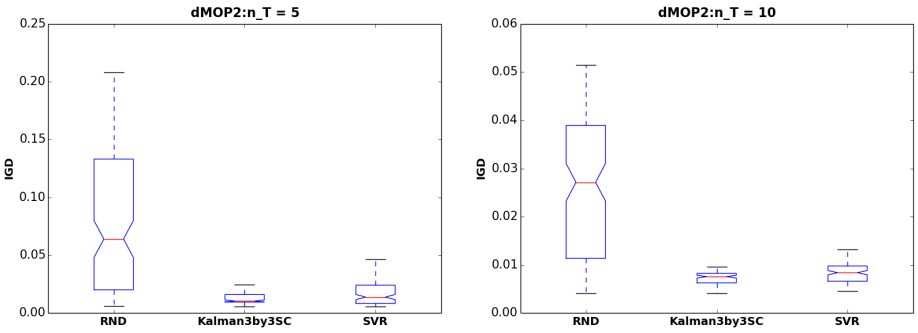
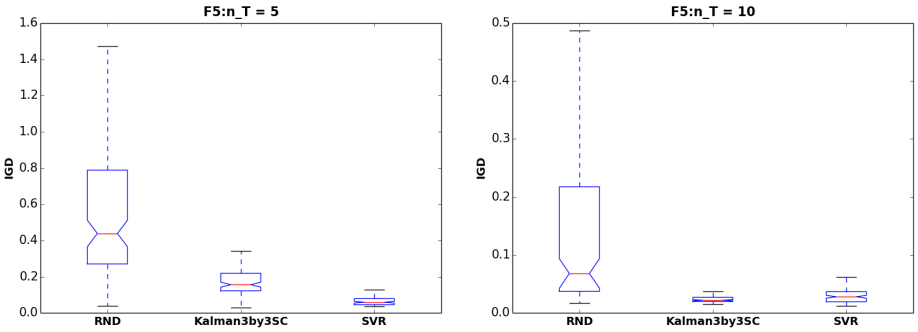


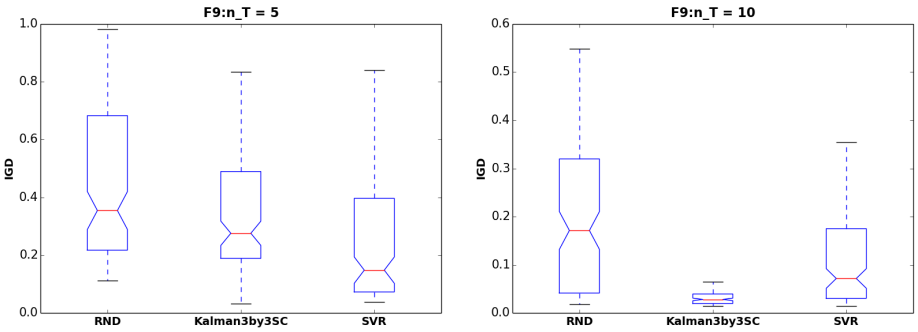
Figure 4.11: MOEA/D-SVR Time Series Formulation Visualization. Blue circles represent the training data and red square denotes the predicted value.



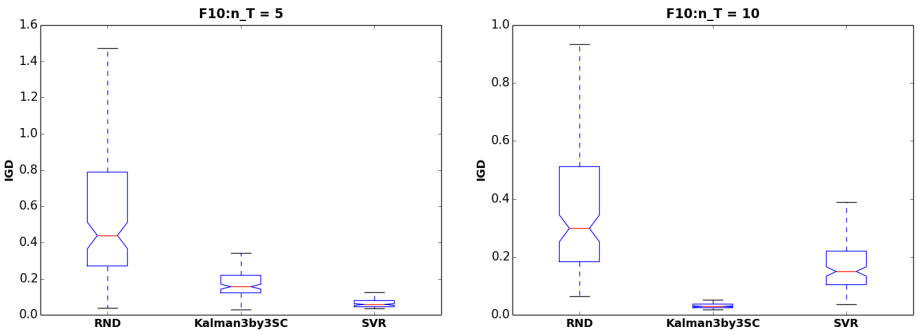
(a) dMOP2



(b) F5



(c) F9



(d) F10

Figure 4.12: Influence of Severity of Change

As n_t changes from 5 to 10, the performance of all algorithms improve. This implies that the smaller the change, the better the algorithm performs. This can be accounted to the fact that when the change is small, the prediction model can be corrected faster and more efficiently from subsequent observations. As seen from Figure 4.12a, the performance of MOEA/D-SVR becomes comparable to that of MOEA/D-KF as the severity of change decreases, i.e., n_t increases. Due to small changes, difference in values used to train SVR are close to each other. The set of values in the training window of SVR could be approximated to a linear time series. Hence the performance of SVR becomes comparable to Kalman Filter, which also assumes the time series to be linear. In other complex problems like F5 (Figure 4.12b), F9 (Figure 4.12c) and F10 (Figure 4.12d), it is observed that performance of MOEA/D-KF becomes better than MOEA/D-SVR. It can be inferred from the discussion that MOEA/D-SVR tends to perform better when the severity of change is higher compared to the other algorithms.

4.8 Chapter Conclusion

A novel Dynamic Multiobjective Evolutionary Algorithm using Support Vector based predictions in the decision space is proposed. It is built on the MOEA/D framework and the Support Vector Regression prediction model is used to estimate the subsequent optimal values of decision variables. In this method, a time series is formed by the near-optimal solutions obtained by the Evolutionary Algorithm in previous changes. Support Vector Machines are used in tandem with the Evolutionary Algorithm to predict new solutions for future generations from the time series, when a change in the environment is detected. Change detection is performed through sentry particles. Experimental results demonstrate that the proposed algorithm

shows significantly improved performances over a number of test benchmark problems.

MOEA/D-SVR performs significantly better in some of the problems, while it performs comparably with MOEA/D-KF in some problems. The performance of MOEA/D-SVR is particularly good in the more complicated problems wherein the environment does not change smoothly from one time instant to another unlike all the other problems, as the proposed prediction model using Support Vector Regression does not make any assumptions about the underlying process or similarity in shape of consecutive POFs. Analysis of the parameter selection visualization provides insights in the decision variable feature space which may be useful in understanding and analyzing real-world problems whose Pareto Optimal characteristics or the linkages between decision variables are unknown.

Chapter 5

Adaptive Constraint Handling in Constrained Dynamic Multiobjective Optimization

5.1 Introduction

Optimization problems are abundant in all walks of life, whether in a scheduling scenario or deciding which product to buy from a multitude of choices. While the problems can be single objective or multiobjective, static or dynamic, constraints are common in most of the problems. However, in the Evolutionary Computation literature the problems are divided into two categories - unconstrained or constrained. Unconstrained problems consist of only finding the decision variables that minimize or maximize the given objective(s). They are commonly referred to as boundary constrained problems, wherein constraints are imposed on the range of values that the decision variables are allowed. In essence, these problems are unconstrained problems. In the previous chapters in the thesis, unconstrained dynamic problems have been explored and a number of mechanisms to solve them

have been discussed. Once the boundary conditions of decision variables are taken into account, the entire search space is feasible space in unconstrained problems.

In constrained problems, however, the presence of constraints alters the feasibility of regions resulting in many types of difficulty to optimization algorithms. Constraints can leave most of the search space untouched and therefore, feasible, while regions close to the original pareto optimal front are rendered infeasible. The resultant pareto optimal front is at the boundary/intersection of feasible and infeasible regions. Other problems cause difficulty in the entire search space. Some problems make the entire unconstrained Pareto-optimal region infeasible in the presence of the constraints. Constraints make the Pareto-optimal region discontinuous, with a number of disconnected continuous regions. In some other cases, the disconnected continuous regions could be just a single point. Such problems can be made even more difficult by making it harder to reach the single optimal points by surrounding them with long infeasible tunnel regions. Another form of difficulty is experienced when the disconnected regions are not uniformly distributed.

Problems with such constraints cause difficulties only in the vicinity of the original Pareto Optimal Front. While the previously discussed problems bring in higher difficulty in the vicinity while the rest of the search space remains continuously feasible, other problems bring in higher complexity by making the transition from feasible to infeasible regions far away from the Pareto-optimal region. Feasible and infeasible regions are interspersed whereby the algorithm has to tunnel through an infeasible region(s) to reach the Pareto-optimal region. The interspersion could also occur along the pareto-optimal region rendering some parts infeasible resulting in disconnected feasible pareto optimal regions.

Constrained test benchmark problems with such difficulties and a tunable manner are proposed in [105]. These problems have been converted to dynamic constrained problems in this thesis to test the performance of dynamic optimization strategies such as diversity introduction, diversity maintenance and prediction techniques combined with an adaptive threshold based constraint handling mechanism.

The rest of the chapter is organized as follows. Section 5.2 provides some background on constraint problems. Section 5.3 reviews related work in the literature on static constraint handling methods as well as those in dynamic environments. Section 5.4 describes the methodology adopted in this chapter to study DCMOPs. Section 5.5 shows the empirical study involving test benchmark problems, performance metrics, results and performance comparison with other algorithms. Section 5.6 provides an analysis on the results. Section 5.7 concludes this chapter.

5.2 Background

It is necessary to consolidate understanding of the problem definitions in constrained environments to able to tackle them better. To that end, a number of definitions are elaborated in this section.

5.2.1 Constrained Multiobjective Optimization Problem Definition

A Constrained Multiobjective Optimization problem can be expressed in its general form mathematically as

$$
\begin{aligned}
&\text{Minimize/Maximize} && f_m(x), m = 1, 2, \ldots, M; \\
&\text{subject to} && g_j(x) \geq 0, j = 1, 2, \ldots, J; \\
&h_k(x) = 0, && k = 1, 2, \ldots, K; \\
&x_i^L \leq x_i \leq x_i^U, && i = 1, 2, \ldots, n.
\end{aligned}
$$

where f_i is the i-th objective function and M is the number of objectives. $f(x) = [f_1(x)f_2(x)...f_m(x)]^T$ forms the objective vector, $f(x) \in \mathbb{R}^M$. A solution x is a vector of n decision variables: $x = [x_1x_2...x_n]^T$. The above general problem is associated with J inequality constraints, $g_j(x)$ and K equality constraints, $h_k(x)$. The last set of constraints are called *variable bounds*, restricting each decision variable x_i to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound. These variable bounds constitute the *decision variable space* $\Omega \in \mathbb{R}^n$, or simply the decision space.

In the presence of constraints g_j and h_k, the entire decision variable space Ω may not be feasible. The feasible region S is the set of all feasible solutions in the context of optimization.

5.2.2 Dynamic Constrained Multiobjective Optmization Problem Definition

When considering dynamic environments, the objectives (by extension, the Pareto Optimal Front(s)), Pareto Optimal Set and/or the constraints may change with time. In this thesis, the focus is to analyze dynamic optimization techniques in constrained environments. Therefore, we consider the constraints to be static while the Pareto Optimal Solutions change with time. This can be represented mathematically in the following form,

$$
\begin{aligned}
&\text{Minimize/Maximize} && f_m(x,t), m = 1,2,\ldots,M;\\
&\text{subject to} && g_j(x) \geq 0, j = 1,2,\ldots,J;\\
&h_k(x) = 0, && k = 1,2,\ldots,K;\\
&x_i^L \leq x_i \leq x_i^U, && i = 1,2,\ldots,n.
\end{aligned}
$$

wherein the time variable, t in the objective expression indicates that the problem changes with time.

5.2.3 Other definitions

Feasibility ratio is a measure of the number of feasible solutions in the population in a generation over the total population size.

$$\textit{Feasibility ratio}, fr = \frac{\text{Number of feasible individuals}}{\text{Total Population Size}}$$

Constraint violation in inequality constraints occurs when $g_j(x) < 0$. In the case of equality constraints, strict adherence to complete equality is not expected. A user-defined tolerance value (δ) is defined, usually very small such as 0.001 or 0.0001 [106]. Therefore, constraint violation can be measured as

$$c_j(x) = \begin{cases} g_j(x), & \text{when } g_j(x) < 0 \\ 0, & \text{otherwise} \end{cases}$$

for inequality constraints. For equality constraints, constraint violation can be expressed as

$$c_k(x) = max(0, |h_k(x)| - \delta)$$

5.3 Related Work

Various constraint handling methods have been proposed in the literature. Constraint handling for single objective problems has been a highly active research area for a number of decades [107] [108] [109]. Approaches proposed for single objective optimization have been directly ported to multiobjective optimization. However, it is much more recently that constraint handling is being considered in Evolutionary Multiobjective Optimization [110]. Any Evolutionary Algorithm encounters three stages of the population during optimization -

- Only infeasible individuals
- Mix of feasible and infeasible individuals
- Only feasible individuals

In initial generations of the evolutionary process, depending on the difficulty and complexity of the problem, the population might contain a mix of feasible and infeasible individuals or only infeasible individuals. The goal of a Constrained Multiobjective Optimization Evolutionary Algorithm (CMOEA) is to ultimately find feasible as well as importantly, optimal solutions. The numerous methods seen in the literature approach these population stages in different perspectives leading to differing performances. They can be grouped into the following main categories:

1. Penalty function based
2. Modified Genetic Operators
3. Repair
4. Multiobjective Approach

5. Preference based

A review of the various groups of methods from numerous works is discussed in detail in the following subsections.

5.3.1 Penalty function based methods

Penalty function based methods are the simplest and most easy to implement (thereby commonly used) constraint handling methods. Constrained optimization problems are generally converted to unconstrained optimization problems by taking the constraint violation into account with the objective functions or fitness value by adding a penalty. These methods can be further divided into *static, death, dynamic* and *adaptive* approaches.

Static penalty function methods are those in which current generation number is not taken into account [108] [111]. In [112], genetic algorithms are used to solve non-linear constrained optimization problems where a multi-stage weight assignment to the penalty applied is designed to handle constraint violation which showed better performance than a single-stage weight assignment.

In death penalty method [108], irrespective of the degree (whether they violate only one constraint to a small extent or many constraints) to which individuals violate constraints, infeasible individuals are discarded without extracting any information from them. While this method is very easy to implement, it ignores the fact that some infeasible individuals may carry important information in some generations compared to their feasible counterparts, making use of which may lead to faster convergence. Also, in highly constrained problems where the ratio of feasible to total search space is very low, it might be very difficult to arrive at a population with a reasonable feasibility ratio, let alone finding optimal solutions. In many problems, finding feasible solutions itself is considered a NP-hard

problem [109]. Many studies have shown that the performance of death penalty methods is inferior to other penalty methods such as adaptive penalty or through use of penalties that drive the solutions towards the feasible region [113] [114].

When current generation number is used in determining the penalties, then the method is termed as dynamic penalty function method. In [115], the authors proposed a non-stationary penalty function which increases with generation number to solve general nonlinear programming problems using real-valued genetic algorithms. As the penalty number increases with increasing generation number, it puts higher selective pressure on the GA to find a feasible solution. A varying fitness function technique is proposed in [116] where the penalty factors are dynamically adjusted during the evolutionary search process. A number of shapes for the penalty function is considered ranging from exponential, linear to square, cubic and quadratic. The proposed method is tested on the cutting stock problem and the Unit Commitment problem. Gradual application of the penalty is crucial for the success of the GA as certain penalty functions reach the maximum very quickly leading to poor performance. Further, the choice of penalty function is highly problem dependent and it is not easy to design a function that would work for a number of problems.

In adaptive penalty function methods [117], information gathered through the evolutionary search process is taken into account to arrive at the penalty to apply. An adaptive constraint handling approach embedded in MOEA/D which adaptively decides on the violation threshold for comparison is proposed in [118]. In [119], a self adaptive penalty function approach is proposed, in which a new fitness value, called distance measure in the normalized fitness-constraint violation space and two penalty values are applied to infeasible individuals to identify the best infeasible individuals

in the current population. The feasibility ratio of the current population determines whether the search should be towards finding feasible solutions or optimal solutions. Initially proposed for single objective optimization, the self adaptive penalty method was extended to multiobjective constrained problems in [110] [106] by implementing the technique on NSGA-II.

5.3.2 Modified Genetic Operators

GENOCOP, Genetic Algorithm for Numerical Optimization of Constrained Problems system assumes linear constraints only and a feasible initial population. The GENOCOP system showed superior performance to traditional methods when applied to the nonlinear transportation problem and also effectively reduces the search space by eliminating equality constraints along with an equal number of decision variables. Strategic Oscillation, originally proposed in an Operations Research technique [109], has subsequently been used in combinatorial and nonlinear optimization problems. It entails an estimation of locating the boundary of the feasible region through an adaptive penalty mechanism in which a mechanism is devised to cross the feasibility boundary back and forth to identify the best direction of movement.

5.3.3 Repair methods

Repairing involves modifying infeasible individuals and pushing them towards the feasible region [120]. This method increases the computational complexity of the algorithm based on the rate at which repairing is done to the infeasible population and repair method is highly problem dependent and need to be designed specifically for each problem [109]. In [118], gradient local search is used, along with the adaptive penalty threshold to further accelerate the rate of convergence by invoking repair method

to repair the infeasible solutions at a probability of 5%. The weakness of Genocop [121] method discussed in the previous subsection lies in its inability to handle nonconvex search spaces, to deal with nonlinear constraints in general. Genocop III [122] is able to overcome this restriction by using a dual population approach. A repair mechanism is implemented by identifying fully feasible points known as reference points and generating random solutions between a reference point and the infeasible solution in consideration.

5.3.4 Multiobjective Approach

In this approach, even single objective optimization problems are solved using Multiobjective Evolutionary Algorithms wherein constraints are incorporated as one or more objectives. In [111], the authors proposed a Multiobjective approach for solving single objective but constrained problems. A two-phase approach is adapted wherein in the first phase, the objective function is completely ignored and the problem is treated as a constraint satisfaction problem. Individual with the least constraint violation is considered as an elite solution and is archived. The genetic search is guided towards minimizing the constraint violation and eventually finding feasible solutions. In the second phase, the objective function and satisfaction of constraints are considered as 2 objectives of a bi-objective problem and are tried to be simultaneously optimized.

An Evolutionary Algorithm of Nondominated Sorting with Radial Slots known as ENORA is proposed in [123], which incorporated the Pareto concept of Multiobjective optimization using the min-max formulation for constraint handling and a new diversity mechanism based on the partitioning of search space in a set of radial slots along which successive populations are positioned. Another algorithm based on multiobjective optimization

techniques to handle constraints is proposed in [124] in which three models of a population-based algorithm generator, an infeasible solution archiving and replacement mechanism are introduced.

5.3.5 Preference based methods

Many methods involve exercising preference of feasible solutions over infeasible solutions. In such methods, infeasible solutions may be considered worse than feasible solutions irrespective of their objective values. One of the most popular techniques proposed is proposed in [27] named as the Constraint Domination principle. The concept of domination in multiobjective context is modified to include constraints. Constraint handling is attained through applying a modified binary tournament selection in choosing the individuals that survive to the next generation. When two individuals are considered in the binary tournament selection, only three situations are possible -

1. both solutions are feasible
2. both solutions are infeasible
3. one is feasible, while the other is infeasible.

A solution i is said to constrained-dominate a solution j, if any of the following conditions is true.

1. Solution i is feasible and solution j is not.
2. Both solutions are infeasible, but solution i has a smaller constraint violation value.
3. Both solutions are feasible, but solution i dominates solution j.

Using the constraint domination principle, feasible solutions are always better ranked than infeasible solutions. Between infeasible solutions, the

one with smaller constraint violation value has a better nondomination rank. This method is easy to implement and shows competitive performance as well, but the selection pressure may not be sufficient in problems with highly constrained environments. In [125], the authors proposed a MOEA in which constraint handling is performed through three nondominated rankings - firstly based on objectives, secondly on the different constraints, and finally ranked based on the combination of all objectives and constraints. This demands a larger computational complexity, while that of the Constraint Domination Principle is not deviant from the original NSGA-II's computational complexity.

Penalty functions based methods are relatively unable to strike the balance between objective functions and penalties applied to constraint violations. Runarsson and Yao introducted the stochastic ranking approach in [126] [127] wherein the objective functions and penalties are stochastically ranked using a probability factor that determines which of the two (objective functions or penalties) determine the rank of an individual.

A CMOEA with ensemble of constraint handling methods is proposed in [128] based on the argument that it is impossible for a single constraint handling method to outperform all other methods on all problems irrespective of the exhaustiveness of parameter tuning. In this CMOEA, self adaptive penalty, superiority of feasible solution and ϵ-constraint are employed as the constraint handling techniques, each method associated with its own population.

5.3.6 Dynamic Constrained Multiobjective Evolutionary Algorithms

Several Evolutionary Algorithms have been proposed to tackle Dynamic Single objective Constrained problems [107]. However, only a handful of

Dynamic Constrained Multiobjective Evolutionary Algorithms (DCMOEA) have been seen in the literature. In [129], the authors implemented the self adaptive penalty function method proposed in [106] for Dynamic Multiobjective Optimization. A bio-inspired Artificial immune system [130] [131] is used to develop a Dynamic Constrained Multiobjective Optimization Artificial Immune System (DCMOAIS) to dynamically track the Pareto fronts of time-varying constrained multiobjective problems with changing variable dimensions. Constraint Domination Principle [105] discussed in the previous section implemented in Dynamic NSGA-II also results in a DCMOEA which outperforms the algorithm proposed in [125].

5.4 Methodology

To bridge the research gap in Evolutionary Dynamic Constrained Multiobjective Optimization, a DCMOEA based on an adaptive constraint handling mechanism inspired from works in the literature [105] [118] along with dynamic optimization strategies are proposed in this work. The following subsections elaborate the constraint handling mechanisms considered and the dynamic optimization techniques to track the time-varying optima.

5.4.1 Constraint Handling Mechanisms

A review of the various constraint handling methods in the literature (refer to section 5.3) indicated that there has not been much work in tackling constraints in the decomposition framework of MOEAs such as incorporating the mechanisms in MOEA/D, MOEA/D-DE or related algorithms. The penalty function based methods involve adding a penalty expression from the constraint violation space to the objective function values, which might distort the neighbourhood relationship underlying the weighted vector

formulation in MOEA/D framework.

The Constraint Domination Principle [105] showed competitive performance in finding feasible solutions to static constrained multiobjective optimization problems. This method was proposed to be incorporated in a domination-based framework wherein the Constraint Domination Principle is used to determine the rank(s) of the population to be used in the nondominated sorting. The principle's main ideology has been extracted and incorporated into the decomposition-based framework of MOEA/D-DE in this work. The evolution of individuals in the population proceeds normally through Differential Evolution in every generation followed by mutation. Each child population needs to be evaluated on whether it can replace individuals in the parent population. In MOEA/D and MOEA/D-DE, the comparison between child and parent population is performed by evaluating their fitness value using the Weighted-Sum or Tchebycheff approach (by taking into account the parent's weight vector). If the child's fitness is better than that of the parent's, then the child replaces the parent and the ideal vector is updated accordingly. In the proposed approach for handling constrained problems, the constraint violation of each parent and child individual is computed during objective function evaluation. When survivor selection needs to be performed, the following steps are employed to determine the individual that can survive to the next generation.

1. If parent has no constraint violation, while the child violates some constraint(s), then the parent solution is not replaced and no update is required.

2. If parent has some constraint violation(s), while the child is feasible, then the child replaces the parent in the population and the ideal point reference is updated.

3. If both have constraint violations, then the solution with lower constraint violation is selected.

4. If both are feasible solutions, then their fitness is evaluated normally using weighted-sum or tchebycheff approach and the solution with lower fitness value is chosen to survive (assuming minimization problem).

However, one of the drawbacks in this approach is that a feasible individual is always preferred over an infeasible individual. Scenarios wherein an infeasible individual might have better objective function values are not taken into account when employing this constraint handling principle. If some amount of information can be incorporated into the evolutionary process from the infeasible individuals (even when compared against feasible individuals) with reasonably small constraint violation, convergence towards feasibility and subsequently optimality may be achieved faster. In order to incorporate information from infeasible individuals, the feasibility/infeasibility definition is altered using an adaptive threshold. Individuals whose constraint violation values are lesser than that of the threshold are considered at par with feasible individuals.

Adaptive Constraint Threshold

The performance of the algorithm would be highly dependent on the threshold value set and care should be taken not to set too low a value, which would mean that only infeasible individuals very close to the feasibility boundary may be affected. Setting too high a value may also result in many infeasible individuals replacing feasible individuals in the population. During initial generations, it is helpful to start with a relatively higher threshold value, which is subsequently gradually reduced when the feasibility

ratio of the population increases, i.e. the number of feasible individuals in the population increases. Range of constraint violation values is highly dependent on each test benchmark problem. Therefore, firstly, the mean constraint violation of the population, CV_{mean} is computed after fitness evaluation as follows,

$$CV_{mean} = \frac{\sum_{i=1}^{N} CV_i}{N}, \tag{5.1}$$

where N denotes the population size, CV_i stands for the constraint violation value of individual i. The feasibility ratio (fr) is calculated as the ratio of the number of feasible individuals in the population to the population size(refer to 5.2.3). Subsequently, the constraint threshold (CV_δ) is calculated as

$$CV_\delta = fr \times CV_{mean}. \tag{5.2}$$

The computed adaptive constraint threshold is used to determine whether the parent or the child solution survives to the next generation.

- If both the solutions are feasible, or if one of them is feasible and the other's constraint violation value is lesser than CV_δ, then the solutions are compared based on their fitness value computed using weighted-sum or tchebycheff approach.
- If both of them are infeasible, then three conditions are possible.
  - If the constraint violation value of both are below the threshold, then comparison is again based on their fitness value.
  - However, if one of the constraint violation values is lesser than CV_δ and the other is not, then the former is chosen.

– If both the CV_i are greater than CV_δ, then the solution with lesser constraint violation value is chosen.

5.4.2 Dynamic Optimization Techniques

The proposed constraint handling mechanism needs to be incorporated in a dynamic MOEA to track the time-varying optima. The Kalman Filter based prediction DMOEA proposed in Chapter 3 is chosen as the main Dynamic Optimization technique to act along with the adaptive constraint threshold based constraint handling mechanism to tackle Dynamic Constrained Multiobjective Optimization Problems(DCMOPs). Further, random reinitialization method (RND) is also considered to study the effect of this strategy in handling DCMOPs. In RND, when change occurs in the problem during evolution, 20% of the population is randomly reinitialized in the decision search space, while the remaining 80% of the population is retained as such. Hypermutation is another method that is considered to introduce diversity in the population after a change is detected in DMOPs. A high value of mutation probability (0.5) is employed to increase exploration in hypermutation after a change has occurred.

Algorithm pseudocode of proposed constraint handling method with the dynamic optimization techniques is provided in Algorithm 5.1 for easy reference.

5.5 Empirical Study

5.5.1 Benchmark Problems

There is a lack of benchmark problems in the context of constrained multi-objective optimization in dynamic environments [129]. Static constrained

Multiobjective Optimization benchmarks proposed by Deb et al in [105] pose significant difficulties to finding pareto optimal solutions in constrained environments. These problems have been modified to incorporate dynamic characteristics. The Pareto Optimal Set of all problems changes with time, and there are nonlinear constraints of varying difficulties applied in the objective space. All problems have 2 objectives and 10 decision variables. DCTP1 has 2 constraints while DCTP2-7 have 1 constraint each of varying difficulty which are attained by assigning different values to 6 parameters in the benchmark function definition. Frequency of change, τ_T and severity of change, n_T are applicable to the designed DCMOPs as well.

5.5.2 Experimental Setup

The parameter values for various components of the proposed DCMOP are tabulated in table 5.1. A population size of 100 is generally employed for bi-objective problems. However, because of the increased difficulty in considering the DCMOPs, a population size of 200 is used for all problems. The problem changes every 10 generations, i.e. the frequency of change, τ_T is 10 and the severity of change, n_T is also set at a reasonable value of 10.

5.5.3 Performance Metrics

The Hypervolume measure or Hypervolume performance indicator [70] was first proposed in [132] as 'size of the space covered'. The Hypervolume indicator is one of the most popularly used measures for the performance of Multiobjective Optimization algorithms due to its theoretically good characteristics [70] [133] [134] [135]. Also, the class of algorithms known as Indicator based algorithms [136] [137] [138] [139] [135] [140], usually employ Hypervolume as the performance indicator to provide feedback to the Optimization algorithm about its performance as the Hypervolume metric

Table 5.1: Experiment Settings

Number of decision variables, n	10 for all test problems
Population size	200 for all problems
Number of Constraints	2 for DCTP1, 1 for DCTP2 to DCTP7
Neighborhood	Size: 20.
Probability that parents are selected from the neighborhood	0.9
Decomposition method	Tchebycheff
Differential Evolution	CR = 1.0 and F = 0.5
Polynomial Mutation	$\eta = 20$, $p_m = 1/n$.
Number of detectors	10
Percentage for RND model	20%
Percentage for HYP model	20%
Hypermutation probability for HYP model	0.5
KF model process noise	Gaussian of N(0, 0.04)
KF model observation noise	Gaussian of N(0, 0.01)
Dynamic Setting	Frequency of change τ_T: 10, Severity of change n_t: 10
Number of changes	40
Number of generations	400
Number of runs	30

does not require the knowledge of the Pareto Optimal Front. However, the high computational complexity of Hypervolume calculation has been frequently criticized in the literature, especially when there are many objective functions. To address this issue, many fast and efficient methods to perform hypervolume calculation/approximation have been proposed

[141] [142] [143] [144].

Hypervolume of a solution set can be defined as the volume of the region dominated by the solution set given the location of a reference point. The Hypervolume measure is computed in the objective space and it contains both convergence and divergence information. A higher value of the measure implies better optimization performance. Hypervolume is the only known indicator that is in compliance with the concept of Pareto dominance, i.e. if one set of solutions dominate another set of solutions, the former's hypervolume value is always higher than that of the latter.

The Hypervolume metric is modified for evaluating performance in dynamic environments, by taking average of the hypervolume values during certain time instances over a run. The chosen time points are the instances immediately before a change occurs. Therefore, the modified Hypervolume metric is given by,

$$MHPV = \frac{1}{|T|} \sum_{t \in T} HPV(P^t, Nadir^*), \tag{5.3}$$

where, T is a set of discrete time points in a run, $|T|$ is the cardinality of T. The Hypervolume of the obtained pareto front at a particular time instant, P^t is calculated by providing a reference point. The reference point is chosen as the estimated nadir point, i.e. the point in the objective space with the worst objective function values that is obtained in the population [145]. A higher value of MHPV implies better dynamic optimization performance. The measure can be combined with the feasibility ratio of the population to evaluate whether constraints have been adhered to.

5.5.4 Results

The statistical results of MHPV values for the DCTP test benchmark problems are tabulated in Table 5.2. The proposed constraint handling mechanism implemented in MOEA/D-DE with the dynamic optimization techniques of Kalman Filter prediction method, random reinitialization and hypermutation are denoted as CMOEA/D-KF, CMOEA/D-RND and CMOEA/D-HYP respectively. The 3by3 variant of Kalman Filter prediction method is employed. Both the standalone(CMOEA/D-KF) and the scoring scheme based method(CMOEA/D-KFSC) are used to solve the DCMOPs.

Table 5.2: Experiment Results of CMOEA/D-KF, CMOEA/D-RND, CMOEA/D-HYP

Problems	CMOEA/D-RND	CMOEA/D-HYP	CMOEA/D-KF	CMOEA/D-KFSC
DCTP1	0.300333 ± 0.001(+)	**0.302992 ± 0.001**	0.297087 ± 0.001(+)	0.301753 ± 0.001(+)
DCTP2	0.444539 ± 0.057(+)	**0.474703 ± 0.011**	0.337455 ± 0.039(+)	0.417221 ± 0.047(+)
DCTP3	0.277976 ± 0.002(+)	**0.288567 ± 0.002**	0.270304 ± 0.001(+)	0.274611 ± 0.004(+)
DCTP4	0.076447 ± 0.013(+)	0.074765 ± 0.012(+)	**0.087591 ± 0.004**	0.061592 ± 0.008(+)
DCTP5	0.275923 ± 0.003(+)	**0.284768 ± 0.003**	0.268160 ± 0.001(+)	0.269971 ± 0.003(+)
DCTP6	0.297524 ± 0.008(-)	0.261173 ± 0.088(+)	0.175568 ± 0.048(+)	**0.298233 ± 0.008**
DCTP7	0.428832 ± 0.002(+)	*0.431772 ± 0.003*	0.208133 ± 0.009(+)	0.431298 ± 0.003(-)

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

Statistical test of T-test for independent samples is performed for the MHPV statistical values. The hypermutation based method performs best

in 4 out of the 7 problems, while the Kalman filter based methods perform best in 2 of the problems. In DCTP7, the hypermutation and scoring scheme based Kalman filter method perform comparatively well.

5.5.5 Performance Comparison

The proposed algorithms are compared with Dynamic NSGA-II (DNSGAIIA and DNSGAIIB) embedded with the constraint domination principle to tackle DCMOPs. DNSGAIIA randomly reinitializes a part of the population when a change occurs, while DNSGAIIB employs a higher mutation probability when a change is encountered.

The statistical results of MHPV values for the various algorithms are presented in Table 5.3.

Table 5.3: Performance Comparison on DCTP DCMOPs

Problems	CMOEA/D-KF	CMOEA/D-KFSC	DNSGAIIA	DNSGAIIB
DCTP1	0.297087 ± 0.001(+)	**0.301753 ± 0.001**	0.287386 ± 0.003(+)	0.288957 ± 0.002(+)
DCTP2	0.337455 ± 0.039(+)	**0.417221 ± 0.047**	0.326238 ± 0.004(+)	0.327419 ± 0.003(+)
DCTP3	0.270304 ± 0.001(+)	0.274611 ± 0.004(+)	0.284389 ± 0.003(-)	**0.285820 ± 0.003**
DCTP4	0.087591 ± 0.004(+)	0.061592 ± 0.008(+)	0.128391 ± 0.011(-)	**0.133737 ± 0.010**
DCTP5	0.268160 ± 0.001(+)	0.269971 ± 0.003(+)	0.281710 ± 0.004(-)	**0.282511 ± 0.003**
DCTP6	0.175568 ± 0.048(+)	**0.298233 ± 0.008**	0.263610 ± 0.008(+)	0.260739 ± 0.049(+)
DCTP7	0.208133 ± 0.009(+)	**0.431298 ± 0.003**	0.382348 ± 0.011(+)	0.390969 ± 0.009(+)

(+) (and (−)) indicates that the difference between the marked entry and the best entry is statistically significant (and insignificant, respectively) using paired two-sample t-test(both are at the 5% significance level).

Statistical test of T-test for independent samples is performed for the MHPV statistical values. The null hypothesis is that the proposed algorithm of Kalman prediction based MOEA, MOEA/D-KF combined with the novel adaptive constraint handling mechanism does not perform significantly better than the compared algorithms at the 95% significance level. However, the results tabulated show that the proposed algorithm performs significantly better than the compared algorithms in 4 out of the 7 benchmark problems.

5.5.6 Discussion

DCTP1 problem has two constraints which render a portion of the unconstrained Pareto Optimal Front infeasible. The two constraints boundary form part of the constrained Pareto Optimal Front. Trend of Hypervolume performance metric over number of changes for DCTP1 is shown in figure 5.1. While all algorithms perform competitively in later changes, CMOEA/D-KF and CMOEA/D-KFSC perform better right from the first change.

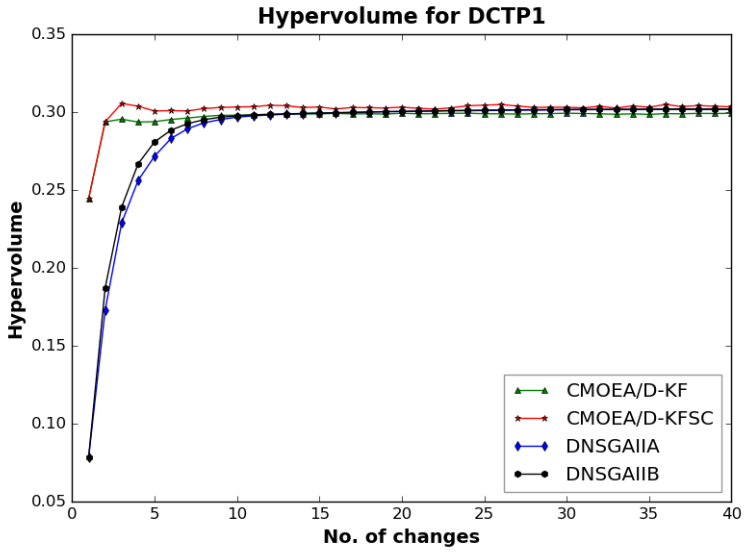


Figure 5.1: Hypervolume trend comparison in DCTP1, $\tau_T = 10, n_T = 10$

DCTP2 to DCTP7 have single constraints but their difficulty varies based on the difficulty posed by the constraints to reach the optimal

solutions. DCTP2 consists of disconnected pareto optimal regions which poses difficulty to the DCMOEA to find as many disconnected regions as possible. CMOEA/D-KFSC performs significantly better than the other algorithms in this problem (refer to Figure 5.2).

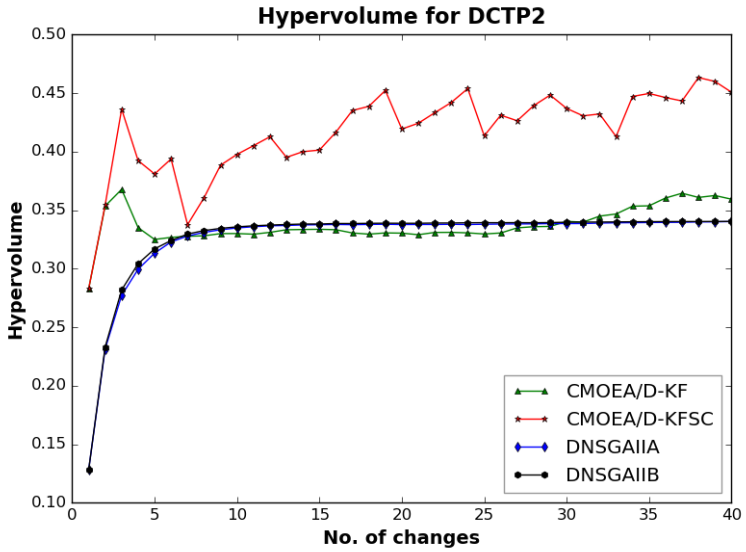


Figure 5.2: Hypervolume trend comparison in DCTP2, $\tau_T = 10, n_T = 10$

In DCTP3, the disconnected pareto optimal regions in DCTP2 reduces to single pareto optimal solutions by increasing the value of d in the benchmark problem definition. The problem complexity is further increased in DCTP4 by increasing the value of parameter a which makes the transition from continuous to discontinuous feasible region far away from the pareto optimal region. In DCTP5, the disconnected regions are not equally distributed in the objective space. DNSGAII algorithm variants, DNSGAIIA and DNSGAIIB perform better than the proposed algorithms in these problems, which may indicate that selective pressure needs to be increased to tunnel to the optimal regions as well as to increase focus on diversity maintenance. The hypervolume trends for these problems are given in Figure 5.3.

The Pareto Optimal Front of DCTP6 lies entirely on a part of the constraint boundary. DCTP6 consists of a number of holes of infeasible

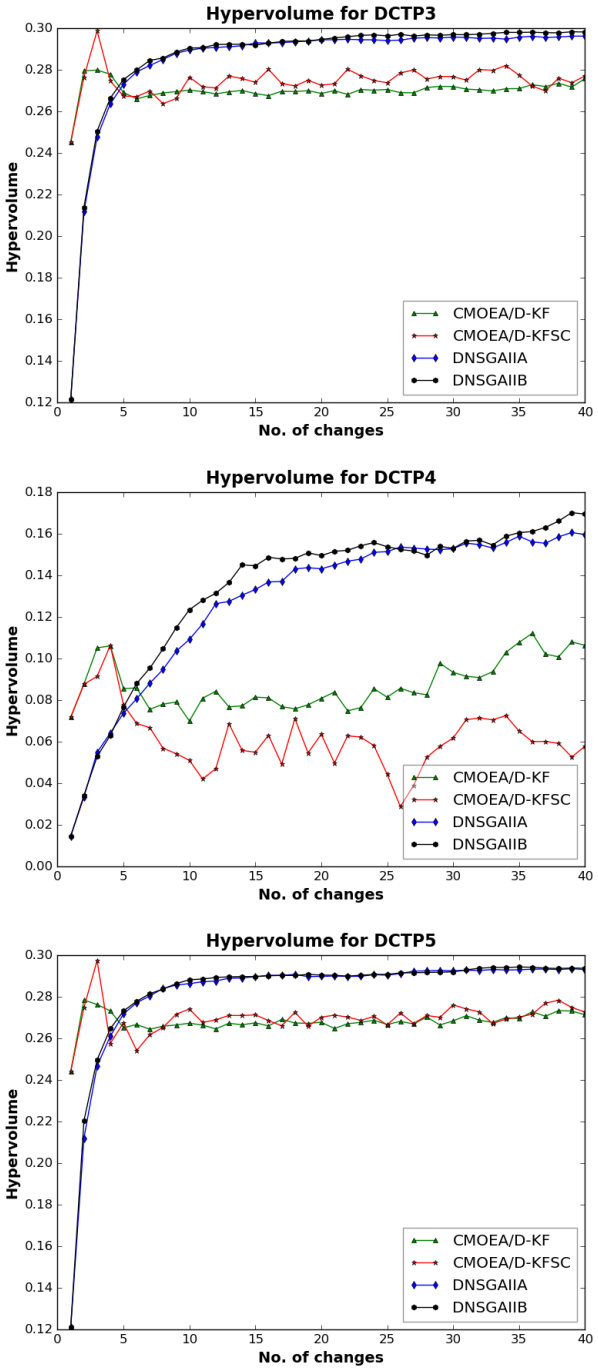


Figure 5.3: Hypervolume trend comparison in DCTP3-5, $\tau_T = 10, n_T = 10$

regions before coming to the island containing the Pareto Optimal Front which significantly increases the difficulty of the problem. Infeasibility in the objective search space comes along the Pareto Optimal Front in DCTP7. This problem renders some portions of the unconstrained Pareto Optimal

Front infeasible, resulting in a disconnected set of continuous regions. It has been pointed out that an algorithm need to maintain adequate diversity right from the beginning of a simulation run to find all (or atleast many) such disconnected regions. The hypervolume trends for these problems are given in Figure 5.4.

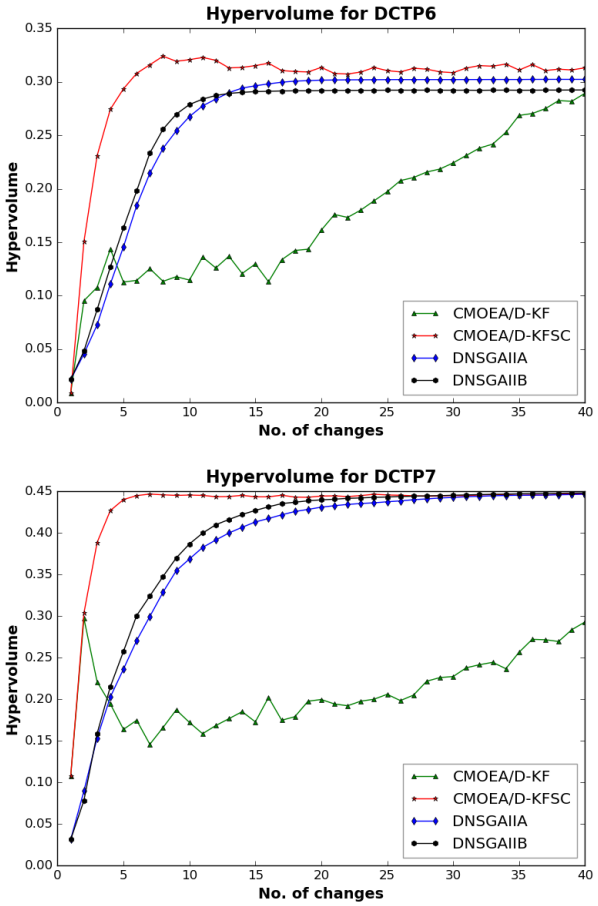


Figure 5.4: Hypervolume trend comparison in DCTP6 and DCTP7, $\tau_T = 10, n_T = 10$

5.6 Analysis

5.6.1 Influence of severity of change

Decreasing change severity parameter value, n_T results in increase in problem difficulty as the distinction between subsequent Pareto Optimal Solutions is

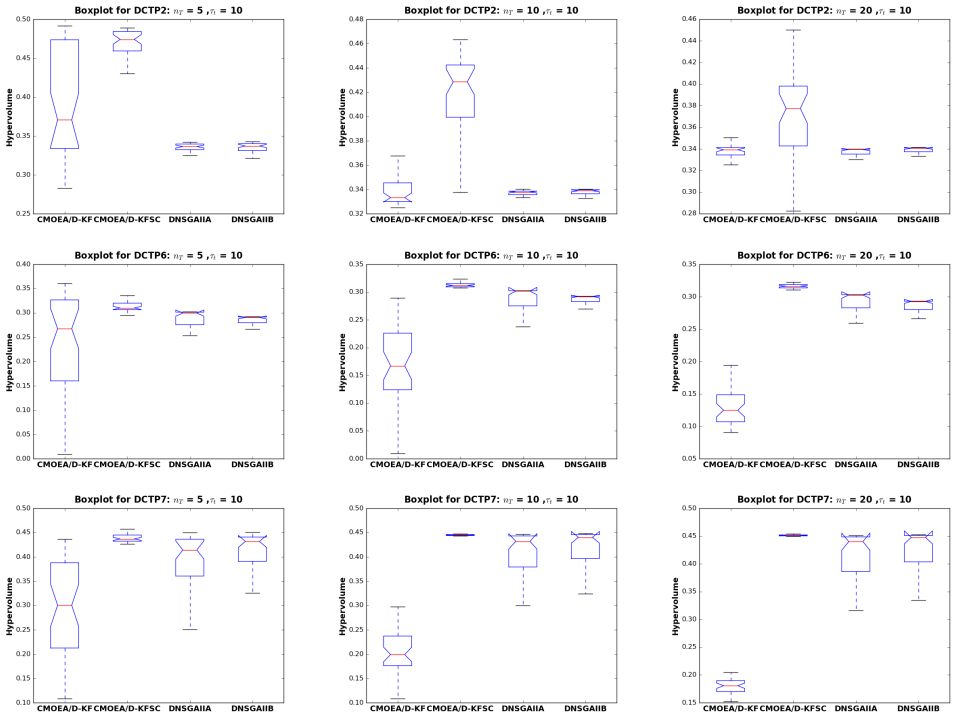


Figure 5.5: Influency of severity of change in DCTP2, DCTP6 and DCTP7

higher. In this section, the severity of change, n_T is provided with 3 values - $5, 10$ and 20. The frequency of change remains constant at a value of 10. The box-plots of MHPV values for CMOEA/D-KF variants and DNSGAII variants are provided in Figures 5.5 and 5.6.

It can be observed in Figure 5.5 that even with increasing problem difficulty from right to left, CMOEA/D-KFSC performs better than the DNSGAII variants.

In DCTP3-5 problems results in the performance comparison section, the DNSGAII variants performed better than that of the proposed algorithms for problem parameters, $n_T = 10$ and $\tau_T = 10$. However, when problem difficulty increases with decreasing n_T value, the proposed algorithms perform better as can be observed in Figure 5.6.

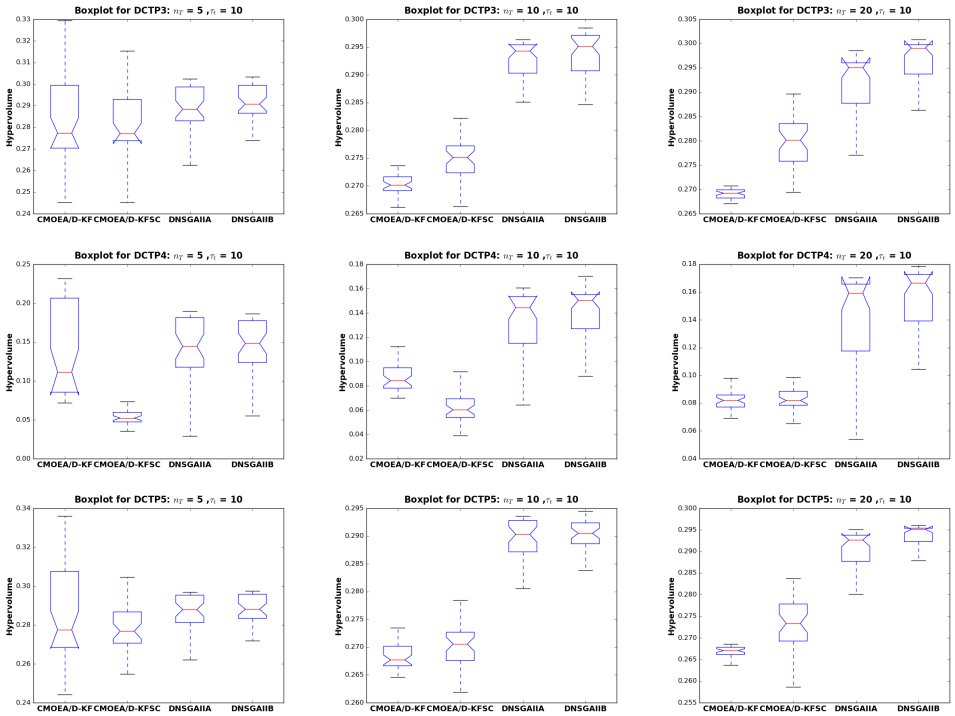
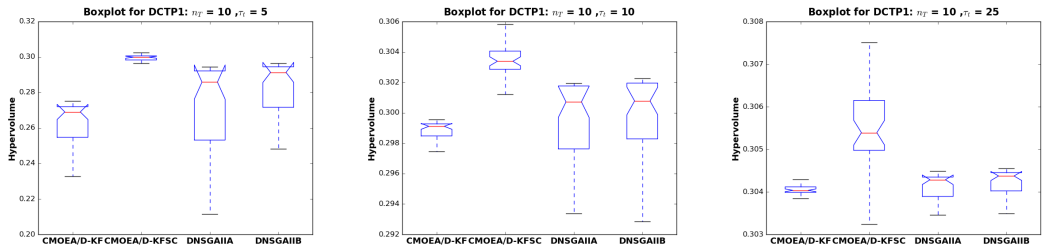


Figure 5.6: Influency of severity of change in DCTP3, DCTP4 and DCTP5

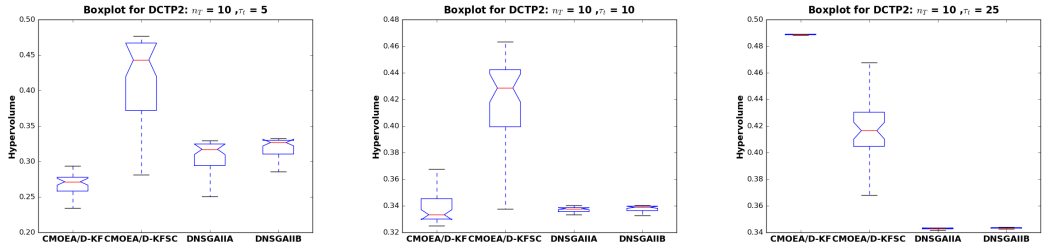
5.6.2 Influence of frequency of change

When the problem changes often, the difficulty posed increases as the DMOEA has to swiftly adapt to the new environment and track the time-varying solutions. This difficulty is compounded in DCMOPs as the constraint handling mechanism needs to find feasible as well as optimal solutions quickly to attain fast convergence which is essential in dynamic scenarios. In this subsection, the frequency of change parameter, τ_T is varied and takes three values - 5, 10, and 25, while the severity of change parameter, n_T is maintained at a constant value of 10.

For DCTP1, CMOEA/D-KFSC provides the best performance for all values of τ_T. It can be observed in Figure 5.7a that the performance of all algorithms improves as the problem difficulty decreases, as can be expected. When problem changes more frequently for DCTP2, i.e. lower value of τ_T, the scoring scheme based method CMOEA/D-KFSC performs highly



(a) DCTP1



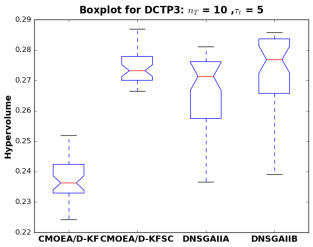
(b) DCTP2

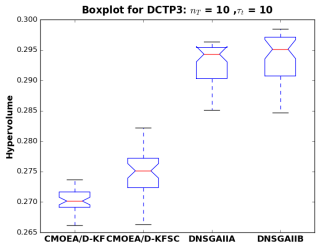
Figure 5.7: Influence of Frequency of Change in DCTP1 and DCTP2

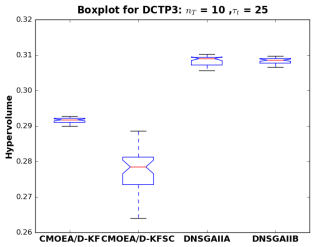
competitively. However, when changes are well-separated in the time scale, i.e. they occur less frequently, the standalone Kalman Filter based method CMOEA/D-KF performs significantly better than the other algorithms (refer to Figure 5.7b).

DNSGAII variants performed better than the proposed algorithms for problems, DCTP3, DCTP4 and DCTP5, for the experiment setting of $n_T = 10$ and $\tau_T = 10$. However, in Figure 5.8a it can be observed that CMOEA/D-KFSC performs better than the DNSGAII variants for lower value of frequency of change parameter, τ_T. Also, it can be observed in Figures 5.8b and 5.8c, the proposed algorithms perform continuously better with increasing problem difficulty compared to the DNSGAII variants.

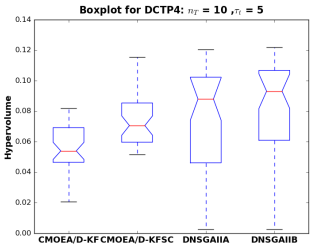
In problems DCTP6 and DCTP7, the proposed algorithms continue to provide best performance for all values of frequency of change. The variance of Hypervolume values (indicated by the vertical length of the box plot) for CMOEA/D-KFSC does not fluctuate much for changing problem difficulty indicating better robustness, compared to the DNSGAII variants whose

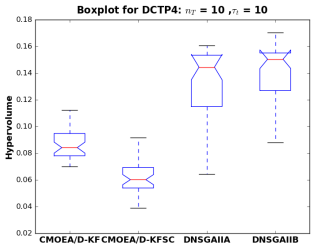


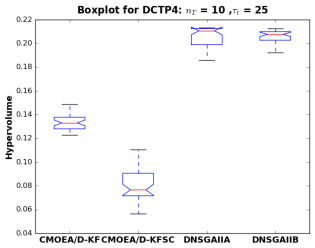




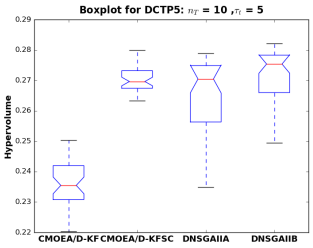
(a) DCTP3

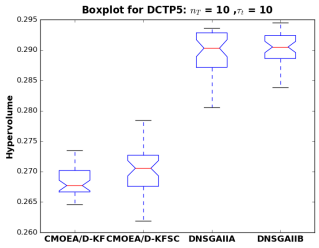


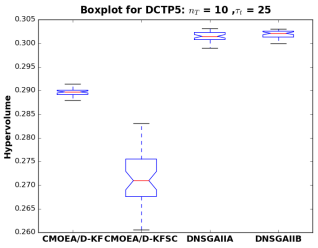




(b) DCTP4

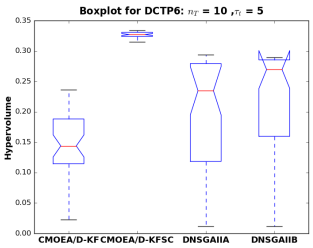


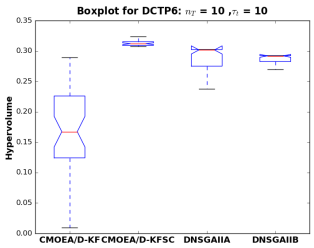


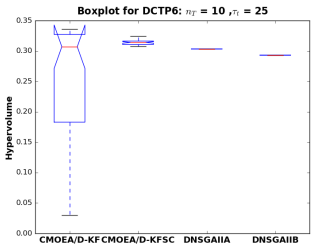


(c) DCTP5

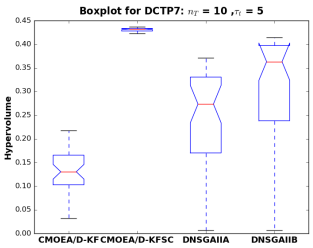
Figure 5.8: Influence of Frequency of Change in DCTP3-5

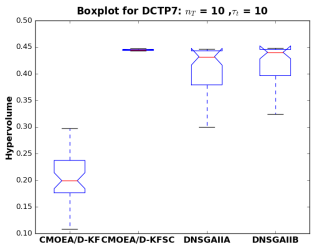


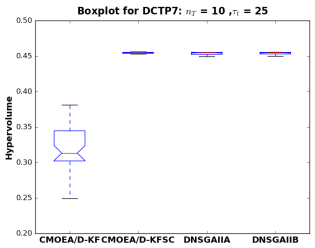




(a) DCTP6







(b) DCTP7

Figure 5.9: Influence of Frequency of Change in DCTP6 and DCTP7

variance increases substantially with increasing problem difficulty.

5.7 Chapter Conclusion

In this chapter, dynamic multiobjective optimization in the presence of constraints has been explored. There has been some amount of work in DMO but only in unconstrained or boundary constrained situations. There is a lack of algorithms and benchmark problems in the field of constrained DMO. In this chapter, the various related works in the literature on constraint handling mechanisms have been reviewed. Further, an existing static constrained multiobjective optimization benchmark set has been modified to make it a dynamic benchmark set. An adaptive threshold based constraint handling has been proposed which has been embedded in a number of DMO algorithms and their performance has been tested on the test benchmark problems. The performance compared against Dynamic NSGA-II algorithm variants is encouraging as the proposed algorithms perform better in four out of the seven benchmark problems. Furthermore, the proposed algorithms have better robustness and tend to perform better with increasing problem difficulty.

Algorithm 5.1 MOEA/D-DE with Kalman Filter prediction and adaptive constraint handling for Constrained Dynamic Multiobjective Optimization

Require:

MOP
A stopping criterion
N: Population size
P: the number of subproblems considered in MOEA/D
A uniform spread of N weight vectors: $\lambda^1, \lambda^2, \ldots, \lambda^P$
T: neighbourhood size
Kalman Filter Parameters

Ensure:

Approximated POF $\{f^1, \ldots, f^N\}$
Approximated POS $\{x^1, \ldots, x^N\}$

Step 1 ➢ Initialization:

1. Generate evenly spread weight vectors. Initialize the neighbourhood of each vector by finding its T closest weight vectors in terms of Euclidean distance.
2. Generate an initial population, $\mathbf{x}^1, \ldots, \mathbf{x}^N$ by uniform random initialization within the decision space. Evaluate objective function values of each solution and set $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i)$.
3. Initialize Kalman Filter matrices and vectors for each solution. Initial population decision variables are set as the initial state of the Kalman Filter.
4. Initialize ideal vector by setting $z_k = \min_{j=1,\ldots,N} f_k^j$ where $k = 1. \ldots, m$
5. Randomly initialize a set of detector individuals within the decision space for change detection.

Step 2 ➢ Update:

1. **Change Detection:** Evaluate the objective function values of the detector individuals and check whether they have changed to indicate a change in the dynamic multiobjective problem. If change is detected go to **Step 2.2**. Otherwise, go to **Step 2.4**
2. If scoring scheme based model, iterate through the steps in Algorithm 3.1
3. If Kalman Filter prediction, perform
   (a) Measurement Update
   (b) Time Update

   Else, if RND perform random reinitialization, or if HYP, perform hypermutation.
4. Reproduction: Mating selection, and Differential Evolution as per normal MOEA/D

Update offspring population with parent population:

1. Compute the mean constraint violation of the population, CV_{mean}

$$CV_{mean} = \frac{\sum_{i=1}^{N} CV_i}{N},$$

where N is population size, and CV_i denotes constraint violation value of individual i.

2. Calculate feasibility ratio, fr of current population

$$fr = \frac{\text{Number of feasible individuals}}{\text{Total Population size}}$$

3. Compute adaptive constraint threshold value, CV_δ for current generation update

$$CV_\delta = fr \times CV_{mean}$$

4. Survivor selection between parent solution, i and child solution, j
   (a) If $CV_i < CV_\delta$ and $CV_j < CV_\delta$, compare individuals based on fitness value using tchebycheff approach
   (b) Else if $CV_i = 0$ and $CV_j > CV_\delta$, choose i
   (c) Else if $CV_j = 0$ and $CV_i > CV_\delta$, choose j
   (d) Else, choose the individual with smaller constraint violation value.
5. Update ideal vector

Step 3 ➢ Stopping Criteria: If stopping criteria is satisfied, then stop and output the final population and their corresponding values in the objective space. Otherwise, go to **Step 2**.

Chapter 6

Conclusions & Directions for Future Research

6.1 Conclusions

Evolutionary Multiobjective Optimization has been explored vastly in the literature. However, there is a research gap to tackle Multiobjective Optimization in dynamic environments using Evolutionary Algorithms. The primary aim of this thesis is to aim to contribute in this research gap by proposing algorithms to tackle Dynamic Multiobjective Optimization problems in unconstrained as well as constrained environments using Prediction techniques and an adaptive constraint handling mechanism embedded in MOEA/D-DE. In chapter 3, a linear Kalman Filter based prediction method was built on the MOEA/D framework to tackle DMOPs. A scoring scheme mechanism was designed to hybridize the Kalman Filter prediction with random reinitialization. The experiment results on the FDA [2], dMOP [26] and F [21] benchmark suite showed that the proposed algorithm shows improved performances on a number of test benchmark problems. The proposed algorithm is able to swiftly predict the time-varying optimal

solutions without needing any additional learning time. One drawback, however, was that the linear Kalman Filter is not fully capable to tackle non-linear problems.

Support Vector Regression, a non-linear prediction mechanism which is data-driven, was considered in Chapter 4. A time series formed by near-optimal solutions obtained by the Evolutionary Algorithm in previous changes, is formulated as training data. Support Vector Machines are used in tandem with the Evolutionary Algorithm to predict subsequent optimal solutions for future generations from the time series, when a change in the environment is detected. The proposed algorithm, MOEA/D-SVR tends to perform well in complicated problems where the environment does not change smoothly from one time instant to the next. MOEA/D-SVR's performance also improves with increase in severity of change. Analysis of the parameter selection module through a heatmap visualization is capable of providing insights for linkages in the decision variable feature space.

Chapter 5 outlines an adaptive threshold based constraint handling mechanism combined with Dynamic Multiobjective Optimization techniques to handle DMOPs in constrained scenarios. The proposed algorithm performs competitively with the existing state-of-the-art and shows improved performances with increase in the difficulty of the problem resulting from increase in severity and frequency of change.

6.2 Directions for Future Research

The Kalman Filter prediction based DMOEA proposed in Chapter 3 showed significantly improved dynamic optimization performance in a number of benchmark problems. Two formulations were proposed - the first one, 2by2 considered only in the first order change in decision variables, while

the second one, 3by3 considered both first and secord order change. The 3by3 formulation showed better performance than the former, and this was further improved by hybridization with the diversity introduction technique of random reinitialization using the scoring scheme mechanism. There is still scope for improvement by adapting the process noise and observation noise matrices which currently take on empirically chosen values from a Gaussian distribution. An adaptive Kalman Filter formulation may be able to show improved performances by taking on problem-dependent parameter values for the matrices.

Hybridization of the Kalman Filter prediction and Support Vector Regression methods through ensembling could also result in taking advantage of both their strengths. Diversity maintenance throughout the evolutionary process could also be crucial in problems which especially test this aspect. The prediction mechanisms could be further strengthened by combining with a diversity maintenance mechanism such as online diversity assessment at each generation.

The scoring scheme mechanism used distance travelled by DMOEA immediately after a change to just before subsequent change as a measure of the prediction mechanism's performance. Indicator-based algorithm formulation could also be embedded, wherein the Hypervolume is employed to evaluate the performance. Other Multiobjective Optimization frameworks such as the Multipopulation approach may also be considered.

The DCMOPs in Chapter 5 consisted of problems whose Pareto Optimal Solutions change with time. The work can be extended to consider scenarios in which the constraints may change with time as well.

Bibliography

[1] Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[2] Marco Farina, Kalyanmoy Deb, and Paolo Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Trans. Evolutionary Computation*, 8(5):425–442, 2004.

[3] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6(0):1 – 24, 2012.

[4] Jrn Mehnen, Tobias Wagner, and Gnter Rudolph. Evolutionary optimization of dynamic multiobjective functions. Technical report, 2006.

[5] Helen G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5514, Washington, D.C. 20375-5320, 1990.

[6] F. Vavak, K. Jukes, and T.C. Fogarty. Learning the local search range for genetic optimisation in nonstationary environments. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 355–360, Apr 1997.

[7] Kalyanmoy Deb, UdayaBhaskara Rao N., and S. Karthik. Dynamic multi-objective optimization and decision-making using modified nsga-ii: A case study on hydro-thermal power scheduling. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 803–817. Springer Berlin Heidelberg, 2007.

[8] John Grefenstette. Genetic algorithms for changing environments. In *Parallel Problem Solving from Nature 2*, pages 137–144. Elsevier, 1992.

[9] H. C. Andersen. An Investigation into Genetic Algorithms, and the Relationship between Speciation and the Tracking of Optima in Dynamic Functions. Honours thesis, Queensland University of Technology, Brisbane, Australia, 1991.

[10] Naoki Mori, Hajime Kita, and Yoshikazu Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In AgostonE. Eiben, Thomas Bck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 149–158. Springer Berlin Heidelberg, 1998.

[11] Shengxiang Yang and Xin Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.*, 9(11):815–834, November 2005.

[12] Helen G. Cobb. Genetic algorithms for tracking changing environments. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 523–530. Morgan Kaufmann, 1993.

[13] Iason Hatzakis and David Wallace. Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *in: Genetic and Evolutionary Computation Conference, GECCO, ACM*, pages 1201–1208. Press, 2006.

[14] Aimin Zhou, Yaochu Jin, Qingfu Zhang, Bernhard Sendhoff, and Edward P. K. Tsang. Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In *EMO*, pages 832–846, 2006.

[15] Anabela Simes and Ernesto Costa. Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains. In Gnter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 306–315. Springer Berlin Heidelberg, 2008.

[16] Anabela Simões and Ernesto Costa. Improving prediction in evolutionary algorithms for dynamic environments. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 875–882. ACM, 2009.

[17] WeeTat Koo, ChiKeong Goh, and KayChen Tan. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. *Memetic Computing*, 2(2):87–110, 2010.

[18] P. Amato and M. Farina. An alife-inspired evolutionary algorithm for dynamic multiobjective optimization problems. In Frank Hoffmann, Mario Kppen, Frank Klawonn, and Rajkumar Roy, editors, *Soft Computing: Methodologies and Applications*, volume 32 of *Advances in Soft Computing*, pages 113–125. Springer Berlin Heidelberg, 2005.

[19] Mard Helbig and AndriesP. Engelbrecht. Dynamic multi-objective optimization using pso. In Enrique Alba, Amir Nakib, and Patrick Siarry, editors, *Metaheuristics for Dynamic Optimization*, volume 433 of *Studies in Computational Intelligence*, pages 147–188. Springer Berlin Heidelberg, 2013.

[20] Shuzhen Wan, Shengwu Xiong, and Yi Liu. Prediction based multi-strategy differential evolution algorithm for dynamic environments. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[21] Aimin Zhou, Yaochu Jin, and Qingfu Zhang. A population prediction strategy for evolutionary dynamic multiobjective optimization. *IEEE T. Cybernetics*, 44(1):40–53, 2014.

[22] Qingfu Zhang, Aimin Zhou, and Yaochu Jin. Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm. *Evolutionary Computation, IEEE Transactions on*, 12(1):41–63, 2008.

[23] J.J. Grefenstette. Evolvability in dynamic fitness landscapes: a genetic algorithm approach. In *Evolutionary Computation, 1999. CEC 99.*

Proceedings of the 1999 Congress on, volume 3, pages –2038 Vol. 3, 1999.

[24] Rasmus K. Ursem. Multinational gas: Multimodal optimization techniques in dynamic environments. In *In Proceedings of the Second Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2000.

[25] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages –1882 Vol. 3, 1999.

[26] Chi-Keong Goh and K. Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 13(1):103–127, Feb 2009.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[28] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evolutionary Computation*, 11(6):712–731, 2007.

[29] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *Trans. Evol. Comp*, 13(2):284–302, April 2009.

[30] Zhenan He, G.G. Yen, and Jun Zhang. Fuzzy-based pareto optimality for many-objective evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 18(2):269–285, April 2014.

[31] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18(4):577–601, Aug 2014.

[32] Jixiang Cheng, G.G. Yen, and Gexiang Zhang. A many-objective evolutionary algorithm with enhanced mating and environmental selections. *Evolutionary Computation, IEEE Transactions on*, 19(4):592–605, Aug 2015.

[33] K. Li, K. Deb, Q. Zhang, and S. Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *Evolutionary Computation, IEEE Transactions on*, 19(5):694–716, Oct 2015.

[34] Handing Wang, Licheng Jiao, and Xin Yao. Two arch2: An improved two-archive algorithm for many-objective optimization. *Evolutionary Computation, IEEE Transactions on*, 19(4):524–541, Aug 2015.

[35] P.C. Roy, M.M. Islam, K. Murase, and Xin Yao. Evolutionary path control strategy for solving many-objective optimization problem. *Cybernetics, IEEE Transactions on*, 45(4):702–715, April 2015.

[36] H. Ishibuchi, N. Akedo, and Y. Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *Evolutionary Computation, IEEE Transactions on*, 19(2):264–283, April 2015.

[37] J.E. Fieldsend and R.M. Everson. The rolling tide evolutionary algorithm: A multiobjective optimizer for noisy optimization problems. *Evolutionary Computation, IEEE Transactions on*, 19(1):103–117, Feb 2015.

[38] Chi-Keong Goh and KayChen Tan. Dynamic evolutionary multi-objective optimization. In *Evolutionary Multi-objective Optimization in Uncertain Environments*, volume 186 of *Studies in Computational Intelligence*, pages 125–152. Springer Berlin Heidelberg, 2009.

[39] PeterA.N. Bosman. Learning and anticipation in online dynamic optimization. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 129–152. Springer Berlin Heidelberg, 2007.

[40] Jurgen Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *Proceeding of GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, 2001.

[41] Su Nguyen, Mengjie Zhang, M. Johnston, and Kay Chen Tan. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *Evolutionary Computation, IEEE Transactions on*, 18(2):193–208, April 2014.

[42] S. Palaniappan, S. Zein-Sabatto, and A. Sekmen. Dynamic multiobjective optimization of war resource allocation using adaptive genetic algorithms. In *SoutheastCon 2001. Proceedings. IEEE*, pages 160–165, 2001.

[43] Peng Xingguang, Xu Demin, and Zhang Fubin. Uav online path planning based on dynamic multiobjective evolutionary algorithm. In *Control Conference (CCC), 2011 30th Chinese*, pages 5424–5429, July 2011.

[44] M. Camara, J. Ortega, and F.J. Toro. Parallel processing for multi-objective optimization in dynamic environments. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8, March 2007.

[45] Hongfeng Wang, Dingwei Wang, and Shengxiang Yang. A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing*, 13(8-9):763–780, 2009.

[46] Mario Camara, Julio Ortega, and Francisco de Toro. A single front genetic algorithm for parallel multi-objective optimization in dynamic environments. *Neurocomputing*, 72(1618):3570 – 3579, 2009. Financial Engineering Computational and Ambient Intelligence (IWANN 2007).

[47] A. Kiruluta, E. Eizenman, and S. Pasupathy. Predictive head movement tracking using a kalman filter. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(2):326–331, 1997.

[48] J.M. del Rincon, D. Makris, C.O. Uruuela, and J.-C. Nebel. Tracking human position and lower body parts using kalman and particle filters constrained by human biomechanics. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(1):26–37, Feb 2011.

[49] W.S. Chaer, R.H. Bishop, and J. Ghosh. A mixture-of-experts framework for adaptive kalman filtering. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(3):452–464, Jun 1997.

[50] Jianguo Yan, Dongli Yuan, Xiaojun Xing, and Qiuling Jia. Kalman filtering parameter optimization techniques based on genetic algorithm.

In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 1717–1720, Sept 2008.

[51] P.D. Stroud. Kalman-extended genetic algorithm for search in non-stationary environments with noisy fitness evaluations. *Evolutionary Computation, IEEE Transactions on*, 5(1):66–77, Feb 2001.

[52] John Lyons and Hadi Nasrabadi. Well placement optimization under time-dependent uncertainty using an ensemble kalman filter and a genetic algorithm. *Journal of Petroleum Science and Engineering*, 109(0):70 – 79, 2013.

[53] Fengyun Qiu, Yong Wang, Mingyan Jiang, and Dongfeng Yuan. Adaptive image restoration based on the genetic algorithm and kalman filtering. In De-Shuang Huang, Laurent Heutte, and Marco Loog, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*, volume 2 of *Communications in Computer and Information Science*, pages 742–750. Springer Berlin Heidelberg, 2007.

[54] Claudio Rossi, Mohamed Abderrahim, and Julio César Díaz. Tracking moving optima using kalman-based predictions. *Evol. Comput.*, 16(1):1–30, March 2008.

[55] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[56] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[57] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997.

[58] Dan Simon. *Optimal State Estimation: Kalman, H, and Nonlinear Approaches*, chapter The particle filter, pages 461–483. John Wiley & Sons, Inc., 2006.

[59] Lam Thu Bui, H.A. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2349–2356 Vol. 3, Sept 2005.

[60] Changhe Li, Shengxiang Yang, and Ming Yang. An adaptive multi-swarm optimizer for dynamic optimization problems. *Evol. Comput.*, 2014.

[61] Jrgen Branke, Thomas Kaussler, Christian Smidt, and Hartmut Schmeck. A multi-population approach to dynamic optimization problems. In I.C. Parmee, editor, *Evolutionary Design and Manufacture*, pages 299–307. Springer London, 2000.

[62] S. Das, A. Mandal, and R. Mukherjee. An adaptive differential evolution algorithm for global optimization in dynamic environments. *Cybernetics, IEEE Transactions on*, 44(6):966–978, June 2014.

[63] Sang you Zeng, Guang Chen, Liang Zheng, Hui Shi, H. De Garis, Lixin Ding, and Lishan Kang. A dynamic multi-objective evolutionary algorithm based on an orthogonal design. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 573–580, 2006.

[64] M. Helbig and A.P. Engelbrecht. Analysing the performance of dynamic multi-objective optimisation algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1531–1539, June 2013.

[65] Jrgen Branke. Optimization in dynamic environments. In *Evolutionary Optimization in Dynamic Environments*, volume 3 of *Genetic Algorithms and Evolutionary Computation*, pages 13–29. Springer US, 2002.

[66] Jrgen Branke. Survey: State of the art. In *Evolutionary Optimization in Dynamic Environments*, volume 3 of *Genetic Algorithms and Evolutionary Computation*, pages 31–52. Springer US, 2002.

[67] Yaochu Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3):303–317, 2005.

[68] Ronald W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. SpringerVerlag, 2004.

[69] Arrchana Muruganantham, Yang Zhao, Sen Bong Gee, Xin Qiu, and Kay Chen Tan. Dynamic multiobjective optimization using evolutionary algorithm with kalman filter. *Procedia Computer Science*, 24(0):66 – 75, 2013. 17th Asia Pacific Symposium on Intelligent and Evolutionary Systems, {IES2013}.

[70] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132, April 2003.

[71] O. Schutze, X. Esquivel, A Lara, and Carlos A Coello Coello. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 16(4):504–522, Aug 2012.

[72] Kanpur Genetic Algorithms Laboratory. Software developed at kangal.

[73] V.A. Shim, Kay Chen Tan, and K.K. Tan. A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[74] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[75] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297.

[76] V. Vapnik and A. Lerner. *Pattern recognition using generalized portrait method*. Automation and Remote Control, 24:774780, 1963.

[77] K.-R. Mller, A.J. Smola, G. Rtsch, B. Schlkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks ICANN'97*, volume 1327 of *Lecture Notes in Computer Science*, pages 999–1004. Springer Berlin Heidelberg, 1997.

[78] A. Muruganantham, K. C. Tan, and P. Vadakkepat. Evolutionary dynamic multiobjective optimization via kalman filter prediction. *IEEE Transactions on Cybernetics*, 46(12):2862–2873, Dec 2016.

[79] Vladimir Cherkassky and Yunqian Ma. Practical selection of {SVM} parameters and noise estimation for {SVM} regression. *Neural Networks*, 17(1):113 – 126, 2004.

[80] L. J. Cao and F. E. H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, Nov 2003.

[81] N.I. Sapankevych and Ravi Sankar. Time series prediction using support vector machines: A survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38, May 2009.

[82] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[83] P.-S. Yu, S.-T. Chen, and I-F. Chang. *Practical Hydroinformatics: Computational Intelligence and Technological Developments in Water Applications*, chapter Real-Time Flood Stage Forecasting Using Support Vector Regression, pages 359–373. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[84] Y. H. Liu and Y. T. Chen. Face recognition using total margin-based adaptive fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 18(1):178–192, Jan 2007.

[85] H. Drucker, Donghui Wu, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, Sep 1999.

[86] Z. Shi and M. Han. Support vector echo-state machine for chaotic time-series prediction. *IEEE Transactions on Neural Networks*, 18(2):359–372, March 2007.

[87] T. Van Gestel, J. A. K. Suykens, D. E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Financial

time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks*, 12(4):809–821, Jul 2001.

[88] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.

[89] T. B. Trafalis and H. Ince. Support vector machine for regression and applications to financial forecasting. *in Proc. IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks 2000 (IJCNN*, 6:348–353, 2000.

[90] Francis E.H Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309 – 317, 2001.

[91] H. Yang, L. Chan, and I. King. *Support vector machine regression for volatile stock market prediction.* in Proc, 2002.

[92] T. B. Trafalis, B. Santosa, and M. B. Richman. prediction of rainfall from WSR- 88D radar using kernel-based methods. *Int. J. Smart Eng. Syst. Design, vol. 5, no*, 5(4):429438, 2003.

[93] W. Lu, W. Wang, A. Y. T. Leung, and S. M. Lo. *R. K. K*, 1:630635, 2002.

[94] L. Tian and A. Noore. a novel approach for short-term load forecasting using support vector machines. *Int. J. Neural Syst., vol. 14, no*, 14(5):329335, August 2004.

[95] P. F. Pai and W. C. Hong. forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. *Electric Power Syst. Res., vol. 74, no*, 74(3):417425, 2005.

[96] M. G. Zhang. short-term load forecasting based on support vector machine regression. *in Proc*, 7:43104314, 2005.

[97] C. C. Hsu, C. H. Wu, S. C. Chen, and K. L. Peng. dynamically optimizing parameters in support vector regression: An application of electricity load forecasting. *in Proc. 39th Annu. Hawaii Int. Conf. on System Sciences (HICSS*, 2:18, 2006.

[98] J. Yang and Y. Zhang. application research of support vector machines in condition trend prediction of mechanical equipment. *in Proc. 2nd Int. Symp. on Neural Networks (ISNN*, 3498:857864, 2005.

[99] S. Gezici, H. Kobayashi, and H. V. Poor. *A new approach to mobile position tracking.* in Proc, 2003.

[100] Q. Yang and S. Xie. an application of support vector regression on narrow-band interference suppression in spread spectrum systems. *Lect*, 3611:442450, August 2005.

[101] K. Ikeda. Effects of kernel function on nu support vector machines in extreme cases. *IEEE Transactions on Neural Networks*, 17(1):1–9, Jan 2006.

[102] Lijuan Cao Francis E.H. Tay. Application of support vector machines in financial time series forecasting. Journal, Omega Tge Internation Journal of Management Science, 2001.

[103] Bo-Juen Chen, Ming-Wei Chang, and Chih-Jen Lin. Load forecasting using support vector machines: a study on eunite competition 2001. *Power Systems, IEEE Transactions on*, 19(4):1821–1830, Nov 2004.

[104] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[105] Kalyanmoy Deb, Amrit Pratap, and T. Meyarivan. *Constrained Test Problems for Multi-objective Evolutionary Optimization*, pages 284–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[106] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, June 2009.

[107] Trung Thanh Nguyen and Xin Yao. *Evolutionary Optimization on Continuous Dynamic Constrained Problems - An Analysis*, pages 193–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[108] Zbigniew Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary Programming*, 1995.

[109] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(1112):1245 – 1287, 2002.

[110] Y. G. Woldesenbet, B. G. Tessema, and G. G. Yen. Constraint handling in multi-objective evolutionary optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 3077–3084, Sept 2007.

[111] S. Venkatraman and G. G. Yen. A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424–435, Aug 2005.

[112] Abdollah Homaifar, Charlene X. Qi, and Steven H. Lai. Constrained optimization via genetic algorithms. *SIMULATION: Transactions of The Society for Modeling and Simulation*, April 1994.

[113] David W Coit and Alice E Smith. Penalty guided genetic search for reliability design optimization. *Computers & Industrial Engineering*, 30(4):895–904, September 1996.

[114] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput.*, 4(1):1–32, March 1996.

[115] J. A. Joines and C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 579–584 vol.2, Jun 1994.

[116] S. Kazarlis and Vassilios Petridis. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, pages 211–220, London, UK, UK, 1998. Springer-Verlag.

[117] M. A. Jan and Q. Zhang. Moea/d for constrained multiobjective optimization: Some preliminary experimental results. In *2010 UK Workshop on Computational Intelligence (UKCI)*, pages 1–6, Sept 2010.

[118] M. Asafuddoula, T. Ray, R. Sarker, and K. Alam. An adaptive constraint handling approach embedded moea/d. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, June 2012.

[119] B. Tessema and G. G. Yen. A self adaptive penalty function based algorithm for constrained optimization. In *2006 IEEE International Conference on Evolutionary Computation*, pages 246–253, 2006.

[120] D. Orvosh and L. Davis. Using a genetic algorithm to optimize problems with feasibility constraints. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 548–553 vol.2, Jun 1994.

[121] Zbigniew Michalewicz and Cezary Z. Janikow. Handling constraints in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1993.

[122] Zbigniew Michalewicz and Girish Nazhiyath. Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints.

[123] F. Jimenez, A. F. Gomez-Skarmeta, G. Sanchez, and K. Deb. An evolutionary algorithm for constrained multi-objective optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1133–1138, 2002.

[124] Z. Cai and Y. Wang. A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 10(6):658–675, Dec 2006.

[125] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An evolutionary algorithm for constrained optimization. In *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*, GECCO'00, pages 771–777, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[126] T. P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, Sep 2000.

[127] T. P. Runarsson and Xin Yao. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(2):233–243, May 2005.

[128] B. Y. Qu and P. N. Suganthan. Constrained multi-objective optimization algorithm with an ensemble of cosntraint handling methods. *Engineering Optimization*, 43(4), 2011.

[129] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. Multi-objective optimization with dynamic constraints and objectives: New challenges for evolutionary algorithms. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 615–622, New York, NY, USA, 2015. ACM.

[130] Zhuhong Zhang and Shuqu Qian. Artificial immune system in dynamic environments solving time-varying non-linear constrained multi-objective problems. *Soft Computing*, 15(7):1333–1349, 2011.

[131] Zhuhong Zhang, Min Liao, and Lei Wang. Immune optimization approach for dynamic constrained multi-objective multimodal optimization problems. *American Journal of Operations Research*, 2(2):193–202, June 2012.

[132] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.

[133] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Theory of the hypervolume indicator: Optimal mu-distributions and the choice of the reference point. In *Proceedings of the Tenth ACM*

SIGEVO Workshop on Foundations of Genetic Algorithms, FOGA '09, pages 87–102, New York, NY, USA, 2009. ACM.

[134] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theor. Comput. Sci.*, 425:75–103, March 2012.

[135] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. *The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration*, pages 862–876. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[136] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007.

[137] Dimo Brockhoff, Tobias Friedrich, and Frank Neumann. *Analyzing Hypervolume Indicator Based Algorithms*, pages 651–660. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[138] Christian Igel, Nikolaus Hansen, and Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1):1–28, March 2007.

[139] Hisao Ishibuchi, Noritaka Tsukamoto, Yuji Sakane, and Yusuke Nojima. Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 527–534, New York, NY, USA, 2010. ACM.

[140] Eckart Zitzler and Simon Künzli. *Indicator-Based Selection in Multiobjective Search*, pages 832–842. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[141] L. Bradstreet, L. While, and L. Barone. A fast incremental hypervolume algorithm. *IEEE Transactions on Evolutionary Computation*, 12(6):714–723, Dec 2008.

[142] Nicola Beume. S-metric calculation by considering dominated hypervolume as klee's measure problem. *Evol. Comput.*, 17(4):477–492, December 2009.

[143] L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, Feb 2012.

[144] Luís M. S. Russo and Alexandre P. Francisco. Extending quick hypervolume. *Journal of Heuristics*, 22(3):245–271, 2016.

[145] Yongtao Cao, Byran J. Smucker, and Timothy J. Robinson. On using the hypervolume indicator to compare pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference*, January 2015.

[146] Muhammad Asif Jan and Rashida Adeeb Khanum. A study of two penalty-parameterless constraint handling techniques in the framework of moea/d. *Applied Soft Computing*, 13(1):128 – 148, 2013.

[147] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng. An adaptive tradeoff model for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 12(1):80–92, Feb 2008.