UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation

by

Mardé Helbig

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

May 2012

Publication data:

Mardé Helbig. Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. Doctorate Thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, May 2012.

Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation

by

Mardé Helbig
E-mail: mhelbig@csir.co.za

Abstract

Most optimisation problems in everyday life are not static in nature, have multiple objectives and at least two of the objectives are in conflict with one another. However, most research focusses on either static multi-objective optimisation (MOO) or dynamic single-objective optimisation (DSOO). Furthermore, most research on dynamic multi-objective optimisation (DMOO) focusses on evolutionary algorithms (EAs) and only a few particle swarm optimisation (PSO) algorithms exist. This thesis proposes a multi-swarm PSO algorithm, dynamic Vector Evaluated Particle Swarm Optimisation (DVEPSO), to solve dynamic multi-objective optimisation problems (DMOOPs). In order to determine whether an algorithm solves DMOO efficiently, functions are required that resembles real world DMOOPs, called benchmark functions, as well as functions that quantify the performance of the algorithm, called performance measures. However, one major problem in the field of DMOO is a lack of standard benchmark functions and performance measures. To address this problem, an overview is provided from the current literature and shortcomings of current DMOO benchmark functions and performance measures are discussed. In addition, new DMOOPs are introduced to address the identified shortcomings of current benchmark functions. Guides guide the optimisation process of DVEPSO. Therefore, various guide update approaches are investigated. Furthermore, a sensitivity analysis of DVEPSO is conducted to determine the influence of various parameters on the performance of DVEPSO. The investigated parameters include approaches to manage boundary constraint violations, approaches to share knowledge between the sub-swarms and responses to changes in the environment that are applied to either the particles

of the sub-swarms or the non-dominated solutions stored in the archive. From these experiments the best DVEPSO configuration is determined and compared against four state-of-the-art DMOO algorithms.

Keywords: dynamic multi-objective optimisation, particle swarm optimisation, dynamic vector evaluated particle swarm optimisation algorithm, benchmark functions, performance measures, guide updates, management of boundary constraint violations, response strategies, knowledge sharing

Supervisor : Prof. A. P. Engelbrecht
Department : Department of Computer Science
Degree : Philosophiae Doctor

"You have to be fast on your feet and adaptive or else a strategy is useless."
– Charles de Gaulle

Acknowledgements

I would like to thank the following people and organisations who played a vital role during the research and write-up of the thesis:

- Prof. Andries Engelbrecht, my supervisor, for his continuous guidance, inspiration, patience and assistance during the entire journey of my PhD.
- My husband, Lutz, for his endless love, support, encouragement and patience throughout the many years it took to complete the thesis.
- My mom, for her love and encouragement and for always believing in me.
- My friends that brought endless cups of coffee, advice, encouragement and laughter to lift my spirits, and that walked with me throughout the many ups and downs. Thank you for being my sound board and allowing me to vent my frustration when it was really necessary. A special thanks to Jacomine, Riëtte, Ronéll, Alize and the coffee time gang (Neeshal, Louis J, Neil and Ralf).
- The Meraka Institute for financing the studies. I would like to thank Hina Patel for her support and providing time for the research during the first years of this endeavour. I also want to extend my gratitude to Keith Fergusson for his advice, encouragement and providing sufficient time to finish the thesis. A warm hearted thank you to Paula Kotzé, my mentor, for her advice and encouragement. I would also like to thank Louis Coetzee for his assistance in setting up the java environment to run my simulations and Thomas Fogwill for his assistance with the mathematical equations. Furthermore, I would like to thank all my collegues who continuously enquired about my progress and provided encouragement along the way.
- The Computational Intelligence Research Group (CIRG), for their advice and assistance, especially with regards to Java progamming. A special thank you to Gary Pamparà, for his endless patience and willingness to assist, even in times when he was swamped with work or studies.
- The Centre for High Performance Computing (CHPC) for the use of their infrastructure to run the simulations and obtaining the required data. A special word of thanks to the technical support team, who provided support when ever problems occurred.

- C.K. Goh for his assistance with the calculation of the true Pareto-optimal fronts (POFs) of the FDA functions and providing the source code of dCOEA . In addition, I would like to thank S. Lechuga who kindly provided the source code of the MOPSO algorithm and K. Deb for making the source code of NSGA-II available on his website.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

> *"What do you want to achieve or avoid? The answers to this question are objectives. How will you go about achieving your desire results? The answer to this you can call strategy."* – *William E Rothschild*

Imagine standing at the airport and looking at the display boards of arriving and departing flights. Suddenly a number of flights are indicated as being delayed, and as you check carefully your flight is one of them. You start to wonder whether you are going to miss your connecting flight and all of the effects that this delay can have on your schedule. However, at the air traffic control room, people start to think about other issues, such as: How will these delays influence the best way of handling all of the incoming and departing aeroplanes? How can they ensure that each plane's waiting time for either landing or take-off is minimised, but in such a way that the possibility of collisions is kept to zero?

The above is just one scenario of an every day life optimisation problem. The issues that the control room have to consider are called objectives. However, these objectives are in conflict with one another: by reducing the possibility of collissions, the waiting time of either landing or departing flights are increased, and vice versa. Furthermore, the delay of flights is an event that causes a change in the environment. Therefore, this is an example of a dynamic multi-objective optimisation problem (DMOOP).

The main objective of this thesis is to propose a new algorithm that solves DMOOPs efficiently.

1.1 Motivation

Most current research in the field of multi-objective optimisation (MOO) focusses on optimisation problems where all of the sub-objectives are static [31, 35, 36, 38]. Research on solving dynamic optimisation problems, on the other hand, strongly focusses on dynamic single-objective optimisation problems (DSOOPs) [11, 13, 37, 89].

However, optimisation problems that occur in situations of everyday life are normally not static in nature and have many objectives that have to be optimised, i.e. DMOOPs. One example of a real-life DMOOP is a steel production plant, where customers place an order for specific products that have to be delivered by a specified date. In order to produce a customer's order, the material has to go through specific production lines. Each production line consists of a number of machines that can only manage a certain load. Since many orders' material is managed in the production lines at the same time, and some orders may require the same machines, the order in which the material of the various orders move through the production line has to be optimised. Since machines can break down, requiring the production lines to be re-optimised, the optimisation of a production plant is an example of a DMOOP.

Multi-objective optimisation problems (MOOPs) with conflicting objectives do not have a single solution. Therefore, MOO algorithms aim to obtain a diverse set of non-dominated solutions, i.e. solutions that balance the trade-off between the various objectives, referred to as the Pareto-optimal front (POF). Another goal of multi-objective algorithms (MOAs) is to find a POF that is as close as possible to the true POF of the problem. Many MOAs store the found non-dominated solutions in an archive. Therefore, if an algorithm finds new non-dominated solutions, the new solutions are compared with the solutions in the archive. If a new solution is dominated by any of the solutions in the archive, it is not placed in the archive. Otherwise, the new solution is placed in the archive and any solutions in the archive that are dominated by the new solution are removed from the archive. When a change in the MOOP occurs, i.e. for example an objective function changes, the solutions in the archive are not necessarily valid for the new objective functions. Furthermore, solutions in the archive that were non-dominated before the change, may have become dominated after the change. Therefore, algorithms solving DMOOPs must have the ability to track the changing POF in order to find non-

dominated solutions that are close to the new true POF, and to remove solutions from the archive that have become dominated after a change occurred in the environment.

Initially not much research has been done on dynamic multi-objective optimisation (DMOO) [1, 58, 117], but in the last few years more researchers focussed on DMOO [2, 17, 46, 67, 96, 100, 129, 135, 165, 156]. However, not much research has been done on solving DMOO using particle swarm optimisation (PSO) [102, 107]. This thesis proposes a new PSO-based DMOO algorithm, namely the dynamic Vector Evaluated Particle Swarm Optimisation (DVEPSO) algorithm.

In order to determine whether an algorithm can solve DMOOPs, functions with specific characteristics that are representative of typical real-world problems are required. These functions are normally referred to as *benchmark functions*. In the field of DMOO, there is a lack of standard benchmark functions and selecting the benchmark functions to test a new DMOO algorithm is not a trivial task. This thesis provides an overview of the benchmark functions that have been proposed in the DMOO literature and proposes new benchmark functions to address the identified limitations of the current DMOOPs. In addition, the characteristics of an ideal benchmark function suite is provided, as well as a list of DMOOPs for each of the identified characteristics.

Functions that quantify the performance of a DMOO algorithm, are referred to as *performance measures* or *performance metrics*. Similar to benchmark functions, there are no standard performance measures for DMOO. Therefore, this thesis provides an overview of the performance measures that are currently used to measure the performance of DMOO algorithms. Furthermore, issues with current DMOO performance measures are discussed.

1.2 Objectives

The primary objective of this thesis is to develop a PSO MOA for solving DMOOPs, namely DVEPSO. In achieving this main objective, the following sub-objectives have been identified:

- Identifying a set of benchmark functions representative of typical real-world problems.

- Identifying a set of performance measures that adequately quantifies the performance of a DMOO algorithm.
- The development and analysis of DVEPSO.

1.3 Contributions

The contributions of this thesis with regards to DMOOPs and performance measures for DMOO are:

- A comprehensive overview of the benchmark functions that are currently used in the DMOO literature.
- The identification of limitations of current DMOO benchmark functions.
- New DMOOPs that address the identified limitations of current DMOOP benchmark functions.
- An ideal DMOO benchmark function suite that contains:
  - characteristics that an ideal DMOOP suite should exhibit.
  - suggested DMOOPs for each identified characteristic.
- A comprehensive overview of performance measures that are currently used to measure the performance of DMOO algorithms.
- The identification of issues with current DMOO performance measures.

Through empirical analysis the following observations were made that contribute to knowledge in the fields of DMOO and PSO:

- Pareto-dominance based guide update approaches lead to improved performance over approaches that do not use Pareto-dominance information.
- Managing boundary constraint violations with the clamping (placing any particle that violates a specific boundary of the search space on or close to the violated boundary) approach produced the best performance.
- Re-initialising particles after a change in the environment occurs lead to improved performance over re-evaluation of the particles.
- For DMOOPs where the POF changes over time (Type II and Type III), removing all solutions from the archive after a change in the environment produced better results than re-evaluating the solutions and removing the solutions that became

dominated after the change. However, for Type I DMOOPs where the POF remains static, removing all solutions from the archive after a change lead to poor performance.

- PSO successfully solves DMOOPs of various types.

1.4 Research Methodology

Firstly, the DMOO literature was reviewed to determine the limitations with regards to:

- the development of DMOO algorithms, especially with reference to PSO algorithms.
- benchmark functions for DMOO. The review revealed that there are no standard benchmark functions for DMOO. Therefore, this thesis proposes an ideal set of DMOOPs that consists of current DMOOPs, as well as newly proposed DMOOPs.
- performance measures to determine whether these performance measures are adequate. Issues with regards to current DMOO performance measures were identified through empirical studies on DVEPSO. These issues are discussed and illustrated in this thesis.

Secondly, problems were identified with vector evaluated particle swarm optimisation (VEPSO) when solving DMOOPs. Therefore, various methods were proposed to adapt VEPSO for DMOO. An empirical analysis of DVEPSO was done to investigate the effect of these proposed changes on the performance of DVEPSO. Using formal hypothesis testing and statistical analysis, a final best performing configuration of DVEPSO was identified.

Thirdly, the best configuration of DVEPSO was compared against current state-of-the-art DMOO algorithms, namely:

- DNSGA-II-A and DNSGA-II-B, two NSGA-II algorithms adapted for DMOO and proposed by Deb *et al.* [46]. The source code of the static NSGA-II was obtained from [109] and adapted for DMOO according to [46].
- dCOEA, a dynamic competitive-cooperative coevolutionary algorithm proposed by Goh and Tan [67]. The source code of dCOEA was obtained from the first author of [67].

- MOPSO algorithm, a PSO algorithm adapted for DMOO by Lechuga [102].

For each of these state-of-the-art DMOO algorithms an empirical analysis was performed to determine the best configuration of the algorithm for the comparison study. Formal hypothesis testing and statistical analysis were performed to compare the performance of these DMOO algorithms and DVEPSO with one another.

1.5 Thesis Outline

The remainder of this thesis is organised in three main parts, namely optimisation background, computational intelligence algorithms and DVEPSO. The outline of each of these sections are provided next.

The outline of the part on optimisation background is as follows:

- **Chapter 2** presents the formal definitions of basic concepts required as background for various types of optimisation problems, namely single-objective optimisation problems (SOOPs), MOOPs and DMOOPs.
- **Chapter 3** provides an overview of DMOO benchmark functions that are currently used. Limitations of the DMOOPs are identified and new DMOOPs are proposed to address the limitations. An ideal set of benchmark functions are presented, highlighting the characteristics of an ideal benchmark function suite. Furthermore, example DMOOPs are suggested for each of the identified characteristics.
- **Chapter 4** provides an overview of DMOO performance measures. In addition, issues with currently used performance measures are illustrated and discussed.

The part on computational intelligence algorithms is organised as follows:

- **Chapter 5** provides basic background on single-objective optimisation (SOO) computational intelligence algorithms that are referred to later in this thesis. Basic concepts of PSO and genetic algorithms (GAs) are discussed.
- **Chapter 6** provides information about population-based algorithms that were used to solve MOOPs and that are referred to in later chapters of the thesis. A description of non-dominated sorting genetic algorithm II (NSGA-II), cooperative-coevolution evolutionary algorithm (CCEA) and multi-objective Particle Swarm Optimisation (MOPSO) are provided.

- **Chapter 7** covers vector-evaluated MOO algorithms. The vector evaluated genetic algorithm (VEGA), as well as the VEPSO algorithm that is inspired by VEGA, are discussed. A differential evolution (DE) version of VEGA, namely vector evaluated differential evolution (VEDE), is also discussed. Furthermore, information is provided about a hybrid algorithm that uses both VEPSO and VEDE to solve MOOPs.
- **Chapter 8** discusses population-based DMOO algorithms. Methods used by DMOO algorithms to detect and respond to changes are covered.

The part on DVEPSO discusses the DMOO algorithm that is proposed in the thesis. The outline of the DVEPSO part is:

- **Chapter 9** introduces the DVEPSO algorithm. The adaptation of VEPSO for DMOO, as well as the various parameters of DVEPSO, are discussed. New guide update approaches that use Pareto-dominance infmormation are proposed. An empirical study is performed to determine the influence of various guide update approaches on the performance of DVEPSO.
- **Chapter 10** presents an empirical study investigating the effect that various knowledge sharing approaches, approaches to manage boundary constraint violations and various responses to a change in the environment have on the performance of DVEPSO.
- **Chapter 11** investigates the performance of DVEPSO in comparison with other DMOO algorithms. An empirical study is discussed that compares the performance of DVEPSO with four other state-of-the-art DMOO algorithms.

Finally, **Chapter 12** concludes the work that has been presented in this thesis.

Additional information is provided in the Appendices as follows:

- **Appendix A** lists and defines the mathematical symbols used in this thesis, categorised according to the relevant chapter in which they appear.
- **Appendix B** provides a list of the important acronyms used or newly defined in the thesis, as well as their associated definitions.
- **Appendix C** discusses the calculation of a DMOOPs true POF.
- **Appendix D** presents the performance measure values and the p-values obtained

from the experiments discussed in Chapters 9, 10 and 11 respectively.

- **Appendix E** lists the publications derived from this research.

Part I

Optimisation Background

Chapter 2

Formal Definitions

> *"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil." – Donald E. Knuth*

In the modern world of today optimisation occurs in many aspects and areas of everyday life. For example, a manufacturer wants to increase his profit and therefore the cost of the manufacturing process has to be as low as possible. If this is approached as an unconstrained SOOP, it can be defined as follows:

Example 2.1: A manufacturer wants to minimise the cost of the manufacturing process.

However, many optimisation problems have more than one goal and some problems occur in a changing environment. Example 2.1 can be defined with more than one goal to increase a manufacturer's profit, namely minimising the cost of the manufacturing process and maximising the number of manufactured goods produced per day.

This chapter provides a theoretical overview of optimisation, presenting theory and definitions that are needed throughout the thesis. It does not give a complete overview of all aspects of MOO, dynamic single-objective optimisation (DSOO) and DMOO. However, this chapter highlights the most important information that is required to understand concepts discussed in later chapters. Section 2.1 discusses the main concepts of optimisation theory, highlighting the different types of optima and characteristics of optimisation problems. The theory of MOO is summarised in Section 2.2, where a MOO

problem is defined and the goal of solving MOO problems is clarified. Section 2.3 discusses DSOO and highlights the various types of environments for DSOOPs. DMOO, and the different types of DMOO problems, are presented in Section 2.4.

2.1 Single Objective Optimisation

This section discusses the main concepts of SOO. Section 2.1.1 discusses SOO theory that is required to understand the main concepts of MOO theory and Section 2.1.2 discusses the type of solutions that can be obtained for SOOPs.

2.1.1 Optimisation Concepts

Each optimisation problem contains one or more objective functions and a set of decision variables and most optimisation problems contain a set of constraints. Optimisation problems can be classified according to a number of characteristics, including the number of decision variables, the type of decision variables, the degree of linearity of the objective functions, the type of constraints, the number of optimisation criteria or objectives and the number of optima [36, 55]. These concepts are discussed in more detail below.

The **objective function** represents the quantity to be optimised, i.e. the quantity to be minimised or maximised. The objective function is also referred to as the *cost function* or *optimisation criterion*. If the problem that has to be optimised is expressed using only one objective function, it is referred to as a SOOP. However, if a problem has more than one objective that have to be optimised simultaneously, it is called a MOOP.

Each objective function has a vector of **decision variables** that influence the value of the objective function. Therefore, a search algorithm iteratively modifies the value of these variables to find the optimum for the objective function. If $\mathbf{x}$ represents the set of variables, the value of the objective function for the specific values of the variables can be quantified by $f(\mathbf{x})$. Therefore, $f(\mathbf{x})$ also quantifies the quality of the candidate solution, $\mathbf{x}$.

A problem with only one decision variable to optimise (only one variable influences the objective function) is referred to as a *univariate* problem. A *multivariate* problem is a problem where more than one variable influence the objective function. When the

type of decision variables is taken into account and a problem's decision variables have only continuous values, i.e. $x_k \in \mathbb{R}$, $\forall k = 1, \ldots, n_x$, the problem is referred to as a *continuous-valued* problem. The domain of a *discrete-valued* optimisation problem has a limited number of discrete values. Combinatorial problems are problems were solutions are permutations of integer-valued variables. When the decision variables can only have 0 or 1 as value, the problem is called a *binary-valued* problem.

When an objective function is linear in its variables, the problem is a *linear* problem. A *quadratic* problem has a quadratic objective function. However, when any other non-linear objective functions are used, the problem is referred to as a *non-linear* problem.

If an optimisation problem has constraints, the set of **constraints** restricts the values that can be assigned to the set of decision variables. Equality constraints restrict a variable to a specific value, for example $g(x_2) = 3$. Inequality constraints can take one of two forms, namely:

- Boundary constraints that restrict the domain of values that can be assigned to each variable and thereby define the *search space*. For example, $-1 \leq x_1 \leq 1$ restricts the value that variable x_1 can have to values between -1 and 1.
- Constraints of the form $c(\mathbf{x}) \leq 0$ or $c(\mathbf{x}) \geq 0$.

Values of $\mathbf{x}$ that satisfy the constraints form the *feasible search space* that is a subset of the search space. Problems that only use boundary constraints are generally referred to as *unconstrained* problems. However, when problems also have equality or inequality constraints, these problems are referred to as *constrained* optimisation problems.

When solving an optimisation problem with either equality or inequality constraints, the optimisation method's goal is to assign values from the specified domain to the decision variables in order to optimise the objective function and to satisfy the constraints. Therefore, the optimisation algorithm searches for a solution in the feasible search space, $\mathbf{x} \in F \subseteq S \subseteq \mathbb{R}^{n_x}$, that will obtain the smallest possible objective function value, $f(\mathbf{x})$, for a minimisation problem (or largest possible value for a maximisation problem). Throughout this thesis, unless stated differently, minimisation is assumed.

Mathematically, a SOOP is defined as:

$$\begin{aligned} minimise: \quad & f(\mathbf{x}) \\ subject\ to: \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n_g \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}\,, \mathbf{x}_{max}]^{n_x} \end{aligned} \tag{2.1}$$

where n_x is the number of decision variables; $\mathbf{x} = (x_1, x_2, \ldots, x_{n_x}) \in S \subseteq \mathbb{R}^{n_x}$; n_g is the number of inequality constraints, $\mathbf{g}$; n_h is the number of equality constraints, $\mathbf{h}$; and $\mathbf{x} \in [\mathbf{x}_{min}\,, \mathbf{x}_{max}]^{n_x}$ refers to the boundary constraints (domain of $\mathbf{x}$), with $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ referring to the lower- and upper bounds of the feasible values for decision variables $\mathbf{x}$. The research in this thesis focuses on unconstrained optimisation problems.

The next section discusses the various types of solutions that can be found when solving a SOOP.

2.1.2 Types of Solutions

This section discusses the type of solutions with various degrees of quality that can be obtained when solving SOOPs.

Solutions found by an optimisation algorithm can be classified according to their quality, where the main types of solutions for a minimisation problem are the global minimum and local minimum. The various degrees of solution quality, in terms of the global minima and local minima, are defined below.

Definition 2.1. Global minima: The solution $\mathbf{x}_i^* \in F$, with $F \subseteq S$, is a global minimum of the objective function f, if

$$f(\mathbf{x}_i^*) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in F,\ \mathbf{x}_i^* \neq \mathbf{x},\ \forall i = 1, \ldots, q \tag{2.2}$$

where q is the number of global minima of the SOOP.

Therefore, the best candidate solutions that lead to the smallest value of the objective function is called the global minima. The various types of minima are illustrated in Figure 2.1, with the point $\mathbf{x}_2$ as the global minimum of the function. It is important to

note that an optimisation problem can have more than one global minimum. A problem with only one solution (or optimum) is a *uni-modal* problem, but if more than one optimum exists, the problem is referred to as a *multi-modal* problem.

Local minima can be either strong or weak, defined as follows:

Definition 2.2. Strong local minima: The solution $\mathbf{x}^*_{N_i} \in N \subseteq F$ is a strong local minimum of the objective function f, if

$$\mathbf{x}^*_{N_i} < f(\mathbf{x}), \quad \forall \mathbf{x} \in N,\ \mathbf{x}^*_{N_i} \neq \mathbf{x},\ \forall i = 1, \ldots, q \tag{2.3}$$

where $N \subseteq F$ is a subset of points in the feasible space that is in the neighbourhood of $\mathbf{x}^*_N$ and q is the number of strong local minima of the SOOP. The point $\mathbf{x}_1$ in Figure 2.1 is a strong local minimum.

Definition 2.3. Weak local minimum: The solution $\mathbf{x}^*_{N_i} \in N \subseteq F$ is a weak local minimum of the objective function f, if

$$f(\mathbf{x}^*_{N_i}) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in N,\ \mathbf{x}^*_{N_i} \neq \mathbf{x},\ \forall i = 1, \ldots, q \tag{2.4}$$

where q is the number of weak local minima of the SOOP. Point $\mathbf{x}_3$ in Figure 2.1 is a weak local minima.

2.2 Multi-objective Optimisation

Many optimisation problems have more than one objective. The manufacturing example given earlier in Example 2.1 can be extended to a MOOP as follows:

Example 2.2: A manufacturer wants to maximise its profit. However, many factors have an influence on profit, for example the time required to manufacture a specific number of products, the time that a specific machine is idle and the cost of the manufacturing process. Therefore, the goals or objectives of the manufacturer are to minimise the time required to manufacture a specific number of products, to minimise the time that a specific machine is idle, and to minimise the cost of the manufacturing process.

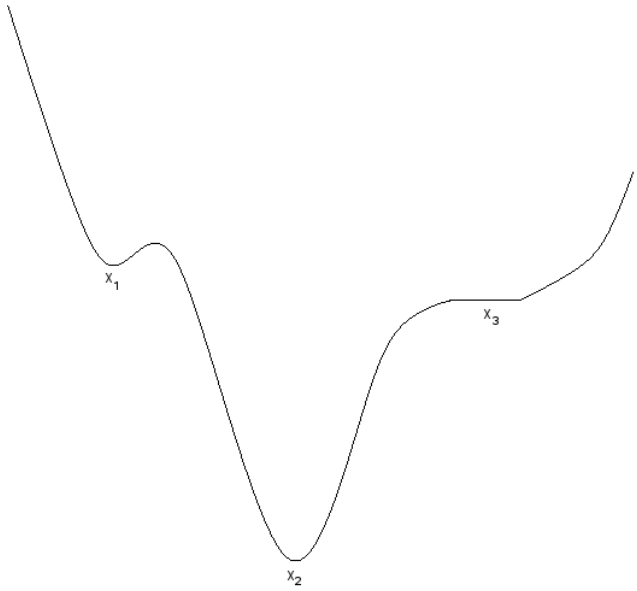


Figure 2.1: Optima of a minimisation function

However, using a specific machine can be more expensive to use than another, and the more expensive machine may require less time to manufacture the same number of products than a machine that is cheaper to operate. Therefore, in order to manufacture the maximum number of products in a certain time, using the more expensive machine will minimise the time required, but will increase the cost.

This example highlights an important problem with many MOOPs, namely that the objectives are in conflict with one another – minimising the time that the more expensive machine is idle increases the operational cost and vice versa. In this thesis, when referring to MOO, MOOPs with conflicting objectives are implied.

This section discusses the theory and definitions with regards to MOO [36, 55]. A MOOP is defined in Section 2.2.1 and the concept of optima is extended for MOO in Section 2.2.2. Section 2.2.3 discusses the goal when solving a MOOP and how this goal differs from situations when solving a SOOP.

2.2.1 Multi-objective Optimisation Problems

This section extends the mathematical definition of a SOOP (refer to Equation 2.1) to mathematically define a MOOP.

Let a single objective function be defined as $f_k : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$. Then $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{n_k}(\mathbf{x})) \in O_{space} \subseteq \mathbb{R}^{n_m}$ represents an *objective vector* containing n_k objective function evaluations, and O_{space} is the *objective space*.

Using the notation defined above, a MOOP can be mathematically defined as follows:

$$\begin{aligned} minimise: \quad & \mathbf{f}(\mathbf{x}) \\ subject\ to: \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n_g \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \end{aligned} \tag{2.5}$$

2.2.2 Pareto-optimal Set and Pareto Optimal Front

For SOOPs, where only one objective is optimised, local and global optima are defined as presented in Section 2.1.2. However, when dealing with a MOOP, the various objectives are normally in conflict with one another, i.e. improvement in one objective leads to a worse solution for at least one other objective. For the manufacturing example (Example 2.2 in Section 2.2) the various objectives, namely to minimise the time required to manufacture a specific number of products, to minimise the time that a specific machine is idle, and ro minimise cost, are in conflict with one another. MOOPs do not have specific optima, but trade-off solutions. Therefore, for MOOPs, the definition of optimality has to be re-defined. This section discusses the new definition of optimality for MOO.

When solving a MOOP the goal is to find a set of trade-off solutions where for each of these solutions no objective can be improved without causing a worse solution for at least one of the other objectives. These solutions are referred to as *non-dominated solutions* and the set of such solutions is called the *non-dominated set* or *Pareto-optimal set (POS)*. The corresponding objective vectors in the objective space that lead to the non-dominated solutions are referred to as the *POF* or *Pareto-front*. These concepts and definitions are now discussed in more detail.

For MOOPs, when one decision vector dominates another, the dominating decision vector is considered as a better decision vector. Decision vector domination is defined as follows:

Definition 2.4. Decision Vector Domination: A decision vector $\mathbf{x}_1$ dominates another decision vector $\mathbf{x}_2$, denoted by $\mathbf{x}_1 \prec \mathbf{x}_2$, if and only if

- $\mathbf{x}_1$ is at least as good as $\mathbf{x}_2$ for all the objectives, i.e. $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2)$, $\forall k = 1, \ldots, n_k$; and
- $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ for at least one objective, i.e. $\exists k = 1, \ldots, n_k \colon f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$.

where n_k is the number of objective functions.

A slightly less strict comparison can be made between two decision vectors using the concept of weak domination, defined as:

Definition 2.5. Weak Decision Vector Domination: A decision vector $\mathbf{x}_1$ weakly dominates another decision vector $\mathbf{x}_2$, denoted by $\mathbf{x}_1 \preceq \mathbf{x}_2$, if and only if

$\mathbf{x}_1$ is at least as good as $\mathbf{x}_2$ for all the objectives, i.e. $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2)$, $\forall k = 1, \ldots, n_k$

The decision vectors that lead to the best trade-off solutions, are called Pareto-optimal, defined as follows:

Definition 2.6. Pareto-optimal: A decision vector $\mathbf{x}^*$ is Pareto-optimal if there does not exist a decision vector $\mathbf{x} \neq \mathbf{x}^* \in F$ that dominates $\mathbf{x}^*$, i.e. $\nexists k \colon f_k(\mathbf{x}) < f_k(\mathbf{x}^*)$. If $\mathbf{x}^*$ is Pareto-optimal, the objective vector, $\mathbf{f}(\mathbf{x}^*)$, is also Pareto-optimal.

Together, all the Pareto-optimal decision vectors form the Pareto-optimal set (POS), defined as:

Definition 2.7. Pareto-optimal Set: The POS, P^*, is formed by the set of all Pareto-optimal decision vectors, i.e.

$$P^* = \{\mathbf{x}^* \in F \,|\, \nexists \mathbf{x} \in F \colon \mathbf{x} \prec \mathbf{x}^*\} \tag{2.6}$$

The POS contains the best trade-off solutions for the MOOP. The corresponding objective vectors form the Pareto-optimal front (POF), which is defined as follows:

Definition 2.8. Pareto-optimal Front: For the objective vector $\mathbf{f}(\mathbf{x})$ and the POS P^*, the POF, $PF^* \subseteq O_{space}$ is defined as

$$PF^* = \{\mathbf{f} = (f_1(\mathbf{x}^*), f_2(\mathbf{x}^*), \ldots, f_{n_k}(\mathbf{x}^*))\},\ \forall \mathbf{x}^* \in P^* \tag{2.7}$$

Therefore, the POF contains the set of objective vectors that corresponds to the POS, i.e. the set of decision vectors that are non-dominated. The POF can have various shapes, e.g. a convex POF or a concave POF, as can be seen in Section 3.1.

Some MOO algorithms make use of ϵ-domination and an ϵ-approximate POF, proposed by Laumanns *et al.* [101], which are extensions of Definitions 2.4 and 2.8 above. With ϵ-domination, a decision vector $\mathbf{x}$ dominates not only all decision vectors as defined in Definition 2.4, but also all decision vectors that are within a distance ϵ of $\mathbf{x}$. The ϵ value can be selected by the decision maker to control the size of the set of solutions [80]. Furthermore, ϵ-domination provides a way for algorithms to find solutions that converge to the POF and that has a good diversity [101]. ϵ-domination for decision vectors and objective vectors, and an ϵ-approximate POF are defined below in Definitions 2.9, **??** and 2.10 respectively.

Definition 2.9. Decision Vector ϵ-Domination: A decision vector $\mathbf{x}_1$ ϵ-dominates another decision vector $\mathbf{x}_2$, denoted by $\mathbf{x}_1 \prec_\epsilon \mathbf{x}_2$, if and only if

- $f_k(\mathbf{x}_1)/(1+\epsilon) \leq f_k(\mathbf{x}_2),\ \forall k = 1, \ldots, n_k,\ \epsilon > 0$; and
- $\exists k = 1, \ldots, n_k \colon f_k(\mathbf{x}_1)/(1+\epsilon) < f_k(\mathbf{x}_2),\ \epsilon > 0$.

Definition 2.10. ϵ-approximate Pareto-optimal Front: For the objective vector $\mathbf{f}(\mathbf{x})$ and an $\epsilon > 0$, the ϵ-approximate POF, $PF^*_\epsilon \subseteq O_{space}$, contains all objective vectors which are not ϵ-dominated by any other objective vector and is therefore defined as

$$PF^*_\epsilon = \{\mathbf{f} = (f_1(\mathbf{x}^*), f_2(\mathbf{x}^*), \ldots, f_{n_k}(\mathbf{x}^*)),\ \forall \mathbf{x}^* \in P^* \,|\, \nexists \mathbf{x} \in F \colon f(\mathbf{x}) \prec_\epsilon f_k(\mathbf{x}^*),\ \forall k = 1, \ldots, n_k\} \tag{2.8}$$

2.2.3 Solving a Multi-objective Optimisation Problem

When solving a MOOP, the goal is to approximate the true POF. If the problem requires a single solution, the best trade-off solution is selected for the specific problem from the set of solutions represented by the POF. Therefore, the goal is to find an approximation of the true POF such that:

- The distance between the found POF and the true POF is minimised.
- The set of non-dominated solutions is as diverse as possible and as evenly spread out along the found POF as possible.
- The set of non-dominated solutions contains as many solutions as possible.
- The solutions that have been found and that forms the found POF are stored for later reference.

Similar to a SOOP having global and local optima, a MOOP can have a global POF or local POFs. Definitions 2.1 to 2.3 for SOO is extended for MOO as follows:

Definition 2.11. Global POF: PF_g^* is the global POF of a DMOOP, $\mathbf{f}$, if

$$\mathbf{f}(\mathbf{x}^*) \prec \mathbf{f}(\mathbf{x}), \quad \forall \mathbf{x} \in F | \mathbf{x} \notin P^*, \forall \mathbf{x}^* \in P^*, \mathbf{x}^* \neq \mathbf{x} \tag{2.9}$$

where P^* is the POS of $\mathbf{f}$.

Therefore, the best candidate solutions that lead to the best trade-off solutions, form the POS and the corresponding values in the objective space result in the global POF or the true POF. A MOOP can have many local POFs, with a local POF defined as follows:

Definition 2.12. Local POF: $PF_{l_i}^*$ is a local POF of a DMOOP, $\mathbf{f}$, if

$$\mathbf{f}(\mathbf{x}_{N_i}^*) \prec \mathbf{f}(\mathbf{x}), \quad \forall \mathbf{x} \in N | \mathbf{x} \notin P^*, \mathbf{x}_{N_i}^* \neq \mathbf{x}, \mathbf{x}_{N_i}^* \in N, \forall i = 1, \ldots, q \tag{2.10}$$

where $N \subseteq F$ is a subset of points in the feasible space that is in the neighbourhood of $\mathbf{x}_{N_i}^*$ and q is the number of local POFs.

When a MOOP has local POFs, an algorithm can become stuck in one of the local POFs and this will prevent the algorithm from converging to the global POF.

2.3 Dynamic Single-objective Optimisation

In many real-world situations the objective function that has to be optimised is not static. A change in the objective function and/or the constraints can lead to a change in the environment. The change in the objective function and/or constraints causes a change in the search landscape, S, and/or the feasible space F, and causes changes to the optima of the problem, i.e. optima can change in position or value, or optima can disappear while new optima can appear. The manufacturing example, Example 2.1, can be extended to illustrate a DSOOP as follows:

Example 2.3: A manufacturer wants to minimise the cost of the manufacturing process. If the cost is calculated by taking the cost of using the machines into account, then if one machine breaks down, the environment changes. There will be idle time while the machine is being replaced and the new machine may not be exactly the same as the previous one – the new machine may be more expensive to use and/or may need longer time to complete the manufacturing process. Therefore, the previous solution cannot be used anymore, and a new solution for the changed situation has to be found.

This section discusses the theory and definitions [55] with regards to DSOO. A DSOOP is mathematically defined in Section 2.3.1 and Section 2.3.2 discusses the various classifications of dynamic environments.

2.3.1 Dynamic Single-objective Optimisation Problem

A DSOOP can formally be defined as follows:

$$\begin{aligned} Minimise: \quad & f(\mathbf{x}, t), \quad && \mathbf{x} = (x_1, \ldots, x_{n_x}) \\ Subject\ to: \quad & g_i(\mathbf{x}, t) \leq 0, \quad && i = 1, \ldots, n_g \\ & h_j(\mathbf{x}, t) = 0, \quad && j = 1, \ldots, n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}\,;\ \mathbf{x}_{max}]^{n_x} \end{aligned} \tag{2.11}$$

In order to solve the DSOOP, the goal is to find

$$\mathbf{x}^*(t) = min_{\mathbf{x} \in F(t)}\ f(\mathbf{x}, t) \tag{2.12}$$

where $\mathbf{x}^*(t)$ is the optimum at time step t and $F(t)$ is the feasible space at time t.

Since the optima change with time, the goal of an optimisation algorithm for dynamic environments is to locate an optimum and track its trajectory as closely as possible, and to find new optima that may appear.

2.3.2 Dynamic Environment Types

Dynamic environments or DSOOPs can change in various ways over time. When a change occurs in the environment, *temporal severity* refers to the frequency of change that the environment experiences and *spatial severity* refers to the extend of change in the position of the optima.

Based on real-world problems, De Jong [91] identified four types of changes that can occur in a dynamic environment:

- **Drifting landscapes**, where the optima moves gradually over time, for example aging equipment in a large production plant.
- **Significant changes in the optima location**, where peaks of high fitness shrink and new regions of high fitness emerge that was previously uninteresting regions, for example competitive market places where opportunities for high profit fluctuate as the levels of competition change over time.
- **Cyclic patterns** in the landscape, where a relatively small number of states re-occur over time, for example seasonal climate changes.
- **Abrubt and discontinuous changes** in the landscape, for example a power station failure on a distribution grid.

Eberhart and Shi [53] defined the following three generic dynamic environment types for SOO:

- **Type I environments** where the location of the optimum in the problem space, $\mathbf{x}^*(t)$, changes, but $f(\mathbf{x}^*(t))$ remains unchanged. The spatial severity, ζ, measures the change in $\mathbf{x}^*(t)$.
- **Type II environments** where $\mathbf{x}^*(t)$ remains unchanged, but the objective function value at $\mathbf{x}^*(t)$, $f(\mathbf{x}^*(t))$, changes.
- **Type III environments** where both $\mathbf{x}^*(t)$ and $f(\mathbf{x}^*(t))$ changes. The change in $\mathbf{x}^*(t)$ is indicated by ζ.

These three types are summarised in Table 2.1.

Table 2.1: Dynamic environment types as defined by Eberhart and Shi [53]

	Optimum Location	
Optimum Value	No Change	Change
No Change	Static	Type I
Change	Type II	Type III

Branke [12] categorised dynamic environments according to the following characteristics:

- **Frequency of change** or temporal severity that determines how often the environment changes.
- **Severity of change** or spatial severity that are normally measured as the distance between the current and the previous optimum.
- **Predictability of change** that indicates whether the changes occur randomly or with a pattern that can be learned or predicted by an algorithm.
- **Cycle length or cycle accuracy** that indicates how long it takes before the environment returns to a previous state and how accurate or similar the returned state is with regards to the previous state.

More recently, Duhain [50] classified dynamic environments as follows:

- **Static environments**, where the environment does not change over time or the changes to the environment have such a small influence on the problem that they do not affect the performance of the algorithm for the duration of the simulation.
- **Progressively changing environments**, where the temporal severity is high, but the spatial severity (change in $\mathbf{x}^*(t)$) is low. Therefore, the environment changes in a progressive manner. Algorithms that solve problems with a progressively changing environment can use knowledge that was obtained earlier (the previous optima) to find the new optima that will be in close proximity of the previous optima.
- **Abruptly changing environments**, where the temporal severity is low, but the

spatial severity is high. Therefore, previous knowledge will not be as useful as in the case of a progressively changing environment.

- **Chaotic environments** where both the temporal and spatial severity are high.

These four types are summarised in Table 2.2. Duhain's classification is similar to De Jong, but more generic and using the concepts of temporal severity and spatial severity.

If the temporal severity is high, the environment changes frequently and therefore an algorithm would have to converge to the optima at a specific time step quickly and adapt quickly after a change to find the new optima. A high spatial severity occurs when $\mathbf{x}^*(t+1)$ differs severely from $\mathbf{x}^*(t)$ and therefore an algorithm has to find the new optima that is far from the previous location in the search space. It is important to note that not all problems' environment will remain one type for the whole duration of the simulation, but can change over time from one type of environment to another.

Table 2.2: Dynamic environment types as defined by Duhain [50]

	Spatial Severity	
Temporal Severity	Low	High
Low	Static	Abrupt
High	Progressive	Chaotic

2.4 Dynamic Multi-objective Optimisation

In most situations the optimisation problem is not static, and has more than one objective. Example 2.2 (refer to Section 2.2) can be extended to illustrate a DMOOP as follows:

Example 2.4: A manufacturer wants to maximise its profit. Therefore, the goals or objectives of the manufacturer are to minimise the time required to manufacture a specific number of products, to minimise the time that a specific machine is idle, and to minimise the cost of the manufacturing process. When a machine breaks down, the environment changes. This change in the environment may also influence more than one

objective function. The breakdown of a machine can occur quite frequently and other changes can also occur. For example, the operational cost of a specific machine may change when it breaks down and is replaced by another machine that is not exactly the same as the replaced machine, the time required to complete the manufacturing process may take longer for a machine as it gets older, etc. Since this manufacturing problem is not static in nature, but dynamic, the previous solutions or POF will not be valid anymore and a new POF has to be found.

This section discusses DMOO in more detail. Section 2.4.1 provides a mathematical definition of a DMOOP and the various types of dynamic DMOOPs are discussed in Section 2.4.2.

2.4.1 Dynamic Multi-objective Optimisation Problem

This section provides a mathematical definition of a DMOOP.

A DMOOP can be defined as:

$$\begin{aligned} Minimise: \quad & \mathbf{f}(\mathbf{x}, t), \quad && \mathbf{x} = (x_1, \ldots, x_{n_x}) \\ Subject\ to: \quad & g_i(\mathbf{x}, t) \leq 0, \quad && i = 1, \ldots, n_g \\ & h_j(\mathbf{x}, t) = 0, \quad && j = 1, \ldots, n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}\,;\ \mathbf{x}_{max}]^{n_x} \end{aligned} \tag{2.13}$$

Unlike DSOOPs with only one objective function, DMOOPs have many objective functions. Therefore, in order to solve the DMOOP the goal is to track the POF over time, i.e.

$$PF^*(t) = \{\mathbf{f}(t) = (f_1(\mathbf{x}^*, t), f_2(\mathbf{x}^*, t), \ldots, f_{n_k}(\mathbf{x}^*, t)\}, \ \forall \mathbf{x}^* \in P^*(t) \tag{2.14}$$

The next section discusses the various types of DMOOPs, as well as the various ways in which the POF can be affected when a change occurs in the environment.

2.4.2 Dynamic Environment Types

This section discusses the categorisation of DMOOPs, as well as the possible influences of a change in the environment on the POF.

Similar to the classification of dynamic environment types for DSOOPs (refer to Section 2.3.2), Farina *et al.* [58] classified dynamic environments for DMOOPs into four categories, namely:

- **Type I environment** where the POS (optimal set of decision variables) changes, but the POF (corresponding objective function values) remains unchanged.
- **Type II environment** where both the POS and the POF change.
- **Type III environment** where the POS remains unchanged, but the POF changes.
- **Type IV environment** where both the POS and the POF remain unchanged, even though an objective function or a constraint may have changed.

These four types are summarised in Table 2.3.

Table 2.3: Dynamic Environment Types for DMOO problems

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

When a change occurs in the environment, the POF can change as follows over time:

1. Existing solutions in the POF becomes dominated and therefore are not part of the POF any more.
2. The shape of the POF remains the same, but its location in the objective space change over time. In these cases the POF shifts over time. This kind of change of the POF occurs with type I DMOOPs and are the easiest kind of DMOOPs to solve.
3. The shape of the POF changes over time. For example:
   - The POF changes from convex to concave or vice versa.
   - The POF changes from a continuous front to a disconnected front, i.e. various disconnected continuous-valued areas.

   This kind of change of the POF occurs with either type II or type III DMOOPs. When the shape of the POF changes over time, an algorithm has to track the

changing POF and obtain a diverse set of solutions for the new shape of the POF. Therefore, if the shape of the POF changes over time, an algorithm may struggle to find a diverse set of solutions after a change has occurred.

4. The density of the solutions in the POF changes over time. For example:
   - The solutions in the POF becomes more/less dense.
   - The number of solutions in the POF becomes more/less.

   This kind of change in the POF can occur with all types of DMOOPs. When the number of solutions or the densitiy of the solutions in the POF change overtime, algorithms may struggle to find a diverse set of solutions.

2.5 Summary

This chapter discussed aspects of optimisation relevant to this thesis. Section 2.1.1 discussed optimisation problems and their characteristics with regards to the problem's objective functions, decision variables and constraints. Different types of solutions exist for an optimisation problem of which the main types are global and local minima, as defined in Section 2.1.2. Section 2.2.1 defined a MOOP and in order to re-define the optima for a MOOP, the concepts of a POS and POF were discussed in Section 2.2.2. Since most MOOPs do not have a single solution because of conflicting objectives, the goal when solving MOOPs were summarised in Section 2.2.3. Furthermore, the concepts of local and global optima for SOO have been extended to define local and global POFs for MOO in Section 2.2.3.

In real life, optimisation problems are not static in nature and change over time. Therefore, both DSOO and DMOO were introduced in this chapter. DSOO was discussed in Section 2.3 and a DSOOP was defined in Section 2.3.1. The environment of a DSOOP can change in various ways, as discussed in Section 2.3.2. However, many dynamic optimisation problems do not have only one objective and therefore DMOO was introduced in Section 2.4 and a DMOOP was defined in Section 2.4.1. Similar to DSOOPs, the environment of a DMOOP and the POF can change in various ways over time, as discussed in Section 2.4.2.

There exist many different approaches that are used to solve optimisation problems:

Population-based algorithms within the field of computational intelligence (CI), such as evolutionary algorithms (EAs), PSO algorithms, and ant algorithms, are widely used to solve optimisation problems. Various population-based approaches that are used to solve MOO and DMOO problems are discussed in Chapters 6, 7 and 8.

The next chapter discusses bechmark functions that are used to evaluate whether an algorithm can solve DMOOPs.

Chapter 3

Analysis of Dynamic Multi-objective Optimisation Benchmark Functions

> *"Without a standard there is no logical basis for making a decision or taking action."* – Joseph M. Juran

Dynamic multi-objective optimisation problems are created by adjusting MOOPs in one or more of the following ways: changing the objective functions over time or changing the constraints over time. This thesis focusses on unconstrained DMOOPs with static boundary constraints and objective functions that change over time.

In order to determine whether an algorithm can solve DMOOPs efficiently, DMOOPs should be used that test the ability of the algorithm to overcome certain difficulties. These DMOOPs are called benchmark functions. One of the main problems in the field of DMOO is a lack of standard benchmark functions. This chapter seeks to address this problem by evaluating the current state-of-the-art benchmark functions presented in the DMOO literature to establish whether they efficiently evaluate the abilities of DMOO algorithms.

MOO benchmark functions adapted to develop DMOOPs and characteristics that an ideal set of MOO benchmark functions should have are discussed in Section 3.1. Current benchmark functions used in the DMOO literature are discussed in Section 3.2. Furthermore, approaches to develop DMOOPs with either an isolated or deceptive POF are proposed. New DMOOPs with complicated POSs, i.e. POSs that are defined by

non-linear functions and where each decision variable has a different POS are introduced. Characteristics that an ideal DMOO benchmark function suite should have, are also presented and benchmark functions are suggested for each identified characteristic. Finally, a summary of this chapter is provided in Section 3.3.

3.1 Multi-objective Optimisation Benchmark Functions

Benchmark functions test how well an algorithm can overcome various types of difficulties when trying to find the true POF. When an algorithm solves a MOOP its goal is to find solutions that are as close as possible to the true POF and that have an uniform spread. Therefore, benchmark problems should test whether an algorithm can achieve this goal when faced with either multi-modality, deception (such as local POFs and isolated optima that may prevent the algorithm from converging towards the true POF; or a POF that is non-convex, discontinuous or non-uniform that may prevent the algorithm from finding an uniform spread of solutions [36, 49].

Section 3.1.1 discusses characteristics of ideal benchmark functions suites. Furthermore, two MOO benchmark function suites, namely the ZDT [38] and DTLZ functions [49], that were adapted to develop DMOOPs are discussed in Sections 3.1.2 and 3.1.3 respectively.

3.1.1 Ideal Benchmark Function Characteristics

This section discusses characteristics that an ideal benchmark function suite should exhibit.

Deb *et al.* [49] constructed the ZDT [38, 169] and DTLZ [49] MOOP suites in such a way that the benchmark functions are:

- easy to construct,
- scalable in terms of the number of decision variables as well as the number of objective functions,

- producing a POF that is easy to understand with the POF's shape and location known, and
- hindering an algorithm to converge to the true POF and to produce a good distribution of solutions.

According to Deb *et al.* [38], an algorithm can be hindered in converging to the true POF when a benchmark function is multi-modal, deceptive, has an isolated optimum, or contains noise. Deceptive functions have at least two optima in the search space, but the search space favours the deceptive optimum, which is a local POF and not the true global POF. If a function is multi-modal, it has many POFs and a DMOO algorithm can become stuck in a local POF. If an open subset of decision variable values maps to a single objective function value, the objective function is referred to as an objective function with *flat regions*, i.e. regions where small perturbations of the decision variable values do not change the objective function value. The lack of gradient information for the flat regions may cause an algorithm to struggle to converge to the optima. For DMOOPs, if the majority of the fitness landscape is fairly flat and no useful information is provided with regards to the location of the POF, the POF is referred to as being *isolated*. Therefore, if the DMOOP has an isolated POF, a DMOO algorithm may struggle to converge towards it. Even if the POF is not completely isolated from the rest of the search space, i.e. the majority of the fitness landscape is not fairly flat, an algorithm may struggle to converge towards the POF if the density of solutions close to the POF is significantly less than in the rest of the search space.

The following properties of the true POF may cause difficulty for an algorithm to find a diverse set of solutions: convexity or non-convexity in the POF, a discontinuous POF, or a non-uniform spacing of solutions in the POS or POF [38, 40]. When a POF is convex, it may be difficult to solve the DMOOP by algorithms that assign a solution's fitness based on the number of solutions that the solution dominates (Pareto ranking) [38]. This fitness assignment favours intermediate or middling solutions that perform reasonably well with regards to all objective functions more than solutions that perform very good with regards to one objective and not so good with regards to the other objectives. Therefore, this fitness assignment may cause bias towards certain portions of the POF that contain intermediate solutions. If the POF is discontinous and has a

set of disconnected continuous sub-regions, an algorithm may struggle to find all regions of the POF. Even though an algorithm may find solutions within each region, when the solutions compete amongst each other (for storage in the archive or for a rank), solutions from certain sub-regions may be outranked and therefore may be removed from the non-dominated solution set. If the POS or POF is not uniformly spaced, an algorithm may struggle to find a diverse set of non-dominated solutions [40].

3.1.2 ZDT Functions

Deb introduced a tunable two-objective optimisation problem, defined as [38]:

$$\begin{aligned} \text{Minimise:} \quad & f(\mathbf{x}) = (f_1(\mathbf{x_I}), f_2(\mathbf{x})) \\ \text{Subject to:} \quad & f_1(\mathbf{x_I}) = f_1(x_1, x_2, \ldots, x_m) \\ & f_2(\mathbf{x_{II}}) = g(\mathbf{x_{II}}) \cdot \ h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}})) \\ & \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n) \end{aligned} \tag{3.1}$$

where $f_1, g > 0$. MOOPs with specific features can be created by changing the f_1, g and h functions:

- the selected h function influences the convexity or discontiuity of the POF.
- a difficult g function affects the level of difficulty that an algorithm experiences when converging to the true POF.
- the selected f_1 function affects the diversity or spread of solutions in the POF.

Based on this two-objective optimisation problem and the guidelines produced by Deb *et al* [38] as discussed in Section 3.1.1, Zitzler, Deb and Thiele introduced six benchmark functions referred to as the ZDT functions (first letter of the surnames of the three authors) [169]. Each of the functions are structured according to Equation (3.1) and addresses one of the six difficulties discussed in Section 3.1.1. The mathematical equations (Equations (3.2) to (3.7)) of these functions are presented below:

$$\text{ZDT1} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n), \;\; x_i \in [0, 1] \end{cases} \tag{3.2}$$

where $m = 30$. ZDT1 has a convex POF that is formed with $g(\mathbf{x_{II}}) = 1$. Therefore the POF of ZDT1 is $1 - \sqrt{f_1}$ and the POS is $x_i = 0,\ \forall i \in \mathbf{x_{II}}$.

$$\text{ZDT2} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(x_1) = x_1 \\ g(\mathbf{x_{II}}) = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1(x_1), g(\mathbf{x_{II}})) = 1 - \left(\frac{f_1}{g}\right)^2 \\ \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n), \;\; x_i \in [0, 1] \end{cases} \tag{3.3}$$

where $m = 30$. The POF is non-convex with POF $= 1 - f_1^2$. The POS of ZDT2 is $x_i = 0,\ \forall i \in \mathbf{x_{II}}$.

$$\text{ZDT3} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(x_1) = x_1 \\ g(\mathbf{x_{II}}) = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1(x_1), g(\mathbf{x_{II}})) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1) \\ where: \\ x_1 \in [0, 1], \;\; \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n) \in [-5, 5] \end{cases} \tag{3.4}$$

where $m = 10$. ZDT3 has a discrete POF that consists of several discontinuous convex parts. The sine function in h causes discontinuity in the POF, but not in the decision space. The POF is $1 - \sqrt{f_1} - f_1\sin(10\pi f_1)$. The POS of ZDT3 is $x_i = 0,\ \forall i \in \mathbf{x_{II}}$.

$$\text{ZDT4} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(x_1) = x_1 \\ g(\mathbf{x_{II}}) = 1 + 10(m-1) + \sum_{i=2}^{m}(x_i^2 - 10\cos(4\pi x_i)) \\ h(f_1(x_1), g(\mathbf{x_{II}})) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ x_1 \in [0,1], \quad \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n) \in [-5,5] \end{cases} \tag{3.5}$$

where $m = 10$. The POF of ZDT4 has 21^9 local POFs and therefore tests the algorithm's ability to deal with multi-modality. The global POF is formed with $g(\mathbf{x_{II}}) = 1$ and is $1 - \sqrt{f_1}$. The global POS is $x_i = 0$, $\forall i \in \mathbf{x_{II}}$. The best local POF can be found with $g(\mathbf{x_{II}}) = 1.25$.

$$\text{ZDT5} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(x_1) = u(x_1) \\ g(\mathbf{x_{II}}) = 1 + 9\sum_{i=2}^{m} v(u(x_i)) \\ h(f_1(x_1), g(\mathbf{x_{II}})) = \frac{1}{f_1} \\ where: \\ x_1 \in \{0,1\}^{30}, \quad \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n) \in \{0,1\}^5 \\ v(u(x_i)) = \begin{cases} 2 + u(x_i), & \text{if } u(x_i) < 5 \\ 1, & \text{if } u(x_i) = 5 \end{cases} \end{cases} \tag{3.6}$$

where $m = 11$. ZDT5 is a deceptive problem where x_i is represented by a binary string. The global POF is formed with $g(\mathbf{x_{II}}) = 10$. The best deceptive POF can be found where $g(\mathbf{x_{II}}) = 11$. The global and local POFs are convex.

$$\text{ZDT6} = \begin{cases} Minimise: f(\mathbf{x}) = (f_1(x_1), g(\mathbf{x_{II}}) \cdot h(f_1(x_1), g(\mathbf{x_{II}}))) \\ f_1(x_1) = 1 - exp(-4x_1)\sin^6(6\pi x_1) \\ g(\mathbf{x_{II}}) = 1 + 9\left(\frac{\sum_{i=2}^{m} x_i}{m-1}\right)^{0.25} \\ h(f_1(x_1), g(\mathbf{x_{II}})) = 1 - \left(\frac{f_1}{g}\right)^2 \\ where: \\ \mathbf{x_{II}} = (x_{m+1}, \ldots, x_n), \quad x_i \in [0,1] \end{cases} \tag{3.7}$$

where $m = 10$. ZDT6 causes two difficulties for algorithms because of the non-uniformity of the search space, namely: (i) the solutions are non-uniformly distributed along the

global POF, and (ii) the solutions are the least dense close to the POF and most dense away from the POF. ZDT6 has a non-convex POF $1 - f_1^2$. The POS of ZDT6 is $x_i = 0$, $\forall i \in \mathbf{x_{II}}$.

The ZDT functions are all two-objective optimisation problems. Therefore, Deb *et al.* [49] introduced test problems that can be scaled in terms of the number of objective functions.

3.1.3 DTLZ Functions

This section discusses two approaches, as well as a benchmark function generator, that were used to develop the Deb, Thiele, Laumanns and Zitzler (DTLZ) benchmark functions.

Spherical Coordinates Approach

Deb *et al.* [49] defined a test problem that has a POF in the first quadrant of a sphere with radius one and where all objective functions have non-negative values (add figure to refer to). Mathematically, using spherical coordinates (θ, γ and $r = 1$), the POF is defined as

$$\text{POF} = \begin{cases} f_1(\theta, \gamma) = \cos\theta \cos\left(\gamma + \frac{\pi}{4}\right) \\ f_2(\theta, \gamma) = \cos\theta \sin\left(\gamma + \frac{\pi}{4}\right) \\ f_3(\theta, \gamma) = \sin(\theta) \\ \text{where } 0 \leq \theta \leq \frac{\pi}{2}, \ \frac{-\pi}{4} \leq \gamma \leq \frac{\pi}{4} \end{cases} \tag{3.8}$$

Any two points of the surface defined by Equation (3.8) are non-dominated if all three objective functions are minimised. By defining the rest of the search space above this surface, the POF is defined as the unit sphere. This can be done by constructing the rest of the search space parallel to the surface defined in Equation (3.8) as follows:

$$\text{POF} = \begin{cases} Minimise: \\ \quad f_1(\theta,\gamma,r) = (1+g(r))\cos\theta\cos\left(\gamma+\frac{\pi}{4}\right) \\ \quad f_2(\theta,\gamma) = (1+g(r))\cos\theta\sin\left(\gamma+\frac{\pi}{4}\right) \\ \quad f_3(\theta,\gamma) = (1+g(r))\sin(\theta) \\ where: \\ \quad 0 \le \theta \le \frac{\pi}{2}, \quad \frac{-\pi}{4} \le \gamma \le \frac{\pi}{4} \\ \quad g(r) \ge 0 \end{cases} \tag{3.9}$$

where the POS is $0 \le \theta^* \le \frac{\pi}{2}$, $\frac{-\pi}{4} \le \gamma^* \le \frac{\pi}{4}$, $g(r)^* = 0$. Although this three-objective problem has three independent variables (θ, γ and r), the variables can be meta-variables and can be considered as a function of n decision variables, i.e. $\theta = \theta(x_1,\ldots,x_n)$, $\gamma = \gamma(x_1,\ldots,x_n)$, $r = r(x_1,\ldots,x_n)$. These functions must adhere to the lower and upper bounds of the three variables and can be used to introduce difficulties to the optimisation problem.

Constraint Surface Approach

Another approach used by Deb *et al.* to develop benchmark functions are based on a constraint surface [49]. Firstly, a search space is defined as follows:

$$\begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) \\ \quad \ldots \\ \quad f_M(\mathbf{x}) \\ where: \\ \quad f_i^L \le f_i(\mathbf{x}) \le f_i^U, \ \forall\, i = 1,2,\ldots,M \end{cases} \tag{3.10}$$

where f_i^L and f_i^U refers to the lower bound and upper bound of the objective function f_i respectively. The POS has only one solution, namely a solution that consists of the lower bound value of each objective, namely $(f_1^L, f_2^L, \ldots, f_M^L)^T$.

A set of constraints, that can be linear or non-linear, can be added to the problem in Equation (3.10), where each constraint eliminates a portion of the original rectangular

region. Therefore, the optimisation problem of Equation (3.10) becomes:

$$\begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) \\ \quad \dots \\ \quad f_M(\mathbf{x}) \\ where: \\ \quad f_i^L \leq f_i(\mathbf{x}) \leq f_i^U \ \ \forall\, i = 1, 2, \dots, M \\ \quad g_j(f_1, f_2, \dots, f_M) \geq 0, \ \ \forall\, j = 1, 2, \dots, J \end{cases} \tag{3.11}$$

In order to solve this MOOP, the goal of an algorithm becomes to find the non-dominated part of the feasible space's boundary. The density of solutions in the search space can be modified by using non-linear functions for f_i.

Benchmark Function Generator

Based on the constraint surface approach, Deb [40, p.361] suggested a generic MOOP generator where the number of objectives can be scaled. Mathematically, the generator is defined as

$$\begin{cases} Minimise: \\ \quad f_1(x_1) \\ \quad \vdots \\ \quad f_{M-1}(\mathbf{x_{M-1}}) \\ \quad f_M(\mathbf{x}) = g(\mathbf{x_M}) \cdot h(f_1, \dots, f_{M-1}, g) \\ where: \\ \quad x_i \in \mathbb{R}^{|x_i|}, \ \forall i = 1, 2, \dots, M \end{cases} \tag{3.12}$$

where POF $= f_M = g \text{ x } h(f_1, f_2, \dots, f_{M-1})$.

Using the concepts of Equations 3.9 and 3.12, Deb *et al.* [49] presented the DTLZ functions. The mathematical equations (Equations 3.13 to 3.19) of these functions are presented below:

$$\text{DTLZ1} = \begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) = \frac{1}{2}x_1x_2\dots x_{M-1}(1+g(\mathbf{x_M})) \\ \quad f_2(\mathbf{x}) = \frac{1}{2}x_1x_2\dots(1-x_{M-1})(1+g(\mathbf{x_M})) \\ \quad \vdots \\ \quad f_{M-1}(\mathbf{x}) = \frac{1}{2}x_1(1-x_2)(1+g(\mathbf{x_M})) \\ \quad f_M(\mathbf{x}) = \frac{1}{2}(1-x_1)(1+g(\mathbf{x_M})) \\ where: \\ \quad g(\mathbf{x_M}) = 100\left(|\mathbf{x_M}| + \sum_{x_i\in\mathbf{x_M}}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right) \\ \quad 0 \le x_i \le 1,\ \forall i = 1,2,\dots,n \\ \quad |\mathbf{x_M}| = k \ \ n = M+k-1 \end{cases} \tag{3.13}$$

where $k = 5$. The POF of DTLZ1 is a linear hyperplane with a POS of $x_i^* = 0.5,\ \ \forall x_i \in \mathbf{x_M}$. The POF of DTLZ1 is $\sum_{m=1}^{M} f_m^* = 0.5$. DTLZ1 introduces the difficulty of deception, since the search space has $(11^k - 1)$ local POFs.

$$\text{DTLZ2} = \begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{x_1\pi}{2}\right)\dots cos\left(\frac{x_{M-1}\pi}{2}\right) \\ \quad f_2(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{x_1\pi}{2}\right)\dots \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \quad \vdots \\ \quad f_M(\mathbf{x}) = (1+g(\mathbf{x_M}))\sin\left(\frac{x_{M-1}\pi}{2}\right) \\ where: \\ \quad g(\mathbf{x_M}) = \sum_{x_i\in\mathbf{x_M}}(x_i-0.5)^2 \\ \quad 0 \le x_i \le 1,\ \forall i = 1,2,\dots,n \\ \quad |\mathbf{x_M}| = k;\ \ n = M+k-1 \end{cases} \tag{3.14}$$

where $k = 10$. The POF of DTLZ2 is a sphere of radius one, namely $\sum_{m=1}^{M}(f_m^*)^2 = 1$. The POS is $x_i^* = 0.5,\ \ \forall x_i \in \mathbf{x_M}$.

$$\text{DTLZ3} = \begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{x_1\pi}{2}\right)\ldots cos\left(\frac{x_{M-1}\pi}{2}\right) \\ \quad f_2(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{x_1\pi}{2}\right)\ldots \sin\left(\frac{x_{M-1}\pi}{2}\right) \\ \quad \vdots \\ \quad f_M(\mathbf{x}) = (1+g(\mathbf{x_M}))\sin\left(\frac{x_{M-1}\pi}{2}\right) \\ where: \\ \quad g(\mathbf{x_M}) = 100\left(|\mathbf{x_M}| + \sum_{x_i\in\mathbf{x_M}}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right) \\ \quad 0 \le x_i \le 1,\ \forall i = 1,2,\ldots,n \\ \quad |\mathbf{x_M}| = k;\ \ n = M+k-1 \end{cases} \tag{3.15}$$

where $k = 10$. Similar to DTLZ2, the POF of DTLZ3 is a sphere of radius one, namely $\sum_{m=1}^{M}(f_m^*)^2 = 1$ with a POS of $x_i^* = 0.5,\ \forall x_i \in \mathbf{x_M}$. However, this MOOP has many local POFs and will test an algorithm's ability to converge to the global POF in the presence of many local POFs.

$$\text{DTLZ4} = \begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{y_1\pi}{2}\right)\ldots cos\left(\frac{y_{M-1}\pi}{2}\right) \\ \quad f_2(\mathbf{x}) = (1+g(\mathbf{x_M}))\cos\left(\frac{y_1\pi}{2}\right)\ldots \sin\left(\frac{y_{M-1}\pi}{2}\right) \\ \quad \vdots \\ \quad f_M(\mathbf{x}) = (1+g(\mathbf{x_M}))\sin\left(\frac{y_{M-1}\pi}{2}\right) \\ where: \\ \quad g(\mathbf{x_M}) = \sum_{x_i\in\mathbf{x_M}}(x_i-0.5)^2 \\ \quad y_i = x_i^{\alpha} \\ \quad 0 \le x_i \le 1,\ \forall i = 1,2,\ldots,n \\ \quad |\mathbf{x_M}| = k;\ \ n = M+k-1 \end{cases} \tag{3.16}$$

where $k = 10$ and $\alpha = 100$. Similar to DTLZ2 and DTLZ3, the POF of DTLZ4 is a sphere of radius, $\sum_{m=1}^{M}(f_m^*)^2 = 1$ and the POS is $x_i^* = 0.5,\ \forall x_i \in \mathbf{x_M}$. However,

by introducing the mapping of the x-variables, a dense set of solutions exists near the $f_M - f_1$ plane.

$$\text{DTLZ5} = \begin{cases} Minimise: \\ \quad f_1(\mathbf{x}) = (1 + g(\mathbf{x_M}))\cos(\theta_1)\cos(\theta_2)\ldots\cos(\theta_{M-1}) \\ \quad f_2(\mathbf{x}) = (1 + g(\mathbf{x_M}))\cos(\theta_1)\cos(\theta_2)\ldots\sin(\theta_{M-1}) \\ \quad \vdots \\ \quad f_M(\mathbf{x}) = (1 + g(\mathbf{x_M}))\sin(\theta_{M-1}) \\ where: \\ \quad g(\mathbf{x_M}) = \sum_{x_i \in \mathbf{x_M}} x_i^{0.1} \\ \quad \theta_i = \dfrac{\pi}{4(1+g(r))}(1 + 2g(r)x_i),\ \forall i = 1, 2, \ldots, n \\ \quad 0 \leq x_i \leq 1,\ \forall i = 1, 2, \ldots, n \\ \quad |\mathbf{x_M}| = k;\ \ n = M + k - 1 \end{cases} \tag{3.17}$$

where $k = 10$. The POF of DTLZ5 is a degenerated curve.

$$\text{DTLZ6} = \begin{cases} Minimise: \\ \quad f_1(x_1) = x_1 \\ \quad \vdots \\ \quad f_{M-1}(\mathbf{x_{M-1}}) = x_{M-1} \\ \quad f_M(\mathbf{x}) = g(\mathbf{x_M})\ \cdot h(f_1, \ldots, f_{M-1}, g) \\ where: \\ \quad g(\mathbf{x_M}) = 1 + \dfrac{9}{|\mathbf{x_M}|}\sum_{x_i \in \mathbf{x_M}} x_i \\ \quad h = M - \sum_{i=1}^{M-1} \dfrac{f_i}{1+g}(1 + \sin(3\pi f_i)) \\ \quad x_i \in \mathbb{R}^{|x_i|},\ \forall i = 1, 2, \ldots, M \end{cases} \tag{3.18}$$

where $k = 20$. DTLZ6 is based on Equation (3.12) and has 2^{M-1} disconnected Pareto optimal regions in the search space.

$$\text{DTLZ7} = \begin{cases} Minimise: \\ \quad f_j(\mathbf{x}) = \frac{1}{\lfloor \frac{n}{M} \rfloor} \sum_{\lfloor (j-1)\frac{n}{M} \rfloor}^{\lfloor j\frac{n}{M} \rfloor} x_i, \;\; j = 1, \ldots, M \\ where: \\ \quad g_j(\mathbf{x}) = f_M(\mathbf{x}) + 4f_j(\mathbf{x}) - 1 \geq 0, \forall j = 1, \ldots, (M-1) \\ \quad g_M(\mathbf{x}) = 2f_M(\mathbf{x}) + \min_{i,j=1;i\neq j}^{M-1} [f_i(\mathbf{x}) + f_j(\mathbf{x})] - 1 \geq 0 \\ \quad h = M - \sum_{i=1}^{M-1} \frac{f_i}{1+g}(1 + \sin(3\pi f_i)) \\ \quad 0 \leq x_i \leq 1, \; \forall i = 1, 2, \ldots, n \end{cases} \tag{3.19}$$

where $n = 10M$. DTLZ7 is based on Equation (3.11) and has M constraints. Its POF is a combination of a hyperplane (represented by constraint g_M) and a straight line (intersection of the first (M-1) constraints with $f_1 = f_2 = \ldots = f_{M-1}$).

Many benchmark functions for DMOO were based on the ZDT and DTLZ static MOO (SMOO) benchmark functions. The next section discusses DMOO benchmark functions that were proposed in the DMOO literature and the gaps that can be identified in the currently available DMOOPs.

3.2 Dynamic Multi-Objective Optimisation Benchmark functions

This section discusses benchmark functions used to evaluate the performance of DMOO algorithms. Benchmark functions that have been proposed in the DMOO literature are discussed in Section 3.2.1. Sections 3.2.2 and 3.2.3 present new approaches to develop DMOOPs with an isolated POF and deceptive POF respectively. New DMOOPs with complicated POSs are introduced in Section 3.2.4. Characteristics of an ideal set or suite of benchmark functions are presented in Section 3.2.5 and DMOOPs are suggested for each characteristic.

3.2.1 Dynamic Multi-Objective Optimisation Benchmark Functions Currently Used

This section discusses benchmark functions used in the DMOO literature to evaluate whether DMOO algorithms can efficiently solve DMOOPs.

Due to space constraints, only POSs and POFs with different characteristics will be illustrated in this section. In all two-objective figures f_2 refers to gh.

Guan *et al.* [74] suggested to create DMOOPs by replacing objective functions with new objective functions over time. The advantage of Guan *et al.*'s approach is that the new objective function(s) can cause a severe change in the DMOOP and by selecting the objective functions carefully, various types of changes can be incorporated into the DMOOP. Recently, Wang and Li [156] presented a DMOOP where the one subfunction of an objective function changes over time. When objective functions are changed over time, as in the approaches followed by Guan *et al.* and Wang and Li, the objective functions should be selected carefully to ensure that the resulting objective functions hinder the algorithm in finding the POF in various ways as discussed in Section 3.1.1. Another approach was followed by Jin and Sendhoff [90], where a two-objective DMOOP is constructed from a three-objective MOO function. The approach of Jin and Sendhoff has been used by various researchers [110, 111, 112, 108]. However, the adherence to the guidelines of Deb *et al.* by the benchmark functions suggested by Guan *et al.*, Wang and Li, and Jin and Sendhoff will depend on the specific objective functions that are used.

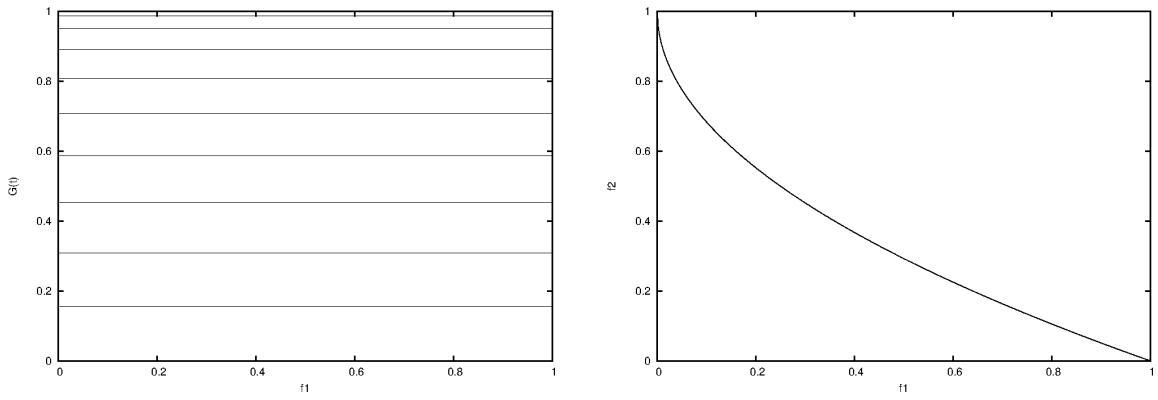
Based on the ZDT [38, 169] and DTLZ [49] functions, Farina *et al.* [58] developed the first suite of DMOOPs, namely the FDA benchmark functions. The FDA functions are constructed in such a way that they are one of the first three DMOOP types of DMOOPs, where either the POS or POF changes over time, or both the POS and POF change over time.

The DMOOPs of the FDA DMOOP suite are easy to construct and the number of decision variables are easily scalable. FDA4 and FDA5 are constructed in such a way that they are easily scalable with regards to both the number of decision variables and the number of objective functions. The FDA benchmark functions are of Type I, II and III DMOOPs and the POF of these DMOOPs is either convex, non-convex or changes from convex to concave over time. Therefore, the FDA DMOOP suite exhibits

the characteristics that benchmark functions should have, as defined by Deb *et al.* [38]. The five FDA DMOOPs are defined as follows:

$$\text{FDA1} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}, t) = 1 + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathsf{x_I} \in [0, 1]; \;\; \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 1]^{n-1} \end{cases} \tag{3.20}$$

For FDA1, values in the decision variable space (POS) change over time, but the values in the objective space (POF) remain the same. Therefore, it is a Type I DMOOP. It has a convex POF with $POF = 1 - \sqrt{f_1}$, as illustrated in Figure 3.1(b). The POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$ as illustrated in Figure 3.1(a). Appendix C explains how to determine the POS and POF of a DMOOP.

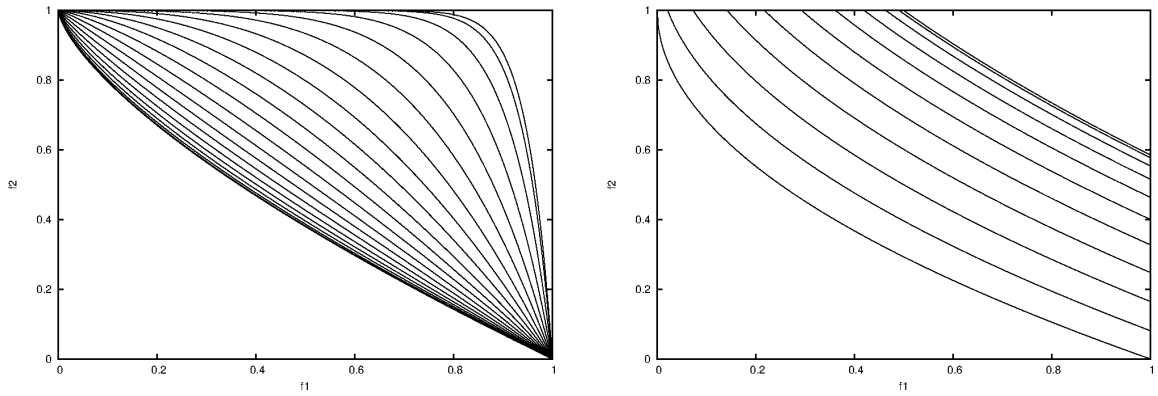


(a) POS (b) POF

Figure 3.1: POS and POF of FDA1 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{FDA2} = \begin{cases} Minimize : f(\mathsf{x},t) = (f_1(\mathbf{x}_\mathsf{I}), g(\mathbf{x}_\mathsf{II}) \cdot h\left(\mathbf{x}_\mathsf{III}, f_1(\mathbf{x}_\mathsf{I}), g\left(\mathbf{x}_\mathsf{II}\right), t\right)) \\ f_1(\mathbf{x}_\mathsf{I}) = x_1 \\ g(\mathbf{x}_\mathsf{II}) = 1 + \sum_{x_i \in \mathbf{x}_\mathsf{II}} x_i^2 \\ h(\mathbf{x}_\mathsf{III}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where : \\ H(t) = 0.75 + 0.75\sin(0.5\pi t), \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ H_2(t) = \left(H(t) + \sum_{x_i \in \mathbf{x}_\mathsf{III}} (x_i - H(t))^2\right)^{-1} \\ \mathbf{x}_\mathsf{I} \in [0,1]; \;\; \mathbf{x}_{\mathsf{II}_\mathsf{i}}, \mathbf{x}_{\mathsf{III}_\mathsf{i}} \in [-1,1] \end{cases} \tag{3.21}$$

FDA2 has a POF that changes from convex to concave. It is a Type II DMOOP, since both the POS and POF change over time. For FDA2, $POF = 1 - f_1^{H(t)^{-1}}$, as illustrated in Figure 3.2(a). The POS of FDA2 is $x_i = 0$, $\forall x_i \in \mathbf{x}_\mathsf{II}$ and $x_i = H(t)$, $\forall x_i \in \mathbf{x}_\mathsf{III}$. It should be noted that many researchers refer to FDA2 as a Type III DMOOP due to an error at the DMOOP definition in [58]. However, before the definition of FDA2 in [58], the explanation of the effect of the h function on the DMOOP states that the h function in FDA2 causes the POF to only change through a change in $\mathbf{x}_{III}$ and that FDA2 is therefore a Type II DMOOP.



(a) POF of FDA2 (b) POF of FDA3

Figure 3.2: POF of FDA2 and FDA3 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{FDA3} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I},t), g(\mathbf{x_{II}},t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I},t) = \sum_{x_i \in \mathbf{x_I}} x_i^{F(t)} \\ g(\mathbf{x_{II}},t) = 1 + G(t) + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ G(t) = |sin(0.5\pi t)| \\ F(t) = 10^{2\sin(0.5\pi t)}, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{I_i}} \in [0,1]; \;\; \mathbf{x_{II_i}} \in [-1,1] \end{cases} \tag{3.22}$$

FDA3 has a convex POF and both the values of the POS and POF change. Therefore it is called a Type II DMOOP. For FDA3, $POF = (1+G(t))\left(1 - \sqrt{\frac{f_1}{1+G(t)}}\right)$, as illustrated in Figure 9.5. The POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to the POS of FDA1 (refer to Figure 3.1(b)). The f_1 function of the two-objective FDA DMOOPs regulate the spread of solutions in objective space. Therefore, when f_1 changes over time, as is the case with FDA3, the spread of solutions in the POF changes over time.

$$\text{FDA4} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}, g(\mathbf{x_{II}},t)), \ldots, f_k(\mathbf{x}, g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x},g,t) = (1 + g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \cos\left(\frac{x_i\pi}{2}\right) \\ f_k(\mathbf{x},g,t) = (1 + g(\mathbf{x_{II}},t)) \left(\prod_{i=1}^{M-1} \cos\left(\frac{x_i\pi}{2}\right)\right) \\ \sin\left(\frac{y_{M-k+1}\pi}{2}\right), \forall k = 2, \ldots, M-1 \\ \vdots \\ f_m(\mathbf{x},g,t) = (1 + g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \sin\left(\frac{x_1\pi}{2}\right) \\ where: \\ g(\mathbf{x_{II}},t) = \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\ G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{II}} = (x_M, \ldots, x_n); \;\; x_i \in [0,1], \; \forall i = 1, \ldots, n \end{cases} \tag{3.23}$$

For FDA4, values in the decision variable space (POS) change over time, but the values in the objective space (POF) remain the same. Therefore, it is a Type I DMOOP. It has a non-convex POF with the true POF (POF) defined as $f_1^2 + f_2^2 + f_3^2 = 1$ for three objective functions, as illustrated in Figure 3.3. The POS of FDA4 is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(b)).

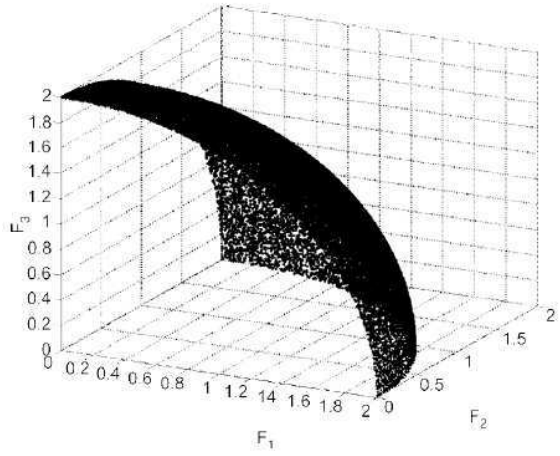


Figure 3.3: POF of FDA4 with three objective functions [58]

$$
\text{FDA5} = \begin{cases}
Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}, g(\mathbf{x_{II}},t)), \ldots, f_k(\mathbf{x}, g(\mathbf{x_{II}},t))) \\
f_1(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \cos\left(\frac{y_i\pi}{2}\right) \\
f_k(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \left(\prod_{i=1}^{M-1} \cos\left(\frac{y_i\pi}{2}\right)\right) \\
\sin\left(\frac{y_{M-k+1}\pi}{2}\right), \forall k = 2, \ldots, M-1 \\
\vdots \\
f_m(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \sin\left(\frac{y_1\pi}{2}\right) \\
where : \\
g(\mathbf{x_{II}},t) = G(t) + \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\
G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\
y_i = x_i^{F(t)}, \;\; \forall i = 1, \ldots, (M-1) \\
F(t) = 1 + 100\sin^4(0.5\pi t) \\
\mathbf{x_{II}} = (x_M, \ldots, x_n) \\
x_i \in [0,1], \; \forall i = 1, \ldots, n
\end{cases} \tag{3.24}
$$

FDA5 has a non-convex POF, where both the values in the decision variable space (POS) and the objective space (POF) change over time. Therefore, it is a Type II DMOOP. Furthermore, the spread of solutions in the POF changes over time. For FDA5 with three

objective functions, the POF is $f_1^2 + f_2^2 + f_3^2 = (1 + G(t))^2$ as illustrated in Figure 3.4. The POS of FDA5 is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(b)).

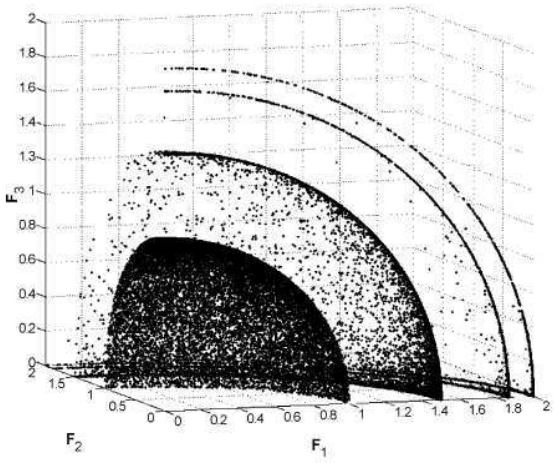


Figure 3.4: POF of FDA5 with three objective functions for four time steps [58]

Many researchers have used the FDA DMOOPs over the years as highlighted in Table 3.1. In Table 3.1 the symbol M indicates that the authors have used a modified version of the specific FDA DMOOP, *I* indicates that the authors have introduced the specific DMOOPs and the column *Other* indicates whether the authors have used DMOOPs other than the FDA set. Table 3.1 shows that most researchers used the FDA1 DMOOP, which is of Type I where the POS changes over time, but the POF remains the same. Clearly, FDA1 is the easiest DMOOP of the FDA suite to solve. Therefore, using the FDA1 DMOOP alone to test whether an algorithm can solve DMOOPs is not sufficient.

Several researchers have used the FDA2 DMOOP. However, the POF of FDA2 changes from a convex to a concave shape only for specific values of the decision variables [46, 117], as can be seen for example in [77, 78] and Figure 4.2. Therefore, even if an algorithm finds Pareto-optimal solutions, it may find a convex POF instead of a concave POF. To address this issue, several modifications to the h or g function of FDA2 have been suggested, as shown in Table 3.2. Underlying problems with FDA3 also lead to several modifications to FDA3 being suggested, as indicated in Table 3.3. In order to

test an algorithm's ability to solve Type III DMOOPs, Talukder [144] modified FDA5 to a Type III DMOO, as indicated in Table 3.4.

A generalisation of the FDA functions, DTF, was suggested by Mehnen *et al.* [117]:

$$\text{DTF} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I},t), g(\mathbf{x_{II}},t) \cdot h(f_1(\mathbf{x_I},t), g(\mathbf{x_{II}},t), t)) \\ f_1(\mathbf{x_I},t) = x_1^{\beta(t)} \\ g(\mathbf{x_{II}},t) = 1 + \sum_{x_i \in \mathbf{x_{II}}}((x_i - \gamma(t))^2 - \cos(\omega(t(\tau)))\pi(x_i - \gamma(t)) + 1) \\ h(f_1,g,t) = 2 - \left(\frac{f_1}{g}\right)^{\alpha(t)} - \left(\frac{f_1}{g}\right)|\sin(\psi(t)\pi f_1))|^{\alpha(t)} \\ where : \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0,1], \;\; \mathbf{x_{II_i}} \in [-1,1] \end{cases} \tag{3.25}$$

where β represents the spread of solutions, α the curvature of the POF, γ the optimal decision variable values or POS, ψ the number of POF sections, and ω the number of local POFs. For example, a Type II DMOOP can be constructed from DTF by setting the following parameter values: $n = 20$, $\alpha(t) = 0.2 + 4.8t^2$, $\beta(t) = 10^{2\sin(0.5\pi t)}$, $\gamma(t) = \sin(0.5\pi t)$, $\psi(t) = ts$ with $s \in \mathbb{R}$ and $\omega(t) \propto \psi(t)$.

DTF is constructed in such a way that the number of disconnected continuous POF sections, the number of local POFs, the curvature of the POF, the spread of the solutions, and the optimal decision variable values that represent the POS can be easily specified.

Table 3.2: Usage of modified FDA2 DMOOP to test algorithms' performance

Year	Authors	Changes	Modified FDA2 DMOOP
2006	Mehnen *et al.* [117]	Changed the g and H_2 functions to develop a Type III DMOOP. POF is $1 - f_1^{H_2(t)}$ and the POS is $x_i = 0$, $\forall x_i \in \mathbf{x_{II}}$ and $x_i = -1$, $\forall x_i \in \mathbf{x_{III}}$.	$\begin{cases} f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 + \sum_{x_i \in \mathbf{x_{II}}} x_i^2 + \sum_{x_i \in \mathbf{x_{III}}}(x_i + 1)^2 \\ h(\mathbf{x_{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where : \\ H_2(t) = 0.2 + 4.8t(\tau)^2 \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0,1]; \;\; \mathbf{x_{II_i}}, \mathbf{x_{III_i}} \in [-1,1] \end{cases}$ (3.26)

Continued on next page

Year	Authors	Changes	Modified FDA2 DMOOP
2007 2010	Deb *et al.* [46] and Liu *et al.* [113]	Developed a Type III DMOOP by changing the h, H and H_2 functions and the calculation of t. The POF is $1-\left(f_1^2\right)^{H_2(t)}$ and the POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{\text{II}}$ and $x_i = -1,\ \forall x_i \in \mathbf{x}_{\text{III}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} x_i^2 \\ h(\mathbf{x}_{\text{III}}, f_1, g, t) = 1 - \left(\left(\frac{f_1}{g}\right)^2\right)^{H_2(t)} \\ where: \\ H_2(t) = H(t) + \sum_{x_i \in \mathbf{x}_{\text{III}}} (x_i - H(t)/4)^2 \\ H(t) = 2\sin(0.5\pi(t-1)) \\ t = 2\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \frac{\tau_t}{\tau_{max} - \tau_t} \\ \mathbf{x}_{\text{I}} \in [0,1];\ \ \mathbf{x}_{\text{II}_i}, \mathbf{x}_{\text{III}_i} \in [-1,1] \\ \tau_{max} = 200,\ \ |\mathbf{x}_{II}| = 5,\ \ |\mathbf{x}_{III}| = 7 \end{cases}$ (3.27)
2007	Zheng [165]	Changed the h function to develop a Type III DMOOP. POF is $\left(1-\sqrt{f_1}\right)^{H_2(t)}$ and POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{\text{II}}, \mathbf{x}_{\text{III}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} x_i^2 \\ h(\mathbf{x}_{\text{III}}, f_1, g, t) = \left(1 - \sqrt{\frac{f_1}{g}}\right)^{H_2(t)} \\ where: \\ H_2(t) = \left(H(t) + \sum_{x_i \in \mathbf{x}_{\text{III}}} (x_i - H(t))^2\right)^{-1} \\ H(t) = 0.75 + 0.75\sin(0.5\pi t) \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x}_{\text{I}} \in [0,1];\ \ \mathbf{x}_{\text{II}_i}, \mathbf{x}_{\text{III}_i} \in [-1,1] \end{cases}$ (3.28)
2008 2009	Isaacs *et al.* [87] and Ray *et al.* [127]	Developed a Type III DMOOP by changing the H_2 function. Very similar to modification made by Mehnen *et al.* [117]. POF is $1 - f_1^{H_2(t)}$ and the POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{\text{II}}$ and $x_i = -1,\ \forall x_i \in \mathbf{x}_{\text{III}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} x_i^2 + \\ \qquad \sum_{x_i \in \mathbf{x}_{\text{III}}} (x_i + 1)^2 \\ h(\mathbf{x}_{\text{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where: \\ H_2(t) = 0.2 + 4.8t^2 \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x}_{\text{I}} \in [0,1];\ \ \mathbf{x}_{\text{II}_i}, \mathbf{x}_{\text{III}_i} \in [-1,1] \end{cases}$ (3.29)

Continued on next page

Year	Authors	Changes	Modified FDA2 DMOOP
2009	Salazar Lechuga [102]	Changed the h function to develop a Type III DMOOP. $1 - f_1^{H_2(t)}$ is the POF and the POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{\text{II}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} x_i^2 \\ h(\mathbf{x}_{\text{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where: \\ H_2(t) = 0.75 + 0.75\sin(0.5\pi t) \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x}_{\text{I}} \in [0,1];\ \ \mathbf{x}_{\text{II}_i}, \mathbf{x}_{\text{III}_i} \in [-1,1] \end{cases}$ (3.30)
2010	Cámara *et al.* [17] [16] [138]	Changed the H and H_2 functions to develop a Type III DMOOP. $1 - f_1^{H_2(t)}$ is the POF and the POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{\text{II}}$ and $x_i = -1,\ \forall x_i \in \mathbf{x}_{\text{III}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} x_i^2 \\ h(\mathbf{x}_{\text{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where: \\ H_2(t) = H(t) + \sum_{x_i \in \mathbf{x}_{\text{III}}} (x_i - H(t)/2)^2 \\ H(t) = z^{-\cos(\pi t/4)} \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x}_{\text{I}} \in [0,1];\ \ \mathbf{x}_{\text{II}_i}, \mathbf{x}_{\text{III}_i} \in [-1,1] \end{cases}$ (3.31)

Tang *et al.* [149] suggested a similar approach than Farina *et al.*, constructing DMOOPs based on the ZDT functions of Deb *et al.* [38]. Three objective functions are constructed similar to the DMOOPs of Farina *et al.* and provide an additional explanation of how the POF is calculated. For two objective DMOOPs, the following format is used:

$$\begin{cases} Minimise: \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}_{\text{I}}), f_2(\mathbf{x}_{\text{II}})) \\ f_1(\mathbf{x}_{\text{I}}) = f_1(\mathbf{x}_{\text{I}}) \\ f_2(\mathbf{x}_{\text{II}}) = u(t)g(\mathbf{x}_{\text{II}})v(t)\left[h\left(f(\mathbf{x}_{\text{I}}), g(\mathbf{x}_{\text{II}})v(t)\right)\right] \end{cases} \tag{3.36}$$

with $u(t)$ and $v(t)$ functions of time t. The selection of $u(t)$ and $v(t)$ lead to the construction of various types of DMOOPs:

- $u(t) = 1$ and $v(t)$ that changes over time, create a DMOOP of Type I.
- $v(t) = 1$ and $u(t)$ that changes over time, create a DMOOP of Type III.
- $u(t)$ and $v(t)$ that change over time, create a DMOOP of Type II.

The formulation of the DMOOP using Equation (3.36) can therefore lead to the creation of various types of DMOOPs by changing the values of $v(t)$ and $u(t)$. It is very similar to the FDA DMOOPs, but by formulating the DMOOP in this way, the required

Table 3.1: Usage of FDA DMOOP to test algorithms' performance

Year	Authors	FDA1	FDA2	FDA3	FDA4	FDA5	Other
2004	Farina *et al.* [58] (*I*)	x	x	x	x	x	
2005	Amato and Farina [1]	x					
2005	Shang *et al.* [135]			x		x	
2006	Hatzakis and Wallis [76]	x					
2006	Mehnen *et al.* [117]	x	M		x		x
2006	Zheng *et al.* [160]	x	x	x			
2007	Bingul [10]	x					
2007	Cámara *et al.* [19] [18]	x	x				
2007	Deb *et al.* [46]		M				
2007	Liu and Wang [112]		x	x			x
2007	Zheng [165]	x	M	M	x	x	
2007	Zhou *et al.* [166]	x M					
2008	Greeff and Engelbrecht [72]	x			x		x
2008	Isaacs *et al.* [87]	x	M				
2008	Talukder [144] [96]		x	M		M	
2008	Tan and Goh [146]	x					
2008	Wang and Dang [153]	x	x	x			
2009	Chen *et al.* [23]	x				x	
2009	Goh and Tan [67] [66]	x					x
2009	Isaacs *et al.* [88]	x	M				
2009	Ray *et al.* [127]	x	M				
2009	Salazar Lechuga [102]	x	M				
2009	Wang and Li [155]	x					x
2010	Cámara *et al.* [17] [16] [138]	x	M	M	x	x	
2010	Greeff and Engelbrecht [71]	x	x		x	x	
2010	Koo *et al.* [100]	x		x			x
2010	Liu *et al.* [113]	x	M				x
2010	Liu *et al.* [110]		x				x
2010	Wang and Li [156]	x	x	x			x
2011	Helbig and Engelbrecht [78]	x	x	x			x

type of DMOOP can be easily created. Since these functions are based on the ZDT functions, they adhere to the characteristics of benchmark functions recommended by Deb *et al.* An example Type III DMOOP using Equation (3.36) where $v(t) = 1$ and $u(t) = t^2$ is:

$$\begin{cases} Minimise : \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}})) \\ f_1(\mathbf{x_I}) = 1 - exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(\mathbf{x_{II}}) = t^2 g\left(1 - \left(\frac{f_1}{g}\right)^2\right) \\ where : \\ g = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25} \\ x_i \in [0,1],\ \forall i = 1, 2, \ldots, 10 \end{cases} \tag{3.37}$$

Table 3.3: Usage of modified FDA3 DMOOP to test algorithms' performance

Year	Authors	Changes	Modified FDA3 DMOOP
2007	Zheng [165]	Modified the f_1 function to develop a Type II DMOOP. POF is $(1 + G(t))\left(1 - \sqrt{\frac{f_1}{1+G(t)}}\right)$ and POS is $x_i = G(t),\ \forall x_i \in \mathbf{x_{II}}$.	$\begin{cases} f_1(\mathbf{x_I}, t) = \frac{1}{\lvert\mathbf{x_I}\rvert}\sum_{x_i \in \mathbf{x_I}} x_i^{F(t)} \\ g(\mathbf{x_{II}}, t) = 1 + G(t) + \sum_{x_i \in \mathbf{x_{II}}}(x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \lvert sin(0.5\pi t)\rvert \\ F(t) = 10^{2\sin(0.5\pi t)} \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{I_i}} \in [0,1];\ \ \mathbf{x_{II_i}} \in [-1,1] \end{cases}$ (3.32)
2008	Talukder [144] [96]	Changed FDA3 from a Type II to a Type III DMOOP by modifying the g function. The POF is $(1 + G(t))\left(1 - \sqrt{\frac{f_1}{1+G(t)}}\right)$ and POS is $x_i = 0,\ \forall x_i \in \mathbf{x_{II}}$.	$\begin{cases} f_1(\mathbf{x_I}, t) = \sum_{x_i \in \mathbf{x_I}} x_i^{F(t)} \\ g(\mathbf{x_{II}}, t) = 1 + G(t) + \sum_{x_i \in \mathbf{x_{II}}} x_i^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \lvert sin(0.5\pi t)\rvert \\ F(t) = 10^{2\sin(0.5\pi t)} \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{I_i}} \in [0,1];\ \ \mathbf{x_{II_i}} \in [-1,1] \end{cases}$ (3.33)
2010	Cámara *et al.* [17]	Modified the f_1 function to develop a Type II DMOOP. POF is $(1 + G(t))\left(1 - \sqrt{\frac{f_1}{1+G(t)}}\right)$ and POS is $x_i = G(t),\ \forall x_i \in \mathbf{x_{II}}$.	$\begin{cases} f_1(\mathbf{x_I}, t) = x_1^{F(t)} \\ g(\mathbf{x_{II}}, t) = 1 + G(t) + \sum_{x_i \in \mathbf{x_{II}}}(x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \lvert sin(0.5\pi t)\rvert \\ F(t) = 10^{2\sin(0.5\pi t)} \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0,1];\ \ \mathbf{x_{II_i}} \in [-1,1] \\ \mathbf{x_{II}} = (x_2, \ldots, x_n) \end{cases}$ (3.34)

Table 3.4: Usage of modified FDA5 DMOOP to test algorithms' performance

Year	Authors	Changes	Modified FDA5 DMOOP
2008	Talukder [144]	Changed FDA5 from a Type II to a Type III DMOOP by modifying the g and F functions. POF is $\sum \mathbf{f}_k^2 = (1 + G(t))^2$ and POS is $x_i = 0,\ \forall x_i \in \mathbf{x}_{II}$.	$\begin{cases} f_1(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{II}, t)) \prod_{i=1}^{M-1} \cos\left(\frac{y_i \pi}{2}\right) \\ f_k(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{II}, t)) \left(\prod_{i=1}^{M-1} \cos\left(\frac{y_i \pi}{2}\right)\right) \\ \sin\left(\frac{y_{M-k+1} \pi}{2}\right), \forall k = 1, \ldots, M-1 \\ \vdots \\ f_m(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{II}, t)) \prod_{i=1}^{M-1} \sin\left(\frac{y_1 \pi}{2}\right) \\ where: \\ g(\mathbf{x}_{II}, t) = G(t) + \sum_{x_i \in \mathbf{x}_{II}} x_i^2 \\ G(t) = \lvert sin(0.5\pi t) \rvert,\ \ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ y_i = x_i^{F(t)},\ \ \forall i = 1, \ldots, (M-1) \\ F(t) = 10^{2 \sin(0.5\pi t)} \\ \mathbf{x}_{II} = (x_M, \ldots, x_n) \\ x_i \in [0, 1],\ \forall i = 1, \ldots, n \end{cases}$ (3.35)

Wang and Li [155, 156] recently also suggested new Type I DMOOPs that are created by adapting the ZDT functions. These functions are shown in Table 3.6.

Based on the construction guidelines of Farina *et al.* [58], Goh and Tan [67] presented three DMOOPs, namely dMOP1, dMOP2 and dMOP3. dMOP1 and dMOP2 have a POF that changes from convex to concave over time, with dMOP1 being a Type III DMOOP and dMOP2 a Type II DMOOP. In the FDA DMOOP suite, FDA2 also has a POF that changes from convex to concave over time, and FDA2 is a Type II DMOOP. However, dMOP1 and dMOP2 do not suffer from the decision variable selection problem that FDA2 suffers from. dMOP1 tests whether a DMOO algorithm can solve problems where the POF changes from convex to concave but the POS remains the same over time, and dMOP2 adds the difficulty of solving this problem with a changing POS and POF. dMOP3 is very similar to FDA1, however the variable that controls the spread of the POF solutions, x_1 in FDA1, changes over time. This may cause an algorithm to struggle to maintain a diverse set of solutions as the POS changes over time. The dMOP benchmark functions are defined as follows:

$$\text{dMOP1} = \begin{cases} Minimize : f(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}), t)) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 + 9\sum_{x_i \in \mathbf{x_{II}}}(x_i)^2 \\ h(f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where : \\ H(t) = 0.75\sin(0.5\pi t) + 1.25 \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0,1];\ \mathbf{x_I} = (x_1) \\ \mathbf{x_{II}} = (x_2, \ldots, x_n) \end{cases} \tag{3.38}$$

The POF of dMOP1 changes from convex to concave over time, but the POF remains the same. Therefore, it is a Type III problem, with $POF = 1 - f_1^{H(t)}$, as illustrated in Figure 3.5. The POS of dMOP1 is $x_i = 0,\ \forall x_i \in \mathbf{x_{II}}$, similar to FDA2.

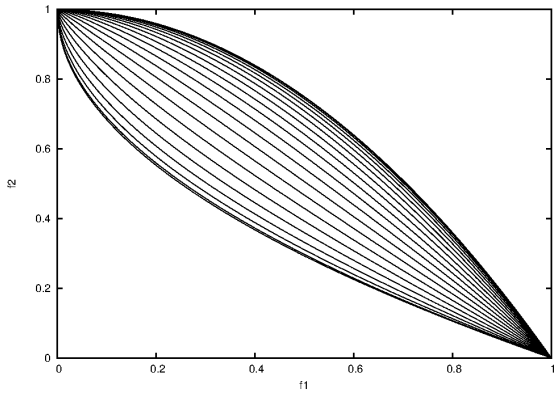


Figure 3.5: POF of dMOP1 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{dMOP2} = \begin{cases} Minimize : f(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t), t)) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}, t) = 1 + 9\sum_{x_i \in \mathbf{x_{II}}}(x_i - G(t))^2 \\ h(f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where : \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \\ G(t) = \sin(0.5\pi t) \\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0,1];\ \mathbf{x_I} = (x_1) \\ \mathbf{x_{II}} = (x_2, \ldots, x_n) \end{cases} \tag{3.39}$$

dMOP2 has a POF that changes from convex to concave, where the values in both the POS and POF change. Therefore, dMOP2 is a Type II problem, with $POF = 1 - f_1^{H(t)}$, similar to dMOP1 (refer to Figure 3.5). The POS of dMOP2 is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(b)).

$$\text{dMOP3} = \begin{cases} Minimize : f(\mathbf{x}, t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t))) \\ f_1(\mathbf{x_I}) = x_r \\ g(\mathbf{x_{II}}, t) = 1 + 9 \sum_{x_i \in \mathbf{x_{II}} \setminus x_r} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ where : \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0, 1]; \;\; r = \bigcup(1, 2, \ldots, n) \end{cases} \tag{3.40}$$

dMOP3 has a convex POF where the POS changes over time, but the POF remains the same. dMOP3 is therefore a Type I DMOOP and the spread of the POF solutions changes over time. Similar to FDA1, for dMOP3, $POF = 1 - \sqrt{f_1}$ (refer to Figure 3.1(b)) and the POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}} \setminus x_r$ (refer to Figure 3.1(a)).

More recently, Li and Zhang [105] and Deb *et al.* [48] presented MOOPs with decision variable dependencies (or linkages). Zhou *et al.* [166] modified FDA1 to incorporate dependencies between the decision variables. The modified FDA1 DMOOP is defined as follows:

$$\text{ZJZ} = \begin{cases} Minimize : f(\mathbf{x}, t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}, t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}, t) = 1 + \sum_{x_i \in \mathbf{x_{II}}} \left(x_i - G(t) - x_1^{H(t)} \right)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g} \right)^{H(t)} \\ where : \\ G(t) = \sin(0.5\pi t) \\ H(t) = 1.5 + G(t) \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0, 1]; \;\; \mathbf{x_{II}} = (x_2, \ldots, x_n) \in [-1, 2]^{n-1} \end{cases} \tag{3.41}$$

For ZJZ, the values of both the POS and POF change over time. Therefore, it is a Type II DMOOP. ZJZ's POF is similar to dMOP1 (refer to Figure 3.5) and changes from convex to concave over time, with $POF = 1 - f_1^{H(t)}$. However, there are non-linear dependencies between the decision variables that make the DMOOP more difficult to

solve. The POS of ZJZ is $x_i = G(t) + x_1^{H(t)}$, $\forall x_i \in \mathbf{x}_{\text{II}}$, as illustrated in Figure 3.6. Changes made to FDA1 to develop new DMOOPs are summarised in Table 3.5.

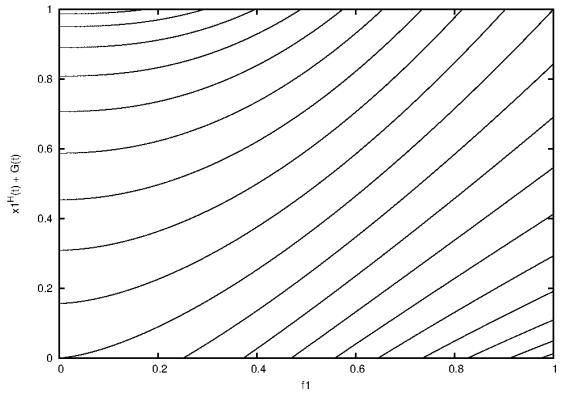


Figure 3.6: POS of ZJZ with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

Table 3.5: Usage of modified FDA1 DMOOP to test algorithms' performance

Year	Authors	Changes	Modified FDA1 DMOOP
2007	Zhou *et al.* [166]	Modified FDA1 from a Type I to a Type II DMOOP with non-linear dependencies between the decision variables. POF is $1 - f_1^{H(t)}$ and POS is $x_i = G(t) + x_1^{H(t)}$, $\forall x_i \in \mathbf{x}_{\text{II}}$.	$\begin{cases} f_1(\mathbf{x}_{\text{I}}) = x_1 \\ g(\mathbf{x}_{\text{II}}, t) = 1 + \sum_{x_i \in \mathbf{x}_{\text{II}}} \left(x_i - G(t) - x_1^{H(t)} \right)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g} \right)^{H(t)} \\ where: \\ G(t) = \sin(0.5\pi t), \quad t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ H(t) = 1.5 + G(t) \\ \mathbf{x}_{\text{I}} \in [0,1]; \quad \mathbf{x}_{\text{II}} = (x_2, \ldots, x_n) \in [-1,1]^{n-1} \end{cases}$ (3.42)

Another shortcoming of the FDA DMOOP suite is that all DMOOP objective functions consist of decision variables with the same rate of change over time. Koo *et al.* [100] suggested two new benchmark functions where each decision variable has its own rate of change, except the variable x_1 that controls the spread of solutions. These two functions, DIMP1 and DIMP2, are defined as follows:

$$\text{DIMP1} = \begin{cases} Minimize: f(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G_i(t))^2 \\ h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^2 \\ where: \\ G_i(t) = \sin\left(0.5\pi t + 2\pi\left(\frac{i}{n+1}\right)\right)^2, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} = (x_1) \in [0,1], \;\; \mathbf{x_{II}} = (x_2, x_3, \ldots, x_n) \in [-1,1]^{n-1} \end{cases} \tag{3.43}$$

The POS of DIMP1 changes over time, but the POF remains the same. Therefore, DIMP1 is a Type I DMOOP, with $POF = 1 - f_1^2$ (as illustrated in Figure 3.7) and the POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(a)).

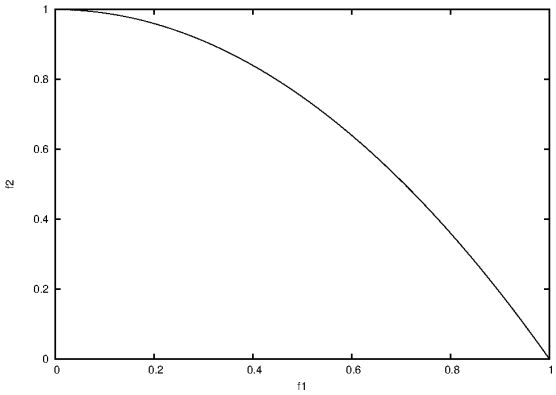


Figure 3.7: POF of DIMP1 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{DIMP2} = \begin{cases} Minimize: f(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t) \cdot (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + 2(n-1) + \\ \quad \sum_{x_i \in \mathbf{x_{II}}} [(x_i - G_i(t))^2 - \\ \qquad 2\cos(3\pi(x_i - G_i(t)))] \\ h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ G_i(t) = \sin\left(0.5\pi t + 2\pi\left(\frac{i}{n+1}\right)\right)^2, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_I} \in [0,1], \;\; \mathbf{x_{II}} \in [-2,2]^{n-1} \end{cases} \tag{3.44}$$

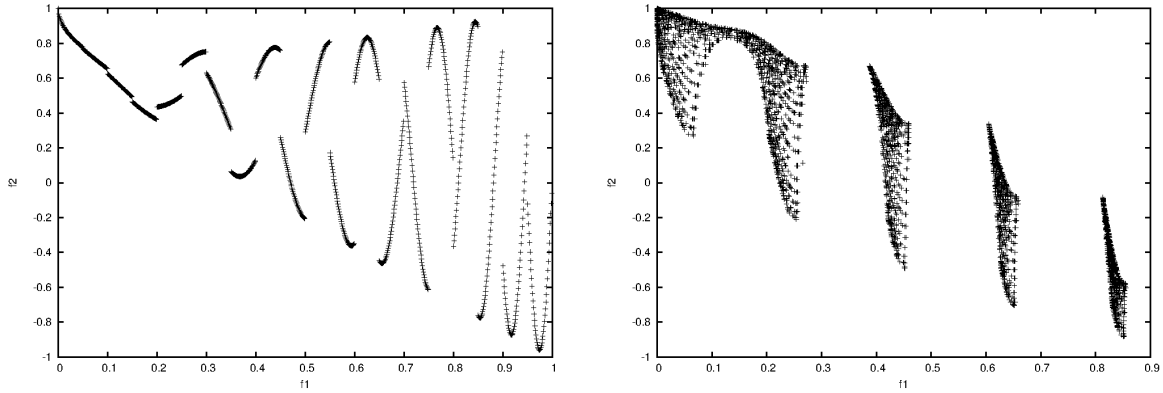
DIMP2 is a Type I problem, since its POS changes over time but its POF remains the same. Similar to FDA1, DIMP2's POF is $1 - \sqrt{f_1}$ (refer to Figure 3.1(b)) and the POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$ (refer to Figure 3.1(a)).

The FDA and dMOP MOOPs only contain DMOOPs with a continuous POF. Two discontinous functions, namely $\mathsf{TP1_{mod}}$ and $\mathsf{TP2_{mod}}$, were presented by Greeff and Engelbrecht [72]. However, these two functions do not allow easy scalability of the number of decision variables. Therefore, $\mathsf{TP1_{mod}}$ and $\mathsf{TP2_{mod}}$ do not adhere to the characteristics of benchmark functions that are recommended by Deb *et al.* Recently, Helbig and Engelbrecht [78] presented two DMOOPs with a discontinuous POF, namely HE1 and HE2. These two functions are based on the ZDT3 [169] MOOP that was developed in such a way that it adheres to the characteristics recommended by *Deb et al.* HE1 and HE2 were developed by adapting ZDT3 to be dynamic and therefore adhere to the benchmark function characteristics recommended by Deb *et al.* HE1 and HE2 are defined as:

$$\mathsf{HE1} = \begin{cases} Minimize: f(\mathbf{x}, t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}), t)) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 + \frac{9}{n-1} \sum_{x_i \in \mathbf{x_{II}}} x_i \\ h(f_1, g, t) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi t f_1) \\ where: \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0,1];\ \mathbf{x_I} = (x_1);\ \mathbf{x_{II}} = (x_2, \ldots, x_n) \end{cases} \tag{3.45}$$

$$\mathsf{HE2} = \begin{cases} Minimize: f(\mathbf{x}, t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}}), t)) \\ f_1(\mathbf{x_I}) = x_i \\ g(\mathbf{x_{II}}) = 1 + \frac{9}{n-1} \sum_{x_i \in \mathbf{x_{II}}} x_i \\ h(f_1, g, t) = 1 - \left(\sqrt{\frac{f_1}{g}}\right)^{H(t)} - \left(\frac{f_1}{g}\right)^{H(t)} \sin(10\pi f_1) \\ where: \\ H(t) = 0.75 \sin(0.5\pi t) + 1.25;\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0,1];\ \mathbf{x_I} = (x_1);\ \mathbf{x_{II}} = (x_2, \ldots, x_n) \end{cases} \tag{3.46}$$

Both HE1 and HE2 have a discontinuous POF, with various disconnected continuous sub-regions. Both are Type III DMOOPs, since their POFs change over time, but their POSs remain the same. For HE1, $POF = 1 - \sqrt{f_1} - f_1 \sin(10\pi t f_1)$ as illustrated in Figure 3.8(a), and for HE2, $POF = 1 - \left(\sqrt{f1}\right)^{H(t)} - f_1^{H(t)} \sin(0.5\pi f_1)$ as illustrated in Figure 9.9. The POS for both HE1 and HE2 is $x_i = 0$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA2.



(a) POF of HE1 (b) POF of HE2

Figure 3.8: POF of HE1 and HE2 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

Avdagić *et al.* [2] introduced an adaptation of the DTLZ problems to develop the following types of benchmark functions: Type I DMOOP where the POS changes coherently over time, but the POF remains the same, Type II DMOOP where the shape of the POS continuously changes and the POF also changes over time, and a Type II DMOOP where the number of objective functions change over time [2]. These benchmark functions are developed from the following general equation:

$$DTLZ_{Av} = \begin{cases} Minimize: & q(\mathbf{x}) = (q_1(\mathbf{x}), \ldots, q_m(\mathbf{x})) \\ q_1(\mathbf{x}) & = a_1 x_1^{c_1} x_2^{c_1} \ldots x_{m-1}^{c_1} (1 - x_m)^{c_1} g_1(\mathbf{x}) + b_1 \\ q_2(\mathbf{x}) & = a_2 x_1^{c_2} x_2^{c_2} \ldots (1 - x_{m-1})^{c_2} (1 - x_m)^{c_2} g_2(\mathbf{x}) + b_2 \\ \vdots & \\ q_{m-1}(\mathbf{x}) & = a_{m-1} x_1^{c_{m-1}} (1 - x_2)^{c_{m-1}} \ldots (1 - x_{m-1})^{c_{m-1}} (1 - x_m)^{c_{m-1}} \\ & g_{m-1}(\mathbf{x}) + b_{m-1} \\ q_m(\mathbf{x}) & = a_m (1 - x_1)^{c_m} (1 - x_2)^{c_m} \ldots (1 - x_{m-1})^{c_m} (1 - x_m)^{c_m} g_m(\mathbf{x}) \\ & + b_m \\ where: & \\ g_i = 1 - d_i \cos(20\pi x_i) & \\ a_i, b_i, c_i, d_i \in \mathbb{R} & \end{cases} \tag{3.47}$$

A Type I DMOOP with a continuously changing POS is created by using Equation (3.47) and setting the following parameter values: $a_i = 1$, $d_i = 0$, $b_i = b_i k$, where k represents the iteration and $c_i = 1$ or $c_i = 2$. Similarly, a Type II DMOOP with

continuously changing POS and POF are developed by setting the following parameter values: $a_i = 1$, $b_i = b_i k$, $c_i k = 5 b_i k$ and $d_i = 0$. To develop a Type II DMOOP with a changing number of objectives, the same parameters are used as those spesified for the Type II DMOOP, with two objective functions being used for a certain number of iterations and then three objective functions are used for the other iterations. These additional types of DMOOPs, which are not part of the FDA benchmark function set, may become important if these kind of changes occur in a real-world problem.

Recently, Huang *et al.* [84] pointed out that all DMOOPs assume that the current found POS does not affect the future POS or POF. To the best knowledge of the author of this thesis, none of the suggested DMOOPs have a POS or POF that depends on the previous POS or POF. Furthermore, most DMOOPs consist of a static number of decision variables and objective functions. Therefore, Huang *et al.* [84] introduced four DMOOPs that incorporate these scenarios, defined as follows:

$$\mathsf{T1} = \begin{cases} Minimize: f(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t)) \\ f_1(\mathbf{x}, t) = \sum_{i=1}^{d_1(t)} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \\ f_2(\mathbf{x}, t) = (x_1 - 1)^2 + \sum_{i=2}^{d_2(t)} \left(x_i^2 - x_{i-1}\right)^2 \\ where: \\ d_1(t) = \lfloor n |\sin(t)| \rfloor \\ d_2(t) = \lfloor n |\cos^3(2t)| \rfloor \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \end{cases} \tag{3.48}$$

with d_1 and d_2 varying the number of decision variables over time. The minimum for f_1 is 0 and the POS for f_1 is $x_i = 0$, $\forall i = 1, \ldots, d_1(t)$. The minimum for f_2 is 0 with the POS $x_i = 1$, $\forall i = 1, \ldots, d_2(t)$. Both the POF and POS remain static, but the number of decision variables changes over time. Therefore, T1 is a type IV DMOOP.

$$\mathsf{T2} = \begin{cases} Minimize: f(\mathbf{x}, t) = (f_1(\mathbf{x}, t), \ldots, f_m(\mathbf{x}, t)) \\ f_1(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}})) \prod_{i=1}^{m(t)-1} \cos\left(\frac{\pi x_i}{2}\right) \\ f_k(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}})) \prod_{i=1}^{m(t)-k} \cos\left(\frac{\pi x_i}{2}\right) \quad \sin\left(\frac{\pi x_{m(t)-k+1}}{2}\right), \\ \forall k = 2, \ldots, m(t) - 1 f_m(\mathbf{x}, t) = (1 + g(\mathbf{x_{II}})) \prod_{i=1}^{m(t)-1} \sin\left(\frac{\pi x_1}{2}\right) \\ where: \\ g(\mathbf{x_{II}}) = \sum_{i=1}^{m(t)} (x_i - 0.5)^2 \\ m(t) = \lfloor M |\sin(0.5\pi t)| \rfloor, \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_i} \in [0, 1] \end{cases} \tag{3.49}$$

with M representing the maximum number of objective functions and m varying the number of objective functions over time. T2 is a Type III DMOOP, since its POF changes over time, but its POS remains the same. The POS of T2 is $x_i = 0.5,\ \forall i = 1, \ldots, m(t)$ and the POF is $\sum_i^{m(t)} \mathbf{f}_i^2 = 1$.

$$\mathsf{T3} = \begin{cases} Minimize : f(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t)) \\ f_1(\mathbf{x}, t) = R(\mathbf{x}, t) \cos\left(\frac{\pi x_1}{2}\right) \\ f_2(\mathbf{x}, t) = R(\mathbf{x}, t) \sin\left(\frac{\pi x_1}{2}\right) \\ where : \\ R(\mathbf{x}, t) = \bar{R}(\mathbf{x}, \mathsf{t} - 1, t) + G(\mathbf{x}, t) \\ \bar{R}(\mathbf{x}, t) = \frac{1}{P} \sum_j^P R_j(\mathbf{x}, t-1) \\ \bar{R}(\mathbf{x}, -1) = 1 \\ G(\mathbf{x}, t) = \sum_{i=2}^n \left(x_i - \bar{R}(\mathbf{x}, t-1)\right)^2, \ \ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_1 \in [0, 1], \ \ x_i \in [\bar{R}(\mathbf{x}, t) - 100, \bar{R}(\mathbf{x}, t) + 100], \ \ \forall i = 2, \ldots, n \end{cases} \tag{3.50}$$

with the value of $R(\mathbf{x}, t)$ depending on previous values of R. Therefore, if a slight error occurs with regards to the found value of R at time t, this error will increase over time, influencing the algorithm's ability to find the solutions at the next time steps. Both the POS and POF remain static. Therefore, T3 is a Type IV DMOOP. The POS is $x_i = \bar{R}(\mathbf{x}, t-1),\ \forall i = 2, \ldots, n$. The POF is $f_1^2 + f_2^2 = 1$. Similar to T1, T4 is a type IV DMOOP, defined as:

$$\mathsf{T4} = \begin{cases} Minimize : f(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t)) \\ f_1(\mathbf{x}, t) = \sum_{i=1}^n \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \\ f_2(\mathbf{x}, t) = (x_1 - r(t))^2 + \sum_{i=2}^n \left(x_i^2 - x_{i-1}\right)^2 \\ where : \\ r(\mathbf{x}, t) = \frac{1}{n} \sum_{x_i \in \mathbf{x}} (x_i - 0) \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \end{cases} \tag{3.51}$$

with r representing the average error of the decision variables of the selected POS (POS^*). Since the POS of T4 is $x_i = 0,\ \forall i = 1, 2, \ldots, n$, the average error of the decision variables of POS^* is $r(\mathbf{x}, t) = \frac{1}{n} \sum_{x_i \in \mathbf{x}} (x_i - 0)$. The selected trade-off solution set, POS^*, is derived from the current POS by a decision making mechanism used by the decision maker. Therefore, for T4, the POF depends on the decision making mechanism used at previous time steps.

Mehnen *et al.* [117] suggested that simpler benchmark functions are required to analyse the effect of different dynamic properties in a more isolated manner. For this reason,

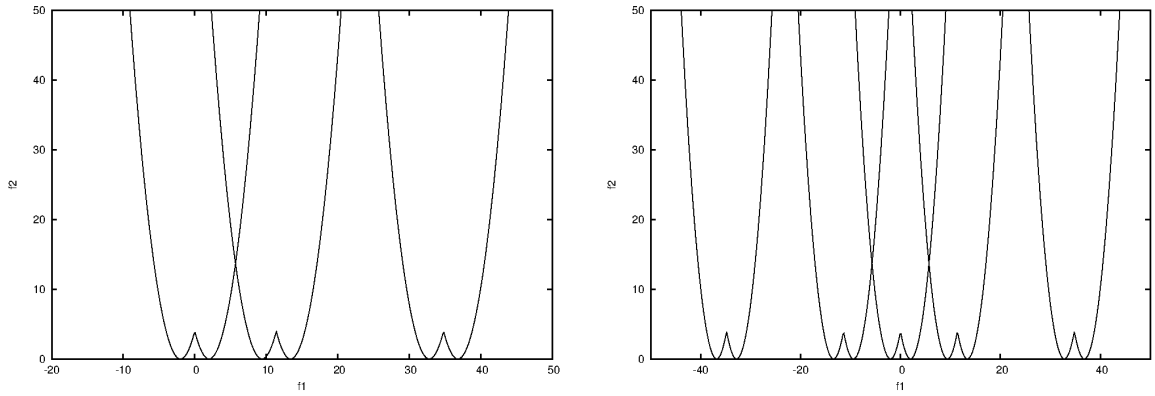
they presented the DSW DMOOPs generator that are based on the static MOOP of Shaffer [131]. The DSW DMOOPs are parabolic and are similar to the sphere function that are typically used to test whether an algorithm can solve DSOOPs. The DSW benchmark generator is defined as:

$$\text{DSW} = \begin{cases} Minimize : f(\mathbf{x},t) = (f_1(\mathbf{x},t), f_2(\mathbf{x},t)) \\ f_1(\mathbf{x},t) = (a_{11}x_1 + a_{12}|x_1| - b_1 G(t))^2 + \sum_{i=2}^{n} x_i^2 \\ f_2(\mathbf{x},t) = (a_{21}x_1 + a_{22}|x_1| - b_2 G(t) - 2)^2 + \sum_{i=2}^{n} x_i^2 \\ where : \\ G(t) = t(\tau)s, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \end{cases} \tag{3.52}$$

with s representing the severity of change. Using Equation (3.52), the following three benchmark functions are created:

$$DSW1 : \begin{cases} x \in [-50, 50]^n, \; a_{11} = 1, \; a_{12} = 0, \; a_{21} = 1, \\ a_{22} = 0, \; b_1 = 1, \; b_2 = 1 \end{cases} \tag{3.53}$$

DSW1 has a dynamic POF and POS, and is therefore a Type II DMOOP. The POS of DSW1 is $x_1 \in [G(t), G(t) + 2]$ and $x_i = 0, \; \forall i = 2, 3, \ldots, n$. The POF is $POF = \left(\sqrt{f_1} - 2\right)^2$ with $f_1 = (x_1 - G(t))^2$, as illustrated in Figure 3.9(a). DSW1 is similar to the spherical SOOP function where the center of the sphere is linearly shifted over time.



(a) POF of DSW1 (b) POF of DSW2

Figure 3.9: POF of DSW1 and DSW2 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$DSW2 : \begin{cases} x \in [-50, 50]^n, a_{11} = 0, a_{12} = 1, a_{21} = 0, \\ a_{22} = 1, b_1 = 1, b_2 = 1 \end{cases} \tag{3.54}$$

Both the POS and POF of DSW2 change over time. Therefore, DSW2 is a Type II DMOOP. DSW2 has a disconnected POS, with $x_1 \in [-G(t)-2, -G(t)] \cup [G(t), G(t)+2]$ and $x_i = 0$, $\forall i = 2, 3, \ldots, n$. If a periodical $G(t)$ is used, the POSs will join and depart periodically. The POF of DSW2 is similar to that of DSW1, namely $POF = \left(\sqrt{f_1} - 2\right)^2$, but with $f_1 = (|x_1| - G(t))^2$, as illustrated in Figure 3.9(b).

$$DSW3 : \begin{cases} x \in [-50, 50]^n, a_{11} = 1, a_{12} = 0, a_{21} = 1, \\ a_{22} = 0, b_1 = 0, b_2 = 1 \end{cases} \tag{3.55}$$

DSW3 has a changing POF and POS, and is therefore a Type II DMOOP. For DSW3 the POS is $x_1 \in [0, G(t) + 2]$ and the POF is $POF = \left(\sqrt{f_1} - G(t) - 2\right)^2$ with $f_1 = x_1^2$. Setting $b_1 = 0$ causes one border of the POS interval for x_1, namely $G(t) + 2$, to change over time, while the other border, 0, remains static.

The DMOOPs that have been discussed above are summarised in Table 3.6 (excluding the FDA and modified FDA functions summarised in Tables 3.1 to 3.4).

None of the DMOOPs discussed in this section have an isolated or deceptive POF. The next section discusses an approach to construct DMOOPs with an isolated POF.

Table 3.6: Usage of other DMOOP to test algorithms' performance

Year	Authors	Other DMOOPs	DMOOPs Definition
2004	Jin and Sendhoff [90] (I)	Constructing two-objective DMOOPs from a three-objective MOOP. Various f_1 and f_2 functions can be used to create Type I to III DMOOPs.	$\begin{cases} Minimize : \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})) \\ \text{Is changed to:} \\ Minimize : (F_1, F_2) \\ where : \\ F_1 = wf_1(\mathbf{x}) + (1-w)f_2(\mathbf{x}) \\ F_2 = wf_1(\mathbf{x}) + (1-w)f_3(\mathbf{x}) \\ \text{with } w \text{ changing over time} \end{cases}$ (3.56)
2006	Liu and Wang [111]		
2007	Li *et al.* [108]		
2007	Liu and Wang [112]		
2010	Liu *et al.* [110]		

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2005	Guan *et al.* [74]	DMOOPs created by replacing objective functions with new objective functions over time. G_1 is an example of a Type III DMOOP.	$\mathsf{G}_1 = \begin{cases} Minimise: \\ \begin{cases} G = (f_1, f_2) \text{ for } t \\ G = (f_1, f_2') \text{ for } t^* \end{cases} \\ where: \\ f_1 = x_1 \\ f_2 = g(\mathbf{x})\left(1 - \left(\frac{x_1}{g(\mathbf{x})}\right)^2\right) \\ f_2' = g(\mathbf{x})\left(1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\right) \\ g(\mathbf{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ x_i \in [0,1] \end{cases}$ (3.57) $\mathsf{G}_2 = \begin{cases} Minimise: \\ \begin{cases} G = (f_1, f_2, f_3, f_4) \text{ for } t \\ G = (f_1, f_2, f_3, f_4') \text{ for } t^* \end{cases} \\ where: \\ f_1 = (x_1 - 2)^2 + 4x_2^2 \\ f_2 = x_1^2 + (x_2 - 3)(x_3 - 3) \\ f_3 = x_2 x_3 x_4 \\ f_4 = x_1 x_4 + x_2 x_3 \\ f_4' = 1/\left(x_2^{1.5} x_3^{2.5} x_4\right) \\ x_i \in [1, 10] \end{cases}$ (3.58)
2005	Guan *et al.* [74] (cont.)	DMOOPs created by replacing objective functions with new objective functions over time.	$\mathsf{G}_3 = \begin{cases} Minimise: \\ \begin{cases} G = (f_1, f_2, f_3, f_4) \text{ for } t \\ G = (f_1, f_2, f_3', f_4') \text{ for } t^* \end{cases} \\ where: \\ f_1 = (x_1 - 2)^2 + 4x_2^2 \\ f_2 = x_1^2 + (x_2 - 3)(x_3 - 3) \\ f_3 = 1 - exp(-4x_1)\sin^6(6\pi x_1) \\ f_4 = x_1 x_4 + x_2 x_3 \\ f_3' = x_2 x_3 x_4 \\ f_4' = 1/\left(x_2^{1.5} x_3^{2.5} x_4\right) \\ x_i \in [1, 10] \end{cases}$ (3.59)

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2006	Mehnen *et al.* [117]	DMOOP DTF enabling easy specification of the number of separated POF sections, the number of local POFs, the curvature of the POF, the spread of the solutions and the optimal decision variable values that represent the POS. Type I-III DMOOPs can be created.	Equation (3.25)
		DSW DMOOP generator that is based on the static MOOP of Shaffer. DMOOP Types I-III can be created.	Equations (3.52) to (3.55)
2007	Tang *et al.* [149]	DMOOPs based on the ZDT functions of Deb *et al.* [38]. Can construct DMOOPs of Type I-III.	Equation (3.36)

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2008	Greeff and Engelbrecht [72]	$\mathsf{TP1}_{\mathsf{mod}}$ and $\mathsf{TP2}_{\mathsf{mod}}$ DMOOPs with discontinuous POFs. Both $\mathsf{TP1}_{\mathsf{mod}}$ and $\mathsf{TP2}_{\mathsf{mod}}$ are Type III DMOOPs.	$TP1_{mod}$: $$\begin{cases} Minimize : \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ f_1(\mathbf{x}) = \begin{cases} -x & \text{for } x \leq 1 \\ -2+x & \text{for } 1 < x \leq 3 \\ 4-x & \text{for } 3 < x \leq 4 \\ -4+x & \text{for } x > 4 \end{cases} \\ f_2(\mathbf{x}) = (x-5)^2 + G(t) \\ where: \\ G(t) = \lvert \sin(0.5\pi t) \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ -100 \leq x \leq 100 \end{cases} \quad (3.60)$$ $TP2_{mod}$: $$\begin{cases} Minimize : \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ f_1(\mathbf{x}) = 2 + (x_2-1)^2 - 10c_1 G(t) + \\ \qquad (x_1-2)^2 \\ f_2(\mathbf{x}) = 9x_1 + (x_2-1)^2 - 10c_2 G(t) \\ where: \\ c_1(\mathbf{x}) = \begin{cases} c_1 & \text{for } c_1 \leq 0 \\ 0 & \text{for } c_1 > 0 \end{cases} \\ c_2(\mathbf{x}) = \begin{cases} c_2 & \text{for } c_2 \leq 0 \\ 0 & \text{for } c_2 > 0 \end{cases} \\ G(t) = \lvert \sin(0.5\pi t) \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_1, x_2 \in [-20, 20] \end{cases} \quad (3.61)$$

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2009	Avdagić *et al.* [2]	Adapted the DTLZ problems to develop a Type II homogenous DMOOP where the POS changes uniformly at each iteration, a non-homogenous Type II DMOOP where the POS continuously changes and results in the POF that changes as well, and a non-homogenous Type II DMOOP where the number of objective functions change over time.	Equation (3.47)
2009	Goh and Tan [67] [66]	Three DMOOPs, namely dMOP1 (Type III), dMOP2 (Type II) and dMOP3 (Type I). dMOP1 and dMOP2 have a POF that changes from convex to concave over time. dMOP3 is very similar to FDA1, however the variable that controls the spread of the POF solutions changes over time.	Equations (3.38) to (3.40)

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2009 2010	Wang and Li [155] Wang and Li [156]	Modified ZDT functions to create the Type I DMZDT DMOOPs. POF of DMZDT1 is $1-\sqrt{f_1}$ and the POS is $\frac{\lvert x_i - t/n_t\rvert}{H(t)} = 0$.	DMZDT1: $\begin{cases} Minimize: \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t)\cdot \\ \qquad h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + \frac{9\sum_{x_i\in\mathbf{x_{II}}} \lvert y_i(t)\rvert}{D-1} \\ h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ y_i(t) = \frac{\lvert x_i - t/n_t\rvert}{H(t)}, \quad \forall i = 2,\ldots,D \\ H(t) = \max\{\lvert 1 - \frac{t}{n_t}\rvert, \lvert -1 - \frac{t}{n_t}\rvert\} \\ t = \left\lfloor \frac{f_c}{FES_c} \right\rfloor \\ \mathbf{x_I} \in [0,1]; \ \mathbf{x_{II}} = (x_2,\ldots,x_n) \in [-1,1]^{n-1} \end{cases}$ (3.62) DMZDT2: $\begin{cases} Minimize: \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t)\cdot \\ \qquad h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + \frac{9\sum_{x_i\in\mathbf{x_{II}}} \lvert y_i(t)\rvert}{D-1} \\ h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^2 \\ where: \\ y_i(t) = \frac{\lvert x_i - t/n_t\rvert}{H(t)}, \quad \forall i = 2,\ldots,D \\ H(t) = max\{\lvert 1 - \frac{t}{n_t}\rvert, \lvert -1 - \frac{t}{n_t}\rvert\} \\ t = \left\lfloor \frac{f_c}{FES_c} \right\rfloor \\ \mathbf{x_I} \in [0,1]; \ \mathbf{x_{II}} = (x_2,\ldots,x_n) \in [-1,1]^{n-1} \end{cases}$ (3.63) POF of DMZDT2 is $1-f_1^2$ and the POS is $\frac{\lvert x_i - t/n_t\rvert}{H(t)} = 0$.

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2009 2010	Wang and Li [155] Wang and Li [156]	(continued) POF of DMZDT3 is $1-\sqrt{f_1}-f_1\sin(10\pi f_1)$. The POF is discontinuous. The POS is $\frac{\lvert x_i - t/n_t\rvert}{H(t)} = 0$. DMZDT4 has many local POFs. POF of DMZDT4 is $1-\sqrt{f_1}$ and the POS is $\frac{\lvert x_i - t/n_t\rvert}{H(t)} = 0$.	DMZDT3: $\begin{cases} Minimize: \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t)\cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + \frac{9\sum_{x_i\in\mathbf{x_{II}}} \lvert y_i(t)\rvert}{D-1} \\ h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1) \\ where: \\ y_i(t) = \frac{\lvert x_i - t/n_t\rvert}{H(t)}, \quad \forall i = 2,\ldots,D \\ H(t) = max\{\lvert 1 - \frac{t}{n_t}\rvert, \lvert -1 - \frac{t}{n_t}\rvert\} \\ t = \left\lfloor \frac{f_c}{FES_c} \right\rfloor \\ \mathbf{x_I} \in [0,1]; \ \mathbf{x_{II}} = (x_2,\ldots,x_n) \in [-1,1]^{n-1} \end{cases}$ (3.64) DMZDT4: $\begin{cases} Minimize: \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t)\cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 10D - 9 + \\ \quad \sum_{x_i\in\mathbf{x_{II}}} \left[y_i(t)^2 - 10\cos(4\pi\lvert y_i\rvert)\right] \\ h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} \\ where: \\ y_i(t) = \frac{\lvert x_i - t/n_t\rvert}{H(t)}, \quad \forall i = 2,\ldots,D \\ H(t) = max\{\lvert 1 - \frac{t}{n_t}\rvert, \lvert -1 - \frac{t}{n_t}\rvert\} \\ t = \left\lfloor \frac{f_c}{FES_c} \right\rfloor \\ \mathbf{x_I} \in [0,1]; \ \mathbf{x_{II}} = (x_2,\ldots,x_n) \in [-1,1]^{n-1} \end{cases}$ (3.65)
2009 2010	Wang and Li [155] Wang and Li [156]	Type II DMOOP, WYL, where an objective changes over time.	WYL: $\begin{cases} Minimize: \\ \quad \begin{cases} DMZDT1 & \text{if } t\%4 = 0 \\ DMZDT2 & \text{if } t\%4 = 1 \\ DMZDT3 & \text{if } t\%4 = 2 \\ DMZDT4 & \text{if } t\%4 = 3 \end{cases} \\ where: \\ \% \text{ is the modulus operator} \end{cases}$ (3.66)

Continued on next page

Year	Authors	Other DMOOPs	DMOOPs Definition
2010	Koo *et al.* [100]	Type I DMOOPs DIMP1 and DIMP2, where each decision variable has its own rate of change, except the variable x_1 that controls the spread of solutions.	Equations (3.43) and (3.44)
2010	Liu *et al.* [113]	DMOP3 is a three-objective Type I DMOOP similar to FDA4 of Farina *et al.* The three-objective POF is $f_1^2 + f_2^2 + f_3^2 = 1$ and the POS is $x_i = G(t),\ \forall x_i \in \mathbf{x}_{\text{II}}$.	DMOP3: $\begin{cases} Minimize : \mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, g(\mathbf{x}_{\text{II}}, t)), \\ \quad f_2(\mathbf{x}, g(\mathbf{x}_{\text{II}}, t)), f_3(\mathbf{x}, g(\mathbf{x}_{\text{II}}, t))) \\ f_1(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{\text{II}}, t))) \cos(0.5\pi x_1) \\ \quad \cos(0.5\pi x_2) \\ f_2(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{\text{II}}, t))) \cos(0.5\pi x_1) \\ \quad \sin(0.5\pi x_2) \\ f_3(\mathbf{x}, g, t) = (1 + g(\mathbf{x}_{\text{II}}, t))) \sin(0.5\pi x_2) \\ where : \\ g(\mathbf{x}_{\text{II}}, t) = \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\ G(t) = \lvert sin(0.5\pi t) \rvert \\ t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x}_{\text{II}} = (x_3, \ldots, x_n) \\ x_i \in [0, 1],\ \forall i = 1, \ldots, n \end{cases}$ (3.67)
2011	Huang *et al.* [84]	Type IV DMOOPs where the current found POS affects the future POS or POF, a Type IV DMOOP where the number of decision variables change over time and a Type II DMOOP where the number of objective functions change over time.	Equations (3.48) to (3.51)
2011	Helbig and Engelbrecht [78]	Type III DMOOPs HE1 and HE2 with a discontinuous POF and based on the ZDT3 [169] MOOP.	Equations (3.45) and (3.46)

3.2.2 Dynamic Multi-objective Optimisation Problems with an Isolated Pareto Optimal Front

Objective functions contain flat regions when an open subset of decision variable values maps to a single objective function value. The POF of DMOOPs with objective functions that have flat regions are also referred to as an isolated POF. The lack of gradient information for the flat regions may cause difficulty for a DMOO algorithm to converge to the POF. However, no DMOOPs with an isolated POF have been proposed. Therefore, this section proposes an approach that can be used to develop DMOOPs with an isolated POF.

Huband *et al.* introduced a suite of static MOOPs referred to as the WFG benchmark functions to address shortcomings of other MOO test suites [85]. One of the shortcomings that the WFG suite addresses, is the development of MOOPs where the objective functions have flat regions. This approach is adapted so that it can be applied to current DMOOPs.

The flat regions are created by mapping the decision variables to new values using the following equation [85]:

$$y_i(x_i, A, B, C) = A + \min(0, \lfloor x_i - B \rfloor)\frac{A(B - x_i)}{B} - \min(0, \lfloor C - y \rfloor)\frac{(1 - A)(x_i - C)}{1 - C} \quad (3.68)$$

where $A, B, C \in [0, 1], B < C,\ B = 0 \implies A = 0 \wedge C \neq 0,\ C = 1 \implies A = 1 \wedge B \neq 0$. All values of x_i between B and C are mapped to the value of A. Therefore, the region between B and C forms the flat region.

This mapping can be applied to existing DMOOPs. Two examples are provided below, namely the adjustment of FDA5 (refer to Equation (3.24)) and dMOP2 (refer to Equation (3.39)):

$$\text{FDA5}_{\text{iso}} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}, g(\mathbf{x_{II}},t)), \ldots, f_k(\mathbf{x}, g(\mathbf{x_{II}},t))) \\ f_1(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \cos\left(\frac{y_i\pi}{2}\right) \\ f_k(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \left(\prod_{i=1}^{M-1} \cos\left(\frac{y_i\pi}{2}\right)\right) \\ \sin\left(\frac{y_{M-k+1}\pi}{2}\right), \forall k = 1,\ldots,M-1 \\ \vdots \\ f_m(\mathbf{x},g,t) = (1+g(\mathbf{x_{II}},t)) \prod_{i=1}^{M-1} \sin\left(\frac{y_1\pi}{2}\right) \\ where: \\ g(\mathbf{x_{II}},t) = \sum_{x_j \in \mathbf{x_{II}}} (y_j - G(t))^2 \\ G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ y_i = x_i^{F(t)}, \;\; \forall i = 1,\ldots,(M-1) \\ y_j = y_j(x_j, A, B, C), \;\; \forall x_j \in \mathbf{x_{II}} \\ F(t) = 1 + 100\sin^4(0.5\pi t) \\ \mathbf{x_{II}} = (x_M,\ldots,x_n), \;\; x_i \in [0,1], \; \forall i = 1,\ldots,n \end{cases} \tag{3.69}$$

where y_j is calculated using Equation (3.68). A, B and C can, for example, be selected as $G(t)$, 0.001 and 0.05 respectively. Similar to FDA5 (refer to Equation (3.24)), FDA5$_{iso}$ is a Type II DMOOP and the POF of FDA5$_{iso}$ is $f_1^2 + f_2^2 + f_3^2 = (1+G(t))^2$ (as illustrated in Figure 3.4). The POS of FDA5$_{iso}$ is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(b)).

$$\text{dMOP2}_{\text{iso}} = \begin{cases} Minimize : f(\mathbf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t) \cdot h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}},t), t)) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}},t) = 1 + 9\sum_{x_i \in \mathbf{x_{II}}} (y_i - G(t))^2 \\ h(f_1,g,t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where: \\ y_i = y_i(x_i, A, B, C), \;\; \forall x_i \in \mathbf{x_{II}} \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0,1]; \;\; \mathbf{x_I} = (x_1), \;\; \mathbf{x_{II}} = (x_2,\ldots,x_n) \end{cases} \tag{3.70}$$

where y_i is calculated using Equation (3.68). Example values for A, B and C are $G(t)$, 0.001 and 0.05 respectively. Similar to dMOP2 (refer to Equation (3.39)), dMOP2$_{iso}$ is a Type II problem, with $POF = 1 - f_1^{H(t)}$ (refer to Figure 3.5). The POS of dMOP2$_{iso}$ is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, similar to FDA1 (refer to Figure 3.1(b)).

The next section discusses an approach that can be used to develop DMOOPs with a deceptive POF.

3.2.3 Dynamic Multi-objective Optimisation Problems with a Deceptive Pareto Optimal Front

DMOOPs with a deceptive POF have at least two optima, but the search space favours the deceptive POF, which is a local POF and not the global POF. Some of the benchmark functions discussed in Section 3.2.1 are multi-modal. However, none of the benchmark functions discussed in Section 3.2.1 has a deceptive optimum. This section proposes an approach that can be used to adjust existing DMOOPs in such a way that the DMOOPs have a deceptive POF.

The WFG suite of Huband *et al.* [85] also introduced an approach to develop MOOPs with a deceptive POF. Similar to their approach to develop MOOPs with isolated POFs, a transformation function is used as follows:

$$y_i(x_i, A, B, C) = \left(\frac{\lfloor y - A + B \rfloor \left(1 - C + \frac{A-B}{B}\right)}{A - B} + \frac{1}{B} + \frac{\lfloor A + B - y \rfloor \left(1 - C + \frac{1-A-B}{B}\right)}{1 - A - B} \right) (|y - A| - B) + 1 \tag{3.71}$$

where $A \in (0, 1)$, $0 < B << 1$, $0 < C << 1$, $A - B > 0$ and $A + B < 1$. A represents the value at which x_i is mapped to zero and therefore the global minimum of the transformation function. B is the "aperture" size of the basin leading to A and C is the value of the deceptive optimum.

By applying this transformation (or mapping) function to existing DMOOPs, DMOOPs with a deceptive POF can be developed. For example, by calculating y_j in Equation (3.69) and y_i in Equation (3.70) using Equation (3.71), FDA5$_{iso}$ and dMOP2$_{iso}$ will have deceptive POFs. Example values for A, B and C are 0.35, 0.001 and 0.05 respectively.

Li and Zhang [106] identified a shortcoming of MOO benchmark functions, namely that the POS is defined by a simple function, e.g. $x_i = sin(0.5\pi t)$. Therefore, they presented MOOPs that have complicated POSs, where the POS is defined by non-linear curves in decision space, e.g. $x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)$, $\forall j = 2, 3, \ldots, n$. This shortcoming is also true for benchmark functions that were developed for DMOO. The next section introduces new DMOOPs with complicated POSs, based on the MOOPs of Li and

Zhang [106].

3.2.4 Dynamic Multi-objective Optimisation Problems with Complicated Pareto Optimal Sets

This section proposes new DMOOPs that have been developed based on the MOOPs of Li and Zhang [106]. The benchmark functions are constructed in such a way that the number of decision variables can be scaled easily, the resulting POFs are easily understood, and the DMOOPs hinder an algorithm to converge to the POF by requiring an algorithm to find a POS that are defined by non-linear curves. Therefore, they adhere to the benchmark function characteristics as defined by Deb *et al.*. The DMOOPs are defined as:

$$\text{HE3} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2 \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2 \\ h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where: \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ J_1 = \{j|\ \text{j is odd and } 2 \le j \le n\} \\ J_2 = \{j|\ \text{j is even and } 2 \le j \le n\} \\ x_i \in [0,1] \end{cases} \tag{3.72}$$

The POF changes over time, but the POS remains the same. Therefore, HE3 is a Type III DMOOP. The POS and POF of HE3 are:

$$POS : x_j = x_1^{0.5\left(\frac{3(j-2)}{n-2}\right)}, \;\; \forall j = 2,3,\ldots,n.$$

$$POF : y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$$

The POF and POS of HE3 are illustrated in Figures 3.10 and 3.11 respectively. It is important to note that, unlike most of the other DMOOPs, the POS of HE3 to HE10

are different for each decision variable.

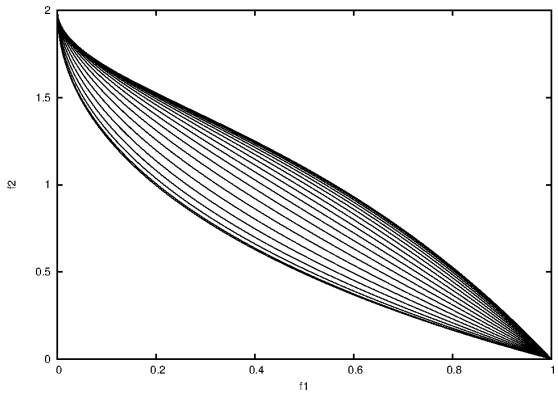
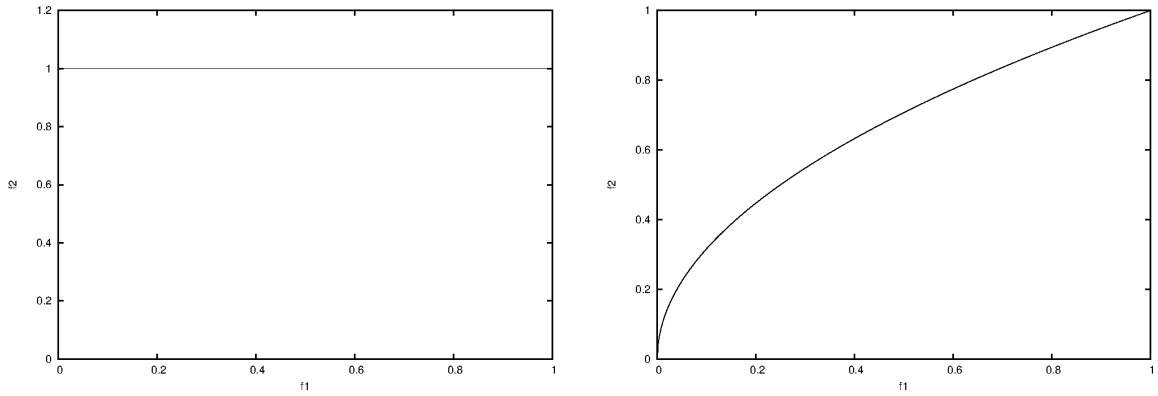


Figure 3.10: POF of HE3 with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations



(a) POS of x_2

(b) POS of x_5

Figure 3.11: POS of HE3 for two decision variables, x_2 and x_5, with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{HE4} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \right)^2 \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \right)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g} \right)^{H(t)} \\ where : \\ H(t) = 0.75 \sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ J_1 = \{j |\ \text{j is odd and } \; 2 \leq j \leq n\} \\ J_2 = \{j |\ \text{j is even and } \; 2 \leq j \leq n\} \\ x_1 \in [0,1], \;\; x_i \in [-1,1], \;\; \forall i = 2,3,\ldots,n \end{cases} \tag{3.73}$$

The POF of HE4 changes over time, but the POS remains the same. Therefore, HE4 is a Type III DMOOP. The POS and POF of HE4 are:

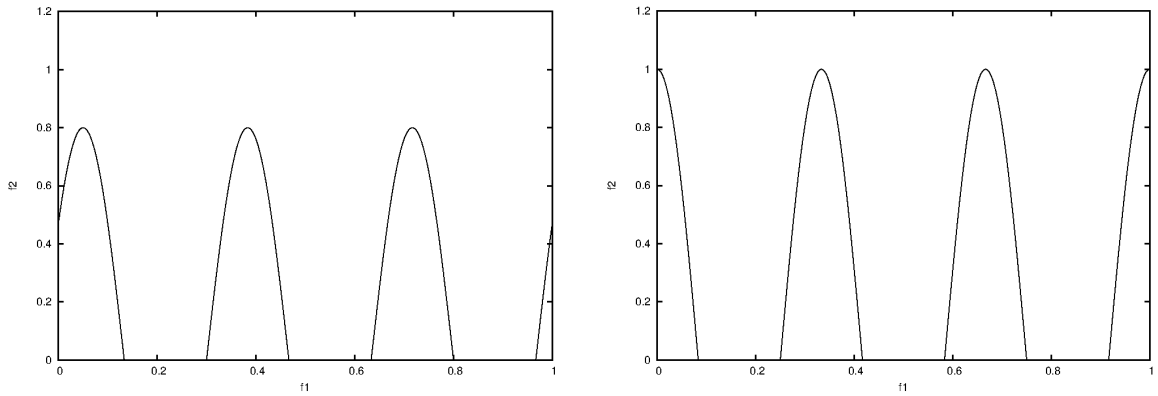
$$POS : x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), \;\; \forall j = 2,3,\ldots,n.$$

$$POF : y = (2 - \sqrt{x_1}) \left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}} \right)^{H(t)} \right]$$

The POS of HE4 is illustrated in Figure 3.12. The POF is similar to the POF of HE3 (refer to Figure 3.10).

$$\text{HE5} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - 0.8 x_1 \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - 0.8 \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) \right)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g} \right)^{H(t)} \\ where : \\ H(t) = 0.75 \sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ J_1 = \{j |\ \text{j is odd and } \; 2 \leq j \leq n\} \\ J_2 = \{j |\ \text{j is even and } \; 2 \leq j \leq n\} \\ x_1 \in [0,1], \;\; x_i \in [-1,1], \;\; \forall i = 2,3,\ldots,n \end{cases} \tag{3.74}$$

HE5 is a Type III DMOOP, since the POF changes over time, but the POS remains the same. The POS and POF of HE5 are:



(a) POS of x_2 (b) POS of x_5

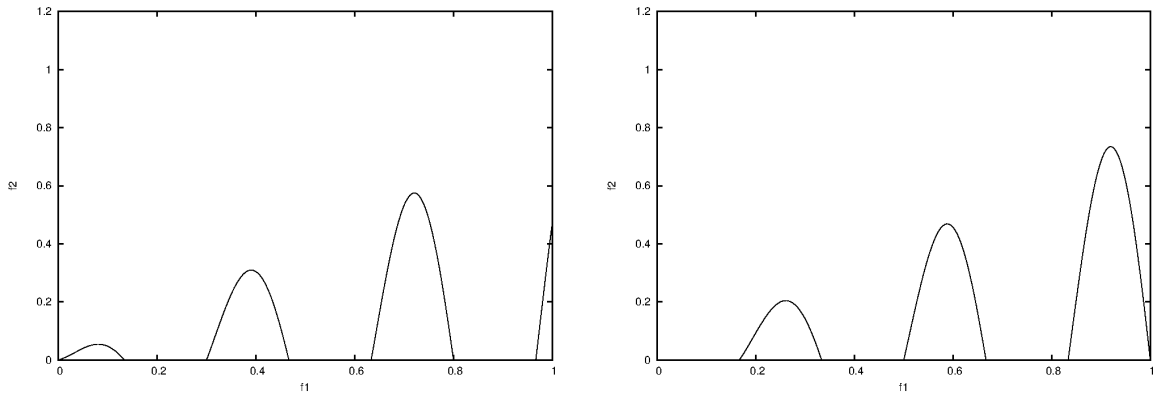
Figure 3.12: POS of HE4 for two decision variables, x_2 and x_5, with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$POS: x_j = \begin{cases} 0.8x_1 \cos\left(6\pi x_1 + \frac{j\pi}{n}\right), & j \in J_1 \\ 0.8x_1 \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), & j \in J_2 \end{cases}$$
$$POF: y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}}\right)^{H(t)}\right]$$

The POS of HE5 is illustrated in Figure 3.13. The POF is similar to the POF of HE3, illustrated in Figure 3.10.

$$\text{HE6} = \begin{cases} Minimize: \mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}), g(\mathbf{x}, t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x}, t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - 0.8x_1 \cos\left(\frac{6\pi x_1 + \frac{j\pi}{n}}{3}\right)\right)^2 \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - 0.8\cos\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where: \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ J_1 = \{j|\ \text{j is odd and } 2 \le j \le n\} \\ J_2 = \{j|\ \text{j is even and } 2 \le j \le n\} \\ x_1 \in [0, 1], \;\; x_i \in [-1, 1], \;\; \forall i = 2, 3, \ldots, n \end{cases} \tag{3.75}$$

For HE6, the POF changes over time, but the POS remains the same. Therefore, HE6



(a) POS of x_2 (b) POS of x_5

Figure 3.13: POS of HE5 for two decision variables, x_2 and x_5, with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

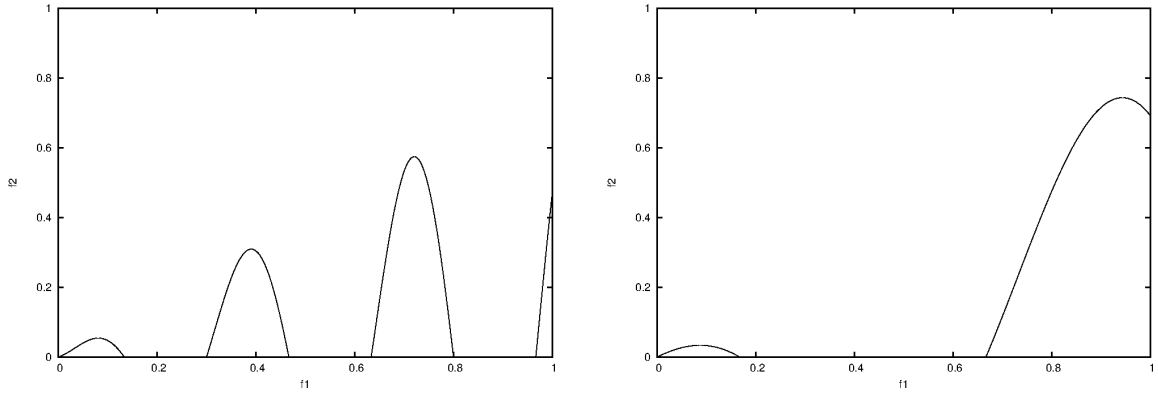
is a Type III DMOOP. The POS and POF of HE6 are:

$$POS : x_j = \begin{cases} 0.8x_1 \cos\left(\frac{6\pi x_1 + \frac{j\pi}{n}}{3}\right), & j \in J_1 \\ 0.8x_1 \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), & j \in J_2 \end{cases}$$

$$POF : y = (2 - \sqrt{x_1}) \left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}}\right)^{H(t)}\right]$$

The POF of HE6 is similar to the POF of HE3 (refer to Figure 3.10). The POS of HE6 is illustrated in Figure 3.14.

$$\text{HE7} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}), g(\mathbf{x}, t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x}, t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \cos\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\ where : \\ H(t) = 0.75 \sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ J_1 = \{j |\ \text{j is odd and } \; 2 \leq j \leq n\} \\ J_2 = \{j |\ \text{j is even and } \; 2 \leq j \leq n\} \\ x_1 \in [0, 1], \;\; x_i \in [-1, 1], \;\; \forall i = 2, 3, \ldots, n \end{cases} \tag{3.76}$$



(a) POS of x_2 (b) POS of x_5

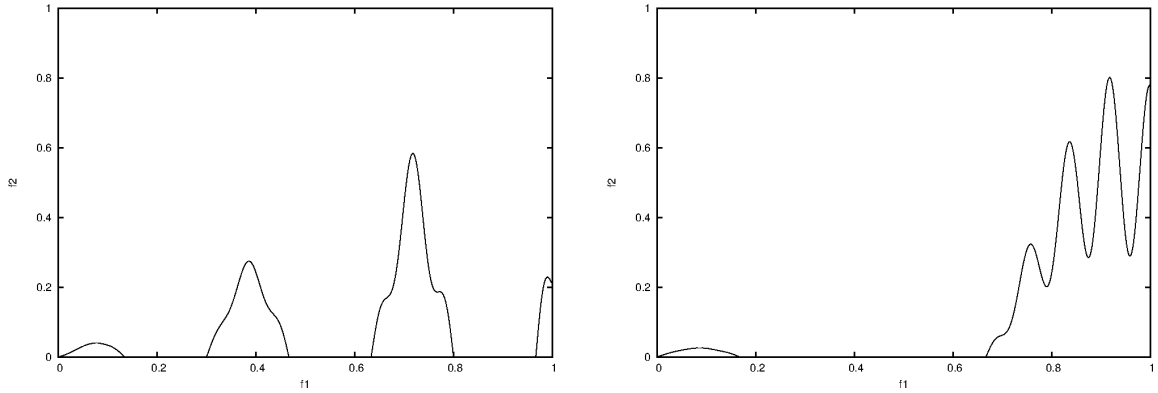
Figure 3.14: POS of HE6 for two decision variables, x_2 and x_5, with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

HE7 is a Type III DMOOP, since the POF changes over time, but the POS remains the same. The POS and POF of HE7 are:

$$POS : x_j = \begin{cases} a\cos\left(\frac{6\pi x_1 + \frac{j\pi}{n}}{3}\right), & j \in J_1 \\ a\sin\left(6\pi x_1 + \frac{j\pi}{n}\right), & j \in J_2 \\ \text{with:} \\ a = \left[0.3x_1^2\cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \end{cases}$$

$$POF : y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}}\right)^{H(t)}\right]$$

The POS of HE7 is illustrated in Figure 3.15. The POF is similar to the POF of HE3, as illustrated in Figure 3.10.



(a) POS of x_2 (b) POS of x_5

Figure 3.15: POS of HE7 for two decision variables, x_2 and x_5, with $n_t = 10$ and $\tau_t = 10$ for 1000 iterations

$$\text{HE8} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{j \in J_1} \left(4y_j^2 - \cos(8y_i\pi) + 1.0\right) \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{j \in J_2} \left(4y_j^2 - \cos(8y_i\pi) + 1.0\right) \\ h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where : \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ J_1 = \{j |\ \text{j is odd and } \; 2 \leq j \leq n\} \\ J_2 = \{j |\ \text{j is even and } \; 2 \leq j \leq n\} \\ y_j = x_j - x_1^{\left(0.5\left(1.0 + \frac{3(j-2)}{n-2}\right)\right)}, \;\; \forall j = 2,3,\ldots,n \\ x_i \in [0,1] \;\; \forall i = 1,2,\ldots,n \end{cases} \tag{3.77}$$

The POF of HE8 changes over time, but the POS remains the same. Therefore, HE8 is a Type III DMOOP. The POS (refer to Figure 3.11) and POF (refer to Figure 3.10) of HE8 are:

$$POS : x_j = x_1^{0.5\left(\frac{3(j-2)}{n-2}\right)}, \;\; \forall j = 2,3,\ldots,n.$$
$$POF : y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}}\right)^{H(t)}\right]$$

$$\text{HE9} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t) \cdot h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(4\sum_{j\in J_1} y_j^2 - \prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2.0) \\ g(\mathbf{x}) = 2 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2.0) \\ h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where: \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ J_1 = \{j|\ \text{j is odd and } 2 \le j \le n\} \\ J_2 = \{j|\ \text{j is even and } 2 \le j \le n\} \\ y_j = x_j - x_1^{\left(0.5\left(1.0+\frac{3(j-2)}{n-2}\right)\right)}, \;\; \forall j = 2,3,\ldots,n \\ x_i \in [0,1]\ \forall i = 1,2,\ldots,n \end{cases} \tag{3.78}$$

For HE9, the POF changes over time, but the POS remains the same. Therefore, HE9 is a Type III DMOOP. The POS (refer to Figure 3.11) and POF (refer to Figure 3.10) of HE9 are:

$$POS : x_j = x_1^{0.5\left(\frac{3(j-2)}{n-2}\right)}, \;\; \forall j = 2,3,\ldots,n.$$
$$POF : y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$$

$$\text{HE10} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x}), g(\mathbf{x},t)\cdot \\ \qquad h(f_1(\mathbf{x}), g(\mathbf{x},t))) \\ f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - \sin(6\pi x_1 + \frac{j\pi}{n})\right)^2 \\ g(\mathbf{x}) = 2 - x_1^2 + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - \sin(6\pi x_1 + \frac{j\pi}{n})\right)^2 \\ h(f_1,g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where: \\ H(t) = 0.75\sin(0.5\pi t) + 1.25, \;\; t = \frac{1}{n_t}\left\lfloor\frac{\tau}{\tau_t}\right\rfloor \\ J_1 = \{j|\ \text{j is odd and } 2 \le j \le n\} \\ J_2 = \{j|\ \text{j is even and } 2 \le j \le n\} \\ x_i \in [0,1]\ \forall i = 1,2,\ldots,n \end{cases} \tag{3.79}$$

The POF of HE10 changes over time, but the POS remains the same. Therefore, HE10 is a Type I DMOOP. The POS (refer to Figure 3.12) and POF (refer to Figure 3.10) of HE10 are:

$$POS : x_j = sin\left(6\pi x_1 + \frac{j\pi}{n}\right), \ \forall j = 2, 3, \ldots, n.$$
$$POF : y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2 - \sqrt{x_1}}\right)^{H(t)}\right]$$

Taking into consideration the benchmark functions currently being used for DMOO (discussed in Section 3.2.1) and the ideal characteristics of benchmark functions (discussed in Section 3.1.1), it becomes clear that many different types of DMOOPs have been suggested to be used as benchmark functions. Therefore, when a new DMOO algorithm has been developed, the selection of benchmark functions to test the algorithm's ability to solve DMOOPs is a daunting task.

3.2.5 Ideal Set of Dynamic Multi-objective Optimisation Benchmark Functions

This section presents the characteristics of an ideal benchmark function set and suggests DMOOPs that can be used to sufficiently test an algorithm's ability to solve DMOOPs.

From Section 3.2.1 the following characteristics were identified that an ideal MOO (static or dynamic) set of benchmark functions should have:

1. The set of benchmark functions should test for the following difficulties to converge towards the POF:
   - Multimodality.
   - Deception.
   - Isolated optimum.
2. The set of benchmark functions should test for the following difficulties to obtain a diverse set of solutions:
   - Convexity or non-convexity in the POF.
   - Discontinuous POF, i.e. disconnected sub-regions that are continuous.
   - Non-uniform distribution of solutions in the POF.
3. The benchmark functions should have various types or shapes of POSs, where the POS is also non-linear curves and not only linear functions.

4. The benchmark functions should have decision variables with dependencies or linkages.

In addition, the following characteristics were identified that an ideal DMOOP benchmark function suite should have:

1. The set of benchmark functions should have a non-uniform distribution of solutions in the POF, where the distribution of solutions changes over time.
2. The shape of the POFs should change over time from convex to non-convex or vice versa.
3. The benchmark functions should have decision variables with different rates of change over time.
4. The benchmark functions should include cases where the POF depends on the values of previous POSs or POFs.
5. The benchmark functions should enable changing the number of decision variables over time.
6. The benchmark functions should enbale changing the number of objective functions over time.

For each characteristic a set of DMOOPs was identified from Sections 3.2.1, 3.2.2 and 3.2.3. The proposed ideal benchmark functions suite from which DMOOPs can be selected to evaluate the performance of dynamic MOAs (DMOAs) are presented in Tables 3.7 and 3.8.

When a selection of DMOOPs are made, it should be done in such a way that various types of DMOOPs are selected for each characteristic, or the benchmark suite should at least have type II DMOOPs for some characteristics. The reason for this is to ensure that an algorithm can overcome a certain difficulty in various types of DMOO environments.

In addition to the benchmark functions listed in Table 3.7, the generic benchmark function generators can be used to create benchmark functions of various types with specific characteristics as outlined in this section, for example DTF (refer to Equation (3.25)), DTLZ$_{Av}$ (refer to Equation (3.47)), DSW (refer to Equation (3.52)), and the DMOOP of Tang (refer to Equation (3.36)).

Table 3.7: Set of DMOO Benchmark Functions for each Identified Characteristic for MOOPs in general

Characteristic	**DMOOP Type: Suggested DMOOPs**
1. DMOOPs that cause difficulties to converge towards the POF: – Multi-modal DMOOPs	Type I: DMZDT4 (Equation (3.65))
– DMOOPs with a deceptive optimum	Various: DMOOPs developed according to Section 3.2.3
– DMOOPs with an isolated optimum	Various: DMOOPs developed according to Section 3.2.2
2. DMOOPs that cause difficulties to find a diverse set of solutions:	
– DMOOP with convex POF	• Type I: FDA1 (Equation (3.20)), DMZDT1 (Equation (3.62)) • Type II: Modified FDA3 functions (refer to Table 3.6) • Type III: dMOP1 (Equation (3.38))
– DMOOPs with non-convex POF	• Type I: DMZDT2 (Equation (3.63)), FDA4 (Equation (3.23)), DMOP3 (Equation (3.67)) • Type II: FDA5 (Equation (3.24)) • Type III: Modified FDA5 functions (Equation (3.35)
– DMOOPs with discontinuous POF	• Type I: DMZDT3 (Equation (3.64)) • Type III: HE1 (Equation (3.45)), HE2 (Equation (3.46))
– DMOOPs with non-uniform spread of solutions	• Type I: dMOP3 (Equation (3.40)) • Type II: FDA5 (Equation (3.24)), Modified FDA3 functions (refer to Table 3.6) • Type III: modified FDA5 functions (Equation (3.35)
3. DMOOPs with various types or shapes of POSs	• Type I, II: DTLZ$_{Av}$ (Equation (3.47)) • Type II: ZJZ (Equation (3.41)), DSW2 (Equation (3.54)), DSW3 (Equation (3.55)) • Type III: HE3-HE10 (Equations (3.72) - (3.79)), Modified FDA2 functions (Equations (3.26)- (3.31))
4. DMOOPs with dependencies or linkages between the decision variables	• Type II: ZJZ (Equation (3.41))

Table 3.8: Set of DMOO Benchmark Functions for each Identified Characteristic for DMOOPs

Characteristic	DMOOP
1. DMOOPs where the distribution of solutions changes over time	• Type I: dMOP3 (Equation (3.40)) • Type II: FDA5 (Equation (3.24)), Modified FDA3 functions (refer to Table 3.6) • Type III: modified FDA5 functions (Equation (3.35)
2. DMOOPs where the POF changes from convex to non-convex or vice versa over time	• Type II: dMOP2 (Equation (3.39)), ZJZ (Equation (3.41)) • Type III: dMOP1 (Equation (3.38)), Modified FDA2 functions (Equations (3.26)- (3.31))
3. DMOOPs with decision variables with different rates of change	• Type I: DIMP1 (Equation (3.43)), DIMP2 (Equation (3.44))
4. DMOOPs where the current POF depends on the previous POF	• Type IV: T3 (Equation (3.50)), T4 (Equation (3.51))
5. DMOOPs where the number of decision variables change over time	• Type IV: T1 (Equation (3.48))
6. DMOOPs where the number of objective functions change over time	• Type I, II: DTLZ$_{Av}$ (Equation (3.47)) • Type III: T2 (Equation (3.49))

3.3 Summary

This chapter provided an overview of the benchmark functions that have been used to test whether DMOO algorithms can overcome specific difficulties that can occur in real-world problems. MOO benchmark functions that have been adapted for DMOO, namely the ZDT and DTLZ MOOPs, were discussed. Since there is no standard DMOO benchmark functions yet, this chapter discussed the characteristics that an ideal benchmark function suite should have and suggested benchmark functions that can be used to test for each

of these recommended characteristics.

The next chapter provides an overview of performance measures suggested for DMOO.

Chapter 4

Analysis of Dynamic Multi-objective Optimisation Performance Measures

> *"You get what you measure. Measure the wrong thing and you get the wrong behaviors."* – John H. Lingle

In order to determine whether an algorithm can solve DMOOPs efficiently, it should be tested against DMOOPs that test the ability of the algorithm to overcome certain difficulties, called benchmark functions. However, to quantify the performance of the algorithm, and to compare the performance of one algorithm against that of another algorithm, performance measures are required.

One of the main problems in the field of DMOO is a lack of standard benchmark functions and standard performance measures. An analysis of benchmark functions for DMOO was presented in Chapter 3. This chapter evaluates the current performance measures presented in the DMOO literature to establish whether they efficiently evaluate the performance of DMOO algorithms.

Section 4.1 discusses performance measures that have been used for MOO and adapted for DMOO. Performance measures currently used in the DMOO literature are discussed in Section 4.2. Section 4.3 highlights issues with current performance measures that are frequently used to measure the performance of DMOO algorithms. Finally, a summary is provided in Section 4.4.

4.1 Static Multi-objective Optimisation Performance Measures

Performance measures enable the quantification and measuring of an algorithm's performance with regards to a specific requirement, such as the number of non-dominated solutions found, closeness to the true POF (accuracy), and the diversity or spread of the solutions. According to Zitzler *et al.* [173], a performance measure is defined as follows:

Definition 4.1. Performance Measure: A m-ary performance measure, P, is a function $P : \Omega^m \rightarrow \mathbb{R}$, that assigns each of the m approximated POFs, $POF_1^*, POF_2^*, \ldots,$ POF_m^* a real value $P(POF_1^*, POF_2^*, ..., POF_m^*)$.

This section discusses static MOO measures that have been adapted in the literature and used in DMOO. The discussion on static MOO performance measures is by no means complete, and the reader is referred to [40, 99, 114, 167] for detailed information on performance measures used for static MOO.

Outperformance relations that are used to evaluate performance measures are discussed in Section 4.1.1. Section 4.1.2 discusses performance measures that quantify an algorithm's performance with regards to accuracy, i.e. the found non-dominated solutions' (POF^*) closeness to the true POF (POF). Performance measures that measure the diversity or spread of the found solutions are discussed in Section 4.1.3. Section 4.1.4 discusses performance measures that measure the overall quality of the found solutions, taking into account both accuracy and diversity.

4.1.1 Outperformance Relations

This section discusses outperformance relations that performance measures should ideally adhere to. When an algorithm solves a MOOP where the objective functions are in conflict with one another, the algorithm tries to find the best possible set of non-dominated solutions, i.e. a set of solutions that are as close as possible to POF and where the solutions are diverse and evenly spread along POF^*. Once POF^* is found, a decision maker selects one of these solutions according to his/her own defined preferences.

Hansen and Jaszkiewicz [75] introduced an outperformance relation under the following assumptions:

- The preferences of the decision maker is not known a priori.
- Let POF_A^* and POF_B^* be two approximated POFs. Then, POF_A^* outperforms POF_B^* if the decision maker finds:
  - a better solution in POF_A^* than in POF_B^* for specific preferences, and
  - for another set of preferences the solution selected from POF_A^* is not worse than solutions found in POF_B^*.
- All possible preferences of the decision maker can be modelled with utility functions that belong to a set of utility functions, U.

Definition 4.2. Outperformance Relation (subject to a set of utility functions): Let A and B be two sets representing approximations of the same POFs. Let $U|A > B \subseteq U$ denote a subset of utility functions for which A is better than B, i.e. $U|A > B = \{u^* \in U|u^*(A) > u^*(B)\}$. Then A outperforms (O) B if there exists a non-empty subset of the utility functions set U for which A achieves better values than B, while the opposite is not true. Mathematically the outperformance relation is defined as: $A\ O_U\ B$ if $U(A > B) \neq \varnothing$ and $U(B > A) = \varnothing$.

The weakest assumption about the decision maker's preferences that is generally made when solving MOOPs is that the utility function is compatible with the dominance relation, i.e. the decision maker prefers non-dominated solutions [128]. Therefore, the decision maker can limit his/her selection of the best solution to the set $ND(A \cup B)$, i.e. the non-dominated solutions in $A \cup B$. Based on the dominance relation assumption, the following three dominance based relations were defined by Hansen and Jaszkiewicz [75]:

Definition 4.3. Weak Outperformance: A weakly outperforms (O_W) B if, for each solution in B, there exists a solution in A that is equal to or dominates the solution in B and at least one solution in A is not contained in B. Weak outperformance is defined as: $A\ O_W\ B$ if $A \neq B$ and $ND(A \cup B) = A$, where $ND(A \cup B)$ is the set of non-dominated solutions obtained from the unified set, $(A \cup B)$.

Definition 4.4. Strong Outperformance: A strongly outperforms (O_S) B if, for each solution in B, there exists a solution in A that is equal to or dominates the solution in B and at least one solution in B is dominated by a solution in A. Strong outperformance is mathematically defined as: $A\, O_S\, B$ if $A \neq B$, $ND(A \cup B) = A$ and $B \backslash ND(A \cup B) = \varnothing$.

Definition 4.5. Complete Outperformance: A completely outperforms B if each solution in B is dominated by a solution in A. Complete outperformance is defined as: $A\, O_C\, B$ if $A \neq B$, $ND(A \cup B) = A$ and $B \cap ND(A \cup B) = \varnothing$.

These outperformance relations only identify whether one set is better than another set, but doesn't quantify with how much the one set is better than the other. However, according to Knowles [99], performance measures that are not compatible with these outperformance relations, cannot be relied on to provide evaluations that are compatible with Pareto dominance. Based on the outperformance relations, Hansen and Jaszkiewicz [75] defined compatibility and weak compatibility with an outperformance relation:

Definition 4.6. Weak Compatibility: A performance measure is weakly compatible with an outperformance relation if, for each pair of non-dominated sets, A and B, where $A\, O\, B$, the performance measure will evaluate A as being not worse than B.

Definition 4.7. Compatibility: A performance measure is compatible with an outperformance relation if for each pair of non-dominated sets, A and B, where $A\, O\, B$, the performance measure will evaluate A as being better than B.

In addition to the outperformance relations, Knowles [99] introduced the concepts of monotony and relativity that are important when evaluating the efficiency of performance measures.

Definition 4.8. Monotony: Let C be a set containing a new non-dominated solution. Then the performance measure will evaluate $A \bigcup C$ as being not worse than A.

Definition 4.9. Relativity: Let D be a set containing the solutions of the true POF. Then the performance measure will evaluate D as being better than any POF^* found by the algorithms being evaluated.

Weak compability with O_W is sufficient for weak monotony and weak relativity [99].

From the above definitions it should be clear that $O_C \subset O_S \subset O_W$, i.e. complete outperformance is the strongest outperformance relation, and weak outperformance is the weakest outperformance relation. Therefore, it is most difficult for a performance measure to be compatible with O_W, and the easiest for a performance measure to be compatible with O_C [99]. According to Knowles [99], performance measures that are not compatible with these outperformance relations, cannot be relied on to provide evaluations that are compatible with Pareto dominance.

If a performance measure is compatible with the concept of monotony, it will not decrease a set's evaluation if a new non-dominated point is added, which adheres to the goal of finding a diverse set of solutions. Furthermore, if a performance measure does not adhere to the concept of relativity, it will evaluate an approximation set as being better than the true POF, which is not accurate.

Knowles [99] evaluated the performance measures frequently used in MOO according to their compatibility with the outperformance relations defined by Hansen and Jaszkiewicz. The MOO performance measures' compatability with the outperformance relations are highlighted below where the performance measures are discussed in more detail.

4.1.2 Accuracy Performance Measures

This section discusses performance measures that are used to measure the accuracy of POF^* that is found by a MOO algorithm, i.e. how close POF^* is to POF.

Generational Distance

The generational distance (GD) measures the convergence of the approximated set towards the true POF (POF). The GD is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n_{POF^*}} d_i^2}}{n_{POF^*}} \tag{4.1}$$

where n_{POF^*} is the number of solutions in POF^* and d_i is the Euclidean distance in the objective space between solution i of POF^* and the nearest member of POF'. POF' contains sampled solutions of POF that are used as a reference set. Therefore, GD determines how close POF^* is to the sampled solutions of POF.

GD is easy to calculate and intuitive. However, knowledge about POF is required and a reference set, POF', has to be available. It is important that the reference set contains a diverse set of solutions, since the selection of the solutions will impact on the results obtained from this performance measure. Furthermore, since the distance metric is used, scaling and normalisation of the objectives is required. GD is not weakly compatible with O_W, but is compatible with O_S. Unfortunately, this performance measure will rate a POF^* with only one solution that is on POF' better than another POF^* that has one hundred solutions that are very close to POF'. Therefore, GD does not adhere to the property of monotony. Furthermore, GD does not adhere to the concept of weak relativity, because any subset of POF' will not necessarily have the best GD value when compared to POF^*s found by MOO algorithms.

It should be noted that GD is computationally expensive, especially for large or unlimited archives or when DMOOPs with a large number of objectives are used.

Inverted Generational Distance

To overcome non-adherence to the concept of monotony by GD, Sierra and Coello Coello [137] introduced the inverse generational distance (IGD). The mathematical definition of IGD is the same as GD in Equation (4.1), except for the way in which the distance is calculated:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n_{POF'}} d_i^2}}{n_{POF'}} \tag{4.2}$$

where $n_{POF'}$ is the number of solutions in POF' and d_i is the Euclidean distance in the objective space between solution i of POF' and the nearest member of POF^*.

IGD is compatible with relativity, since POF' obtains an IGD value of zero and POF^* will only receive an IGD value of zero if $POF^* = POF'$. Furthermore, IGD is compatible with monotony, because it will rate a POF^* with more non-dominated solutions that are close to POF as a better set than another POF^* that only has one solution that falls within POF'. However, IGD is computationally expensive to calculate for a larger POF' or a larger POF^*, since for each solution in POF', the distance between that solution and each of the solutions in POF^* has to be calculated. The usage of the distance function also requires scaling and normalisation of the objective function values, as is the case with GD.

Error Ratio

Van Veldhuizen [151] introduced the error ratio that measures the ratio of non-dominated solutions in POF^* that are elements of POF' to the non-dominated solutions in POF^* that are not elements of POF'. The error ratio is defined as

$$E = \frac{\sum_{i=1}^{n_{POF^*}} e_i}{n_{POF^*}} \tag{4.3}$$

where $e_i = 0$ if $x_i \in POF'$, $\forall x_i \in POF^*$ and $e_i = 1$ if $x_i \notin POF'$, $\forall x_i \in POF^*$. A small error ratio indicates a good performance.

If POF_A^* has two solutions with one solution in POF', $E = 0.5$. However, if POF_B^* has one hundred solutions with one solution in POF' and the other solutions very close to POF', $E = 0.99$. According to E, POF_A^* is a better set of solutions than POF_B^*. However, POF_B^* is more desirable. Therefore, E is only weakly compatible with O_C. E has weak relativity, because any subset of POF' will achieve the lowest E value, namely $E = 0$. It is not compatible with monotony, because if a non-dominated solution is added to POF^* that is not an element of POF', it will increase E.

The compatibility of the accuracy performance measures with the outperformance relations and the concepts of monotony and relativity is summarised in Table 4.1. In Tables 4.1 to 4.3 and Tables 4.4 to 4.11, M and R refer to the concepts of monotony and relativity respectively, C and W indicate that the performance measure is compatible or weakly compatible with the relation respectively and "–" indicates that the performance measure is neither compatible nor weakly compatible with the relation.

Table 4.1: Compatibility of accuracy performance measures

Performance Measure	O_W	O_S	O_C	M	R
GD	–	C	C	–	–
IGD	W	C	C	W	C
E	–	–	W	–	W

4.1.3 Diversity Performance Measures

This section discusses performance measures that are used to measure the diversity of the solutions contained in POF^*. Diversity can be measured either by measuring how

evenly the solutions are spread along POF^* or the extent of POF^*.

Number of Solutions

The easiest performance measure to calculate is the number of non-dominated solutions (NS) in POF^*. Van Veldhuizen [151] referred to this metric as the overall nondominated vector generation (ONVG). Even though this measure does not provide any information with regards to the quality of the solutions, it provides additional information when comparing the performance of various algorithms. For example, one algorithm may have a better GD value, but only half of the NS that have been found by the other algorithm.

NS is not weakly compatible with any of the outperformance relations. According to Knowles [99], weak compatability with O_W is necessary to ensure weak monotony. However, with NS this is not the case. Adding a non-dominated solution to POF^* increases, and thereby improves, NS. Therefore, NS is compatible with monotony. Furthermore, NS is weakly compatible with relativity only if the size of POF^* is smaller or equal to the size of POF'.

Spacing Metric of Schott

The *Spacing* metric, introduced by Schott [132], measures how evenly the points of POF^*, are distributed in the objective space. Spacing is calculated as:

$$\mathcal{S} = \sqrt{\frac{1}{n_{POF^*} - 1} \sum_{m=1}^{n_{POF^*}} (d_{avg} - d_m)^2}$$

with

$$d_m = min_{j=1,\ldots,n_{POF^*}; j \neq i} \left\{ \sum_{k=1}^{n_k} |f_k(\mathbf{x}) - f_{kj}(\mathbf{x})| \right\} \tag{4.4}$$

where d_m is the minimum value of the sum of the absolute difference in objective function values between the m-th solution in POF^* and any other solution in POF^*, d_{avg} is the average of all d_m values and n_k is the number of objective functions. If $\mathcal{S} = 0$, the non-dominated solutions of POF^* is uniformly spread or spaced [40]. However, this does not mean that the solutions are necessarily good, since they can be uniformly spaced in POF^*, but not necessarily uniformly spaced in POF [99, 55].

The spacing metric of Schott is not even weakly compatible with O_W [99]. Adding a non-dominated solution to POF^* will not necessarily decrease the value of S and POF'

does not necessarily have the lowest spacing metric value. Therefore, S does not adhere to the principles of either monotony or relativity.

It should be noted that this performance measure was designed to be used with other performance measures, has a low computational cost, and can provide useful information about the distribution of the found solutions [99]. Since the Euclidean distance is used in the calculation of the measure, the objectives should be normalised before calculating the measure.

Spacing Metric of Deb

S provides information with regards to how evenly the non-dominated solutions are spaced on POF^*. However, it does not provide any information with regards to the extent of spread of the solutions. To address this shortcoming, Deb [42] introduced a measure of spread, defined as:

$$\Delta = \frac{\sum_{k=1}^{n_k} d_k^e + \sum_{i=1}^{n_{POF^*}} |d_i - d_{avg}|}{\sum_{k=1}^{n_k} d_k^e + n_{POF^*} d_{avg}} \tag{4.5}$$

with d_i any distance measure between neighbouring solutions, d_{avg} is the mean of these distance measures and d_k^e is the distance between the extreme solutions of POF^* and POF'.

Similar to S, Δ is not compatible with O_W and does not adhere to monotony or relativity.

Maximum Spread

Zitzler [167] introduced a measure of maximum spread that measures the length of the diagonal of the hyperbox that is created by the extreme function values of the non-dominated set. The maximum spread is defined as:

$$MS = \sqrt{\sum_{k=1}^{n_k} \left(\overline{POF_i^*} - \underline{POF_i^*} \right)^2} \tag{4.6}$$

where $\overline{POF_k^*}$ and $\underline{POF_k^*}$ is the maximum and minimum value of the k-th objective in POF^* respectively. A high MS value indicates a good extend (or spread) of solutions.

This measure can be normalised in the following way [40]:

$$MS_{norm} = \sqrt{\frac{1}{n_k} \sum_{k=1}^{n_k} \left(\frac{\overline{POF_k^*} - \underline{POF_k^*}}{\overline{POF_k} - \underline{POF_k}} \right)^2} \tag{4.7}$$

If POF_A^* outperforms POF_B^* (weakly, strongly or completely), but the non-dominated solutions of POF_B^* have a larger extent than the non-dominated solutions of POF_B^*, then POF_A^* will obtain a higher MS value. Therefore, MS is not weakly compatible with any of the outperformance relations. Adding a non-dominated solution to POF^* will not necessarily lead to a higher MS value. Therefore, MS adheres to weak monotony. POF' will obtain the maximum MS value, but even a POF^* that only has two non-dominated solutions at the extreme points of POF' will also obtain the maximum MS value. Therefore, MS adheres to weak relativity.

C-Metric

The set coverage metric (C-metric) introduced by Zitzler [167] measures the proportion of solutions in set B that are weakly dominated by solutions in set A. The C-metric is defined as:

$$C(A,B) = \frac{|\{b \in B |\ \exists a \in A \colon a \preceq b\}|}{|B|} \tag{4.8}$$

If $C(A,B) = 1$, all solutions in set B are weakly dominated by set A and if $C(A,B) = 0$ no solution in set B is weakly dominated by set A. Let POF_A^* and POF_B^* be the approximated POFs found by two algorithms with $POF_A^* \subset POF_B^*$, $ND(POF_B^*) = POF_B^*$. Then $C(POF_A^*, POF_B^*) < 1$ and $C(POF_B^*, POF_A^*) = 1$. Therefore, POF_B^* outperforms POF_A^*. Under the assumption that, if $C(A,B) = 1$ and $C(B,A) < 1$ evaluates set A as being better than set B, the C-metric is compatable with O_W [99].

It is important to note that the domination operator is not a symmetric operator, i.e. $C(A,B)$ is not necessarily equal to $1 - C(B,A)$. Therefore, if many algorithms are compared against each other, this metric would have to be calculated twice for each possible combination of algorithms. However, it should be noted that the C-metric is cycle-inducing, in other words if more than two sets are compared, the sets may not be ordered and in these cases no conclusions can be made [99].

If POF is known, set A can be selected as the set of sampled points of the true POF, POF', and set B as the POF^* found by the algorithm. Then the C-metric can be calculated separately for each algorithm. Let POF_A^* and POF_B^* be the approximated POFs found by two algorithms as defined above and POF' a reference set with $ND(POF') = POF'$ and $POF_B^* \subseteq POF'$. Then, $C(POF', POF_A^*) = C(POF_B^*) = 1$

and $C(POF^*_A, POF') < C(POF^*_B, POF')$. In order to ensure compatibility with O_W, POF^*_A has to be evaluated by the C-metric as being worse than POF^*_B. Therefore, the following assumption should be made: if $C(R,E) = C(R,E) = 1$ and $C(E,R) < C(D,R)$, then E performs worse than D with regards to the C-metric, where D and E are two sets that are compared with one another using the reference set R [99]. Under this aforementioned assumption, the C-metric is compatible with O_W when a reference set is used.

The C-metric does not adhere to the concept of monotony, since POF^* can add a non-dominated solution that is weakly dominated by the set that POF^* is compared against. However, the C-metric is weakly compatible with relativity, since $C(POF^*, POF')$ cannot obtain a higher C-metric value than $C(POF', POF^*)$.

U-measure

Leung and Wang [103] introduced the U-measure to measure the diversity of the found non-dominated solutions. Let $R = r_k$ be the set of reference points, where r_k is the extreme point of objective k of the union of all non-dominated solution of all POF^*'s found by the algorithms for the same POF that are compared with one another. Let χ be the set $\{d_i\}$ and $\overline{\chi}$ the set of $\{d_j\}$, where d_i is the distance between two neighbouring solutions and d_j is the distance between a reference point, r_k, and its nearest neighbour. Let d^*_{avg} be the average of the distances in χ and let $\overline{\chi}^*$ be the set $\{d'_j | d'_j = d_j + d^*_{avg}\}$. Then, the U-measure is defined as:

$$U = \frac{1}{n_{POF^*}} \sum_{j=1}^{n_{POF^*}} \left| \frac{d'_j}{d_{ideal}} - 1 \right|$$

with

$$d_{ideal} = \sum_{j=1}^{n_{POF^*}} \frac{d'_j}{n_{POF^*}} \tag{4.9}$$

A smaller U-measure value indicates better uniformity of the non-dominated solutions of POF^*. Since distances are calculated in the U-measure, the objectives have to be normalised. Similar to S and Δ, the U-measure is not weakly compatible with any of the outperformance relations and does not adhere to monotony or relativity.

Table 4.2 summarises the compatibility of the diversity performance measures with

the outperformance relations and the concepts of monotony and relativity. In Tables 4.2 to 4.3, C* and W* indicate that the performance measure is either compatible or weakly compatible with the relation, but only under certain conditions.

Table 4.2: Compatibility of diversity performance measures

Performance Measure	O_W	O_S	O_C	M	R
NS	–	–	–	C	W*
S	–	–	–	–	–
Δ	–	–	–	–	–
MS	–	–	–	W	W
C	W*	C	C	–	W
U	–	–	–	–	–

4.1.4 Combined Performance Measures

This section discusses performance measures that measure the quality of the solutions of the found POF, by taking into account both the accuracy and diversity of the set of solutions.

Hypervolume

The hypervolume or S-metric (first referred to as "the size of the space covered") measures how much of the objective space is dominated by a non-dominated set [171, 172]. The definition of a dominated region and the traditional definition of the hypervolume are as follows:

Definition 4.10. Dominated Region: Let $\mathbf{f}_1 = \{f_{1_1}, f_{1_2}, \ldots, f_{1_k}\}$ be a solution in the objective space and $\mathbf{f}_{\text{ref}}$ a reference vector dominated by $\mathbf{f}_1$. Then the region that is dominated by $\mathbf{f}_1$ and bounded by $\mathbf{f}_{\text{ref}}$ is defined as the set,

$$R(\mathbf{f}_1, \mathbf{f}_{\text{ref}}) \triangleq \left\{ \mathbf{f}_r \,|\, \mathbf{f}_r \prec \mathbf{f}_{\text{ref}} \text{ and } \mathbf{f}_1 \prec \mathbf{f}_r, \;\; \mathbf{f}_r \in \mathbb{R}^K \right\} \tag{4.10}$$

Let A be a non-dominated set of vectors, $\mathbf{f}_i$, for $i = 1, \ldots, |A|$. Then the region dominated by A and bounded by the reference vector, $\mathbf{f}_{\text{ref}}$, is defined as the set:

$$R(A, \mathbf{f}_{\text{ref}}) \triangleq \bigcup_{i=1,\ldots,|A|} R(\mathbf{f}_i, \mathbf{f}_{\text{ref}}) \tag{4.11}$$

Definition 4.11. Hypervolume: The hypervolume (HV) or S-metric of set A with respect to the reference vector $\mathbf{f}_{\text{ref}}$ is the hyperarea or Lebesgue integral of the set $R(A, \mathbf{f}_{\text{ref}})$.

The reference vector can be any vector outside the feasible objective space, since this will result in a non-negative value for all possible non-dominated sets in the feasible objective space. Usually, the reference vector or reference point that is used in the HV calculation is the vector that consists of the worst value for each objective of the union of all non-dominated solutions of all POF^* that are compared against each other. It should be noted that the selected reference vector will affect the ordering of the non-dominated sets that are compared against each other, since all of the non-dominated sets use the same reference vector [99]. A high HV value indicates a good approximation set.

The HV is compatible with O_W if the upper boundary of the dominated region is set in such a way that all feasible non-dominated sets that are evaluated have a positive HV value. The HV is therefore compatible with the outperformance relations. The HV is weakly compatible with monotony and weakly compatible with relativity. It is scaling independent and no prior knowledge of the true POF is required. According to Zitzler *et al.* [168] the HV is the only performance measure in the literature that has the following two qualities:

- If an approximation set A dominates another set B the HV provides a strictly better value for A.
- If a set obtains the maximum possible HV value for a MOOP, it contains all Pareto-optimal objective values.

One flaw of the HV is that it is biased towards convex areas of the POF [168]. Furthermore, it is computationally expensive to calculate, with a computational cost of $O(n^{k+1})$ with k representing the number of objectives [99]. However, recent research developed algorithms that reduce the computational cost of the HV. For example, Fonseca *et al.* proposed an $O(|A|\log|A|)$ algorithm [62] and Beume and Rudolph proposed an algorithm with a complexity of $O(|A|^{k/2})$ [8], where A is the non-dominated set and k is the number of objectives.

Hypervolume Ratio

To overcome the bias of the HV towards convex regions of the POF, Van Veldhuizen [151] proposed the hypervolume ratio (HVR), defined as:

$$HVR = \frac{HV(POF^*)}{HV(POF)} \tag{4.12}$$

The HVR normalises the HV and, assuming that the maximum HV is obtained by the true POF, the value of the HVR will be between 0 and 1. A high HVR indicates a good approximated POF. It should be noted that, for the HVR calculation, the reference vector is selected as the worst objective values for each objective from the union of the non-dominated solutions of all POF^* that are compared against each other, as well as POF'.

Similar to the HV, the HVR is compatible with O_W if the upper boundary of the dominated region is set in such a way that all feasible non-dominated sets that are evaluated have a positive HV value. Therefore, the HVR is compatible with the outperformance relations. Furthermore, the HVR is weakly compatible with monotony and relativity.

ϵ-metric

Zitzler *et al.* [173] presented the ϵ-metric to compare approximated sets. It measures the factor by which an approximation set is worse than another approximation set with respect to all objectives, i.e. it provides the factor ϵ where for any solution in set B there is at least one solution in set A that is not worse by a factor of ϵ in all objectives. The ϵ-measure uses the concept of ϵ-dominance.

Using the definitions of objective vector domination and objective vector ϵ-domination (refer to Section 2.2.2), the ϵ-metric is defined as:

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall \mathbf{f}_2 \in B | \exists \mathbf{f}_1 \in A \colon f_1(\mathbf{x}_k) \prec_\epsilon f_2(\mathbf{x}_k)\} \tag{4.13}$$

The ϵ-metric is not weakly compatible with O_W, but is compatible with O_S and O_C. The ϵ-metric is not weakly compatible with monotonoy, but is weakly compatible with relativity.

The compatibility of the combined performance measures with the outperformance relations are summarised in Table 4.3.

The next section discusses how these MOO performance measures were adapted for DMOO. Performance measures developed specifically for DMOO are also discussed.

Table 4.3: Compatibility of combined performance measures

Performance Measure	O_W	O_S	O_C	M	R
HV	C*	C*	C*	W*	W*
HVR	C*	C*	C*	W*	W*
I_ϵ	–	C	C	–	W

4.2 Current Dynamic Multi-objective Optimisation Performance Measures

This section discusses performance measures that are currently being used to evaluate the performance of DMOO algorithms. Section 4.2.1 discusses performance measures that measure the accuracy of the found POF. Performance measures that are used to measure the diversity of the non-dominated solutions are discussed in Section 4.2.2. Section 4.2.3 discusses the measurement of an algorithm's robustness after an environment change occurs. Combined performance measures that measure accuracy and diversity of the non-dominated solutions are discussed in Section 4.2.4.

4.2.1 Accuracy Performance Measures

This section discusses performance measures that are used to measure the accuracy of a POF^*.

GD Measure

Mehnen *et al.* [117] used the GD metric to evaluate the performance of algorithms solving DMOOPs. They calculated the GD metric in decision space, since the DMOOPs that were used in the study had POSs that dynamically shifted over time, and named the performance measure G_τ. If GD is calculated in decision space, GD measures the distance of the approximated POS, POS^*, to the true POS, POS. Zhou *et al.* [166] used the GD metric (and the variance of GD) in objective space for DMOO, but referred to the performance measure as the distance indicator D. A number of other researchers have used GD to evaluate DMOO algorithms, as shown in Table 4.5. Goh and Tan [67] adapted GD for DMOO as follows:

$$VD = \frac{1}{\tau}\sum_{t=1}^{\tau} VD(\tau)$$

with

$$VD(\tau) = \frac{\sqrt{n_{POF^*}\sum_{i=1}^{n_{POF^*}} d_i^2\ (\tau \% \tau_t)}}{n_{POF^*}}$$

where τ is the current iteration number and τ_t is the frequency of change. The performance measure, referred to as variational distance (VD), is calculated in the decision space every iteration just before a change in the environment occurs.

Similar to GD, VD is not weakly compatible with O_W, but is compatible with O_S and O_C. It is not weakly compatible with monotony, but is weakly compatible with relativity. Since distance is used in the calculation of VD, the objectives have to be normalised.

When solving DMOOPs, similar to VD, the performance measure is calculated every iteration just before an environmental change occurs. Therefore, prior knowledge of when changes occur is required. However, if a performance measure is calculated while the algorithm is running (also referred to as online calculation), prior knowledge about changes in the environment is not required. In this case the performance measure can be calculated on the non-dominated solutions that were obtained at the iteration just before the change occurred. Furthermore, if the performance measure is calculated after the algorithm has completed its runs (also referred to as offline calculation), the algorithm can keep record of the iterations when changes occurred.

Success Ratio

Similar to the error ratio (refer to Section 4.1.2), Mehnen *et al.* [117] used the success ratio to quantify the ratio of the found solutions that are members of the true POF. The success ratio is defined as

$$SC_\tau = \frac{|\{\mathbf{x}|f(\mathbf{x}) \in POF'\}|}{n_{POF^*}} \tag{4.14}$$

where a high success ratio, SC_τ, indicates good performance.

If an algorithm finds many non-dominated solutions that are not pareto-optimal but very close to POF', the POF^* will obtain a lower SC_τ value than an algorithm that

finds only one pareto-optimal solution. Therefore, SC_τ is only weakly compatible with O_C and is not weakly compatible with either O_W or O_S.

If a non-dominated solution is added to POF^* that is not Pareto-optimal, the value of SC_τ decreases and therefore SC_τ is not compatible with monotony. Since POF' will obtain the same SC_τ value than subsets of POF', SC_τ is weakly compatible with relativity.

The compatibility of the accuracy performance measures with the outperformance relations and the concepts of monotony and relativity is summarised in Table 4.4.

Table 4.4: Compatibility of accuracy performance measures

Performance Measure	O_W	O_S	O_C	M	R
GD	–	C	C	–	W
SC_τ	–	–	W	–	W

In Tables 4.5 to 4.8, x indicates that the performance measure was used, x* indicates that the performance measure was calculated in decision space and x$^\triangleright$ indicates that the variance of the performance measure was used. The usage of the accuracy performance measures in the DMOO literature is summarised in Table 4.5. Table 4.5 shows that most researchers have used the GD or VD performance measure to quantify the accuracy of POF^*.

4.2.2 Diversity Performance Measures

This section discusses performance measures that are used to measure the diversity of the solutions contained in the approximated POF.

MS' measure

Goh and Tan [67] introduced an adapted version of MS (refer to Equation (4.7) in Section 4.1.3) to measure how well POF^* covers POF'. Contrary to MS, the adapted MS, MS', takes into account the proximity of POF^* to POF'. MS' is defined as:

$$MS' = \sqrt{\frac{1}{n_k}\sum_{k=1}^{n_k}\left[\frac{\min[\overline{POF_k^*},\overline{POF_k'}] - \max[\underline{POF_k^*},\underline{POF_k'}]}{\overline{POF_k'} - \underline{POF_k'}}\right]^2} \tag{4.15}$$

Table 4.5: Usage of DMOO accuracy performance measures

Year	Authors	Accuracy				
		GD	**IGD**	**VD**	**Acc**	**SC**
2004	Farina *et al.* [58]	x, x^*				
2005	Guan *et al.* [74]	x, $x^{\triangleright}$				
2006	Hatzakis and Wallace [76]	x, $x^{\triangleright}$, x^*, $x^{*\triangleright}$				
2006	Mehnen *et al.* [117]	x^*				x
2007	Cámara *et al.* [18] [15]				x	
2007	Li *et al.* [108]		x			
2007	Zhou *et al.* [166]	x, $x^{\triangleright}$				
2008	Isaacs *et al.* [87]	x, x^*				
2008	Tan and Goh [146]			x^*		
2009	Cámara *et al.* [16]				x	
2009	Chen *et al.* [23]	x				
2009	Wang and Li [155]		x, $x^{\triangleright}$			
2009	Goh and Tan [67] [66]			x^*		
2009	Isaacs *et al.* [88]	x, x^*				
2009	Salazar Lechuga [102]	x, $x^{\triangleright}$				
2009	Ray [127]	x, x^*				
2010	Cámara *et al.* [17] [138]				x	
2010	Koo *et al.* [100]			x^*		
2010	Wang and Li [156]		x			
2011	Helbig and Engelbrecht [78]			x	x	

Similar to MS, MS' is not weakly compatible with any of the outperformance relations. Adding a non-dominated solution to POF^* will not necessarily lead to a higher MS' value. Therefore, MS' adhere to weak monotony. POF' will obtain the maximum MS value, but even a POF^* that has only two non-dominated solutions at the extreme points of POF' will also obtain the maximum MS value. Therefore, MS' adheres to weak relativity.

PL **measure**

Since many diversity performance measures are based on the Euclidean distance and therefore do not take the shape of the POF into account, Mehnen *et al.* [117] introduced a performance measure, the PL measure, that is based on path lengths or path integrals. The length of the path between two solutions is defined as:

Definition 4.12. Length of Path between Two Solutions: Let γ be the path between two solutions in objective space, $\mathbf{a}$ and $\mathbf{b}$, that is differentiable in $[\mathbf{a}, \mathbf{b}]$. Then the length of a path between $[\mathbf{a}, \mathbf{b}]$ on γ is defined as:

$$L(\gamma, \mathbf{a}, \mathbf{b}) := \int_b^a |\dot{\gamma}| \, \mathrm{dt} = \int_b^a \sqrt{\dot{\gamma_1}^2 + \ldots + \dot{\gamma_m}^2} \, \mathrm{dt} \tag{4.16}$$

where $\dot{\gamma}$ is the derivative of γ and $|\dot{\gamma}|$ is the Euclidean norm of $\dot{\gamma}$.

The PL performance measure is the normalised product of the path between sorted neighbouring solutions on POF, defined as

$$\begin{aligned} PL_\tau &:= \frac{\ln\left(\prod_{\mathbf{f}(\mathbf{x}_i) \in POF} \zeta(\mathbf{x})\right)}{\ln e^{L_{POF}}} \\ &= \frac{\sum_{\mathbf{f}(\mathbf{x}_i) \in POF} \ln(\zeta(\mathbf{x}))}{L_{POF}} \end{aligned} \tag{4.17}$$

where $\zeta(\mathbf{x}_i) = L(\gamma, \mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_{i+1})) + 1$ and $\mathbf{f}$ represents the objective functions. For the calculation of PL, a solution is considered as being in POF if the solution is within an

Table 4.6: Usage of DMOO combined performance measures

Year	Authors	Quality					
		HV	**HVR**	**HVD**	**HV**$_{max}$	ϵ_{bin}	**OSPA**
2007	Deb *et al.* [46]		x				
2007	Cámara *et al.* [18] [15]	x					
2007	Li *et al.* [108]		x				
2007	Zheng [165]		x		x, x$^{\triangleright}$		
2007	Zhou *et al.* [166]			x			
2008	Greeff and Engelbrecht [72]	x					
2008	Talukder [144]	x					
2008	Talukder [96]		x				
2009	Avdagić *et al.* [2]	x					
2009	Cámara *et al.* [16]	x			x		
2010	Azevedo and Araújo [3]	x					
2010	Cámara *et al.* [17] [138]		x	x			
2010	Greeff and Engelbrecht [71]	x					
2010	Kim *et al.* [97]					x	
2010	Wang and Li [156]	x					
2011	Deb [41]		x				
2011	Helbig and Engelbrecht [78]		x				
2011	Tantar *et al.* [150]						x

ϵ-region near POF.

In order to calculate the PL performance measure, an analytic closed description of the true POF is required. However, according to Mehnen *et al.* the calculation of the PL measure is complicated when a DMOOP:

- has more than two objectives, or
- has a discontinuous POF.

In these situations Mehnen *et al.* [117] recommend the usage of S [132] (refer to Section 4.1.3).

PL is not weakly compatible with the outperformance relations. However, it is weakly compatible with monotony, since the value of PL increases when a new non-dominated solution that is within ϵ-distance of POF is added to POF^*.

Set Coverage Metric

Guan *et al.* [74] introduced a set coverage measure that is based on the S and D metrics introduced by Zitzler [167]. The HV of the objective space that is dominated by POF^* but not by POF', referred to as the D-metric, is defined as

$$D(POF^*, POF') = HV(POF^* + POF') - HV(POF') \tag{4.18}$$

The set coverage metric is then defined as

$$\eta = \frac{D(POF^*, POF')}{HV(POF')} + \frac{D(POF', POF^*)}{HV(POF')} \tag{4.19}$$

Therefore, the set coverage metric, η, is the normalised sum of the:

- HV of the objective space that is dominated by POF^* and not by POF', and
- HV of the objective space that is dominated by POF' and not by POF^*.

η is weakly compatible with O_W if the HV is weakly compatible with O_W. Therefore, η is weakly compatible with O_W if the reference vector is selected in such a manner that all feasible non-dominated sets that are evaluated have a positive HV value. If the reference vector is selected in this manner, η is compatible with all the outperformance relations. Furthermore, η is then weakly compatible with monotony and weakly compatible with relativity.

Pareto Front Extent

Zhang and Qian [164] introduced the coverage scope (CS) measure to quantify the average width or coverage of the non-dominated set. CS is calculated by averaging the

maximum distance between each solution in POF^* and the other solutions in POF^*. Therefore, CS is defined as

$$CS = \frac{1}{n_{POF^*}} \sum_{i=1}^{n_{POF^*}} \max\{\| \mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j) \|\} \tag{4.20}$$

with $\mathbf{x}_i, \mathbf{x}_j \in POF^*$, $i \geq 1$ and $j \leq n_{POF^*}$.

A higher CS value indicates a better performance. CS is similar to S [132] (refer to Section 4.1.3), but use the maximum distance where S uses the minimum distance between the non-dominated solutions in POF^*. Similar to S, CS is not weakly compatible with the outperformance relations. Furthermore, CS is not weakly compatible with monotony, since adding a non-dominated solution to POF^* can decrease the CS value. The CS value of POF' can be less than the CS value of POF^*. Therefore, CS is not weakly compatible with relativity.

A summary of the compatibility of diversity performance measures is shown in Table 4.8. In Table 4.8, C_o presents C_{orig} and C_m presents C_{mod}. Table 4.8 indicates that most researchers used S and MS to quantify the diversity of the non-dominated solutions in POF^*.

Table 4.7: Compatibility of diversity performance measures

Performance Measure	O_W	O_S	O_C	M	R
MS'	–	–	–	W	W
PL	–	–	–	W	–
η	C*	C*	C*	W*	W*
CS	–	–	–	–	–

4.2.3 Robustness Performance Measures

This section discusses performance measures that quantify the robustness of the algorithm, i.e. how well the algorithm recovers after an environment change occurs.

Stability Measure

The effect of the changes in the environment on the accuracy (*acc* defined in Equation 4.23) of the algorithm can be quantified by the measure of stability that was introduced by Weicker [157] for DSOO and adapted for DMOO by Cámara *et al.* [138].

Table 4.8: Usage of DMOO diversity performance measures

Year	Authors	Diversity/Spread											
		NS	**C**$_o$	**C**$_m$	**PFE**	**Spac** (Schott)	**U**	**PL**	**MS**	**Spread** (Deb)	**Entropy**	**Accum**	**N**$_E$
2005	Guan *et al.* [74]	x		x									x
2006	Mehnen *et al.* [117]					x		x					
2006	Zheng *et al.* [160]									x			
2008	Greeff and Engelbrecht [72]	x				x							
2008	Tan and Goh [146]								x				
2008	Wang and Dang [153]		x				x						
2009	Avdagić *et al.* [2]		x										
2009	Goh and Tan [67] [66]								x				
2009	Chen *et al.* [23]										x		
2010	Azevedo and Araújo [3]										x	x	
2010	Greeff and Engelbrecht [71]					x							
2010	Koo *et al.* [100]								x				
2010	Liu [110]						x						
2011	Helbig and Engelbrecht [78]					x			x				
2011	Zang *et al.* [164]		x		x	x							

Stability is defined as

$$stab(t) = \max\{0, acc(t-1) - acc(t)\} \tag{4.21}$$

where a low *stab* value indicates good performance.

The compatibility of *stab* depends on the definition of *acc*. If *acc* as defined in Equation (4.23) is used, *stab* is compatible with O_W if *acc* is compatible with O_W. Under these conditions, *stab* is compatible with the outperformance relations and weakly compatible with monotony and relativity.

Reactivity Measure

Cámara *et al.* [138] presented a measure of reactivity based on the reactivity performance measure introduced by Weicker [157] for DSOO. Reactivity measures how long it takes for an algorithm to recover after a change in the environment, by determining how long it takes for an algorithm to reach a specified accuracy threshold. The reactivity performance measure is defined as

$$react(t, \epsilon) = \min\left\{t' - t | t < t' < \tau_{max}, t' \in \mathbb{N}, \quad \frac{acc(t')}{acc(t)} \geq (1 - \epsilon)\right\} \tag{4.22}$$

where τ_{max} is the maximum number of iterations or generations.

Similar to *stab*, *react* is weakly compatible with O_W if *acc* is weakly compatible with O_W. *react*'s compatibility with monotony and relativity also depends on *acc*'s compatibility with monotony and relativity.

The compatibility with the outperformance relations by the robustness performance measures is summarised in Table 4.9.

Table 4.9: Compatibility of robustness performance measures

Performance Measure	$\mathbf{O_W}$	$\mathbf{O_S}$	$\mathbf{O_C}$	**M**	**R**
stab	C*	C*	C*	W*	W*
react	C*	C*	C*	W*	W*

Table 4.10 summarises the usage of performance measures that quantifies robustness in the DMOO literature.

Table 4.10: Usage of DMOO robustness performance measures

Year	Authors	Robustness	
		Stb	React
2007	Cámara *et al.* [18] [15]	x	x
2009	Cámara *et al.* [16]	x	x
2010	Cámara *et al.* [17] [138]	x	x
2011	Helbig and Engelbrecht [78]	x	

4.2.4 Combined Performance Measures

This section discusses performance measures used to quantify the overall quality of the found POF, i.e. they do not measure only one aspect such as convergence to the true POF or the diversity of the solutions.

Accuracy Measure

A measure of accuracy introduced by Weicker for DSOO [157] was adapted by Cámara *et al.* [138] for DMOO. This measure quantifies the quality of the solutions as a relation between the HV of POF^* and the maximum HV that has been found so far. The accuracy measure is defined as

$$acc(t) = \frac{HV(POF^*(t))}{HV_{max}(POF^*(t))} \tag{4.23}$$

The accuracy measure, *acc*, is compatible with O_W if the upper boundary of the dominated region is set in such a way that all feasible non-dominated sets that are evaluated have a positive HV value (refer to Section 4.1.4). Under these conditions, *acc* is compatible with the outperformance relations and weakly compatible with monotony and relativity.

Hypervolume Difference

Zhou *et al.* [166] suggested to use the hypervolume distance (HVD) to measure the quality of the found POF. HVD is defined as

$$HVD = HV(POF') - HV(POF^*) \tag{4.24}$$

However, when the true POF is unknown, the HVD cannot be used. Zheng used the maximum HV to measure the quality of the found POF [165].

Cámara *et al.* [17] extended the definition of their accuracy measure (Equation (4.23))

to use the HVD when the true POF is known. The alternative accuracy measure is defined as

$$acc_{alt}(t) = |HV(POF'(t)) - HV(POF^*(t))| \tag{4.25}$$

where $acc_{alt}(t)$ is the absolute HVD at time t. The absolute values ensure that $acc_{alt}(t) \geq 0$, even if $HV(POF^*) > HV(POF')$. HVD is compatible with the outperformance relations if HV is compatible with the outperformance relations.

Optimal Subpattern Assignment Measure

Recently Tantar *et al.* [150] introduced performance measures that are based on performance measures used in quantifying the tracking quality in multi-object tracking problems. The performance measures are developed based on the optimal subpattern assignment (OSPA) measure that can be used to compare sets with different cardinality [133].

Let $P = (F, X, N)$ define a DMOOP with F and X representing a set of objective functions and a set of decision variables respectively. N represents the neighbourhood function described by a ball of center c and radius r, defined as

$$N(c, r) = \{\mathbf{x} \in X |\ d(\mathbf{x}, c) < r \text{ and } \exists h | \mathbf{x}hc\} \tag{4.26}$$

where d is the distance between a solution, $\mathbf{x}$, and the center point of the neighbourhood, c, and $\exists h | \mathbf{x}hc$ indicates that the neighbourhood can be reached through a transformation h.

Let A and B respresent two approximated POFs with cardinality of m and n respectively. Then the following two performance measures are defined:

$$M_{loc}(X, Y) = \left(\frac{1}{n_{POF_B^*}} \min_{j \in P} \left\{ \sum_{i=1}^{n_{POF_A^*}} d(x_i, y_{j(i)})^p \right\} \right)^{\frac{1}{p}} \tag{4.27}$$

where $d(\mathbf{x}, \mathbf{y}) = min\{c, d(\mathbf{x}, \mathbf{y})\}$ is the minimum distance between two solutions that are cut off by c. When comparing A and B, the solutions from B that are in the neighbourhood of a given solution from A are determined by considering all permutations of solutions from B, referred to as the set P. M_{loc} quantifies the quality of the coverage of A as compared to B. A drawback of this performance measure is its computational cost, because of the calculation of permutations for each solution under consideration.

The other performance measure is defined as

$$M_{card}(X,Y) = \left(\frac{c^p(n_{POF_B^*} - n_{POF_A^*})}{n_{POF_B^*}} \right)^{\frac{1}{p}} \tag{4.28}$$

M_{card} is a cardinality penalty function that is used when $|A| \neq |B|$, and is zero if the two sets have the same cardinality. M_{card} measures the influence of the cardinality difference on the overall quality of the larger set, with the cut-off term as the error quantification factor.

The OSPA metric is then defined as:

$$OSPA(X,Y) = M_{loc}(X,Y) + M_{card}(X,Y) \tag{4.29}$$

$OSPA$ is not weakly compatible with the outperformance relations. However, $OSPA$ is weakly compatible with monotony.

Table 4.11 summarises the compatibility with the outperformance relations by the combined performance measures.

Table 4.11: Compatibility of combined performance measures

Performance Measure	O_W	O_S	O_C	M	R
HVD	C*	C*	C*	W*	W*
acc_{alt}	C*	C*	C*	W*	W*
$OSPA$	–	–	–	W	–

When algorithms solve DMOOPs, five major issues should be taken into consideration when selecting performance measures to quantify the performance of the algorithms, namely: algorithms losing track of the changing POF, the effect of outlier solutions in the found POF, boundary constraint violations, calculating the performance measures in either the objective or decision space, and comparing the performance of the various algorithms. These issues are discussed in the next section.

4.3 Issues with Current Dynamic Multi-objective Optimisation Performance Measures

Section 4.2 discussed a number of performance measures that have been used to quantify the performance of algorithms on DMOOPs. Even though these measures have been

used in a number of articles, they suffer from a number of problems mostly related to aspects of dynamic environments. These problems make general applicability of these performance measures to all DMOOPs not possible.

Section 4.3.1 discusses misleading results that can occur when algorithms lose track of the changing POF. The effect of outlier solutions in POF^* on the quantification of an algorithm's performance is discussed in Section 4.3.2. Section 4.3.3 discusses the effect of boundary constraints violations on the performance of DMOO algorithms. Furthermore, performance measures can be calculated in either the objective or decision space as discussed in Section 4.3.4. Finally, Section 4.3.5 discusses issues when comparing the performance of the various algorithms.

4.3.1 Losing Track of the Changing Pareto Optimal Front

When a DMOO algorithm loses track of the changing POF, and POF changes over time in such a way that its HV value decreases, many of the current performance measures will give misleading results. Figure 4.1 illustrates example POFs where the POF changes over time in such a way that, if the HV is calculated with the reference vector being the worst values of each objective, the HV will decrease over time. A decrease in the HV will occur if for each example the POF changes from convex to concave. Figure 4.1 illustrates three such POFs. In Figure 4.1, the first POF is represented by the bottom line and the last POF by the top line.

The problem of losing track of the POF was first observed by Helbig and Engelbrecht [77], where five algorithms were used to solve the FDA2 DMOOP. These algorithms included a dynamic VEPSO (DVEPSO) which uses clamping to manage boundary constraint violations (DVEPSO-A) [77], DVEPSO that uses per element re-initialisation to manage boundary constraint violations (DVEPSO-B) [77], NSGA-II where a percentage of individuals are randomly selected and replaced with newly created individuals if an environment change occurs (DNSGA-II-A) [46], NSGA-II where, after an environment change, a percentage of individuals are randomly selected and replaced with individuals that are mutated from existing individuals (DNSGA-II-B) [46], and dCOEA [67]. Figure 4.2 illustrates example POFs obtained by these algorithms in comparison with POF (Figure 4.2(f)). It is clear from these figures that DNSGA-II-A, DNSGA-II-B

and dCOEA lost track of the changing POF, with the DVEPSO algorithms being more successful in tracking the POF. It is therefore expected that the values of the performance measures should be better for the DVEPSO algorithms than for the evolutionary algorithms.

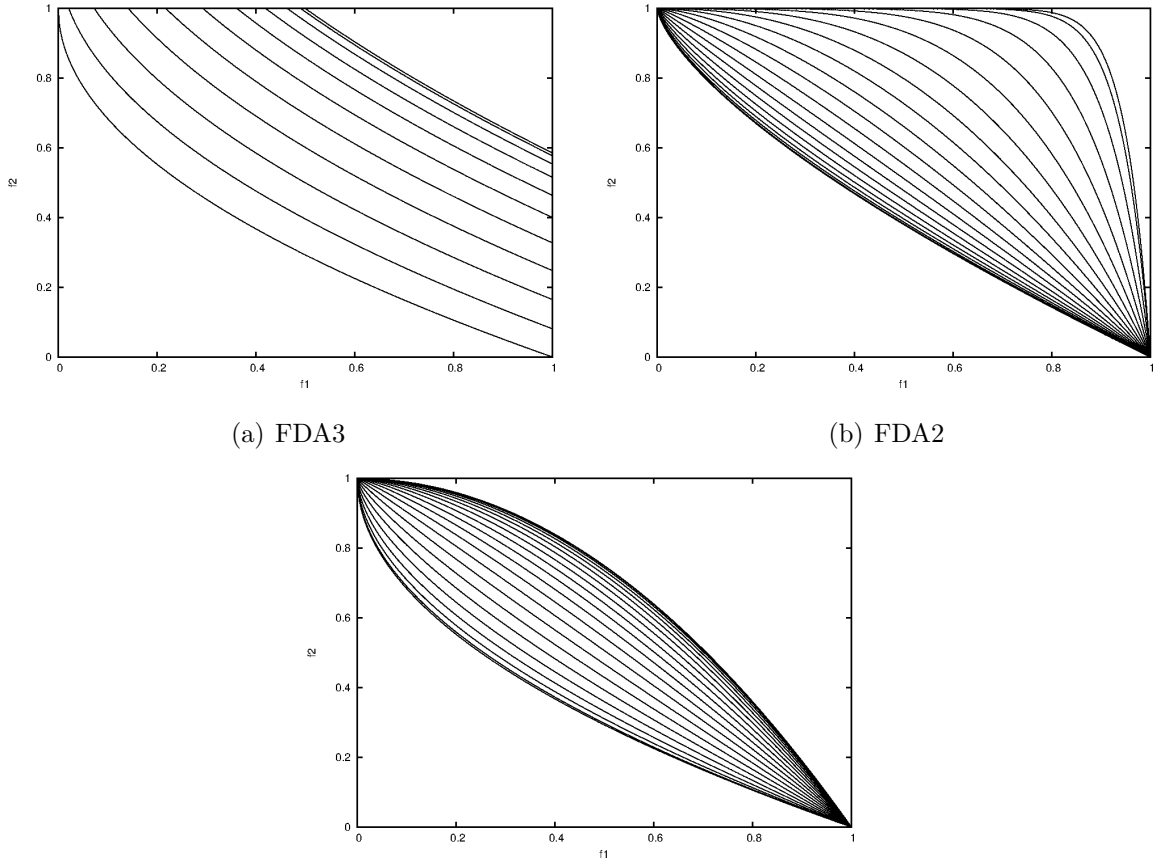
The performance measure values of these algorithms for a change frequency of ten are presented in Table 4.12. In Table 4.12 NS refers to the number of non-dominated solutions found, S refers to the spacing measure defined by Schott [132], HVR refers to the HV ratio [108], *Acc* and *stab* refer to measures of accuracy and stability presented by Cámara *et al.* [18], and VD and MS refer to the adapted generational distance and maximum spread performance measures for dynamic environments proposed by Goh and Tan [67].

As shown in Table 4.12, performance measures VD and MS indicate the DVEPSO algorithms to be better than the evolutionary algorithms. However, the measures that make use of the HV, namely HVR, *Acc* and *stab*, rank the evolutionary algorithms as being better than the DVEPSO algorithms. This occurs since the HV value of POF decreases over time and therefore from the time where an algoritm loses track of the changing POF, its HV value is higher than that of POF and therefore higher than that of algorithms that are tracking the changing POF. Since the HV value of POF decreases over time, HVR (which divides the HV of POF^* by the HV of POF) still does not address this problem.

Tables 4.5 and 4.6 show that the following papers used the HV or HVR without using any accuracy measure that requires knowledge of the true POF: [2, 3, 15, 18, 16, 17, 41, 46, 72, 71, 96, 138, 144, 165]. Therefore, if any of the algorithms that were evaluated in these studies lost track of the changing POF, the performance measure values that were obtained may be misleading.

The issue of an algorithm losing track of the changing POF is unique to DMOO. In order to overcome this problem, acc_{alt} proposed by Cámara *et al.* (refer to Equation (4.25) in Section 4.2.4) should be used when the POF is known. Furthermore, if acc_{alt} is used for *acc*, *stab* will also be reliable even if an algorithm loses track of POF.

If POF is unknown, as is the case with most real-world problems, the deviation of the performance measures that use the HV measure should also be calculated. If



(a) FDA3

(b) FDA2

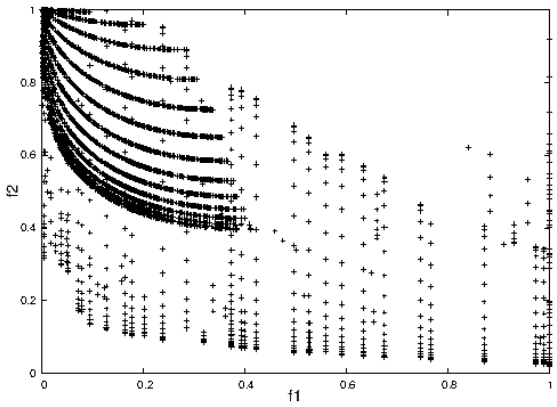
(c) dMOP2

Figure 4.1: Examples of DMOOPs where the HV value of POF decreases over time

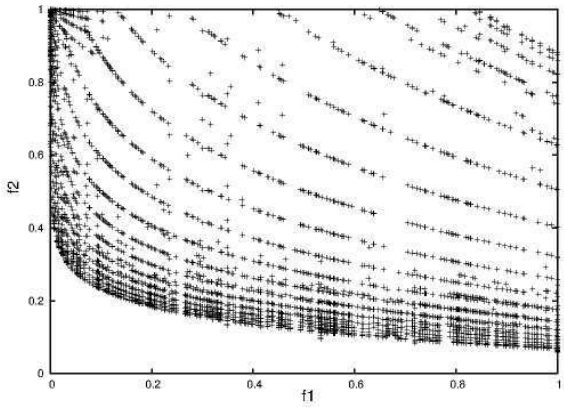
Table 4.12: Performance Measure Values for FDA2

τ_t	Algorithm	NS	S	HVR	Acc	Stab	VD	MS	R
10	DVEPSO-A	**73.4**	0.00118	0.99533	0.97848	0.00049	0.45824	**0.90878**	4
10	DVEPSO-B	63	0.00391	0.99905	0.98157	0.00029	**0.43234**	0.88916	3
10	DNSGAII-A	39.4	0.00044	1.0044	0.98681	$9.565\text{x}10^{-06}$	0.71581	0.77096	2
10	DNSGAII-B	39.6	**0.00042**	**1.00441**	**0.98683**	$\mathbf{9.206\text{x}10^{-06}}$	0.71681	0.77866	**1**
10	dCOEA	38.4	0.00051	1.00209	0.98454	0.00122	0.70453	0.61923	5

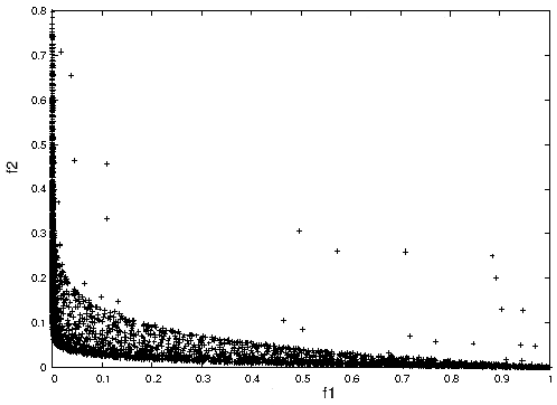
the performance measure's deviation varies much more for certain algorithms, it may indicate that one or more of the algorithms have lost track of the changing POF and



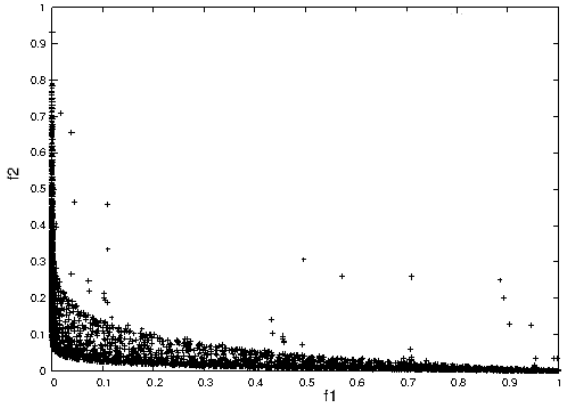
(a) POF^* found by DVEPSO-A



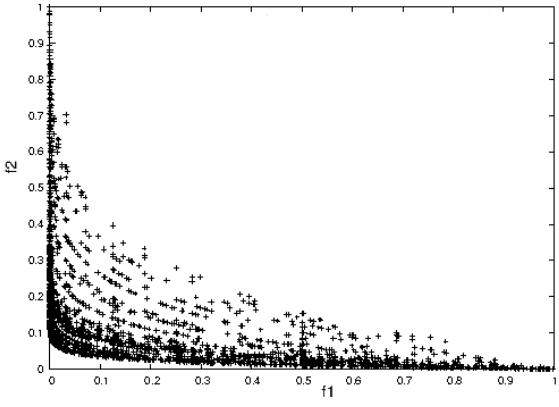
(b) POF^* found by DVEPSO-B



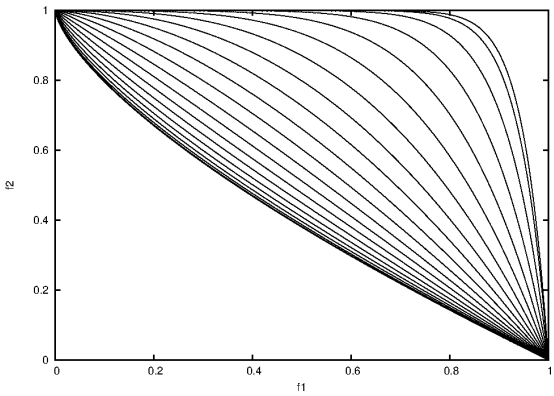
(c) POF^* found by DNSGA-II-A



(d) POF^* found by DNSGA-II-B



(e) POF^* found by dCOEA



(f) POF of FDA2

Figure 4.2: POF and POF^* found by various algorithms for FDA2 with $n_t = 10$, $tau_t = 10$ and 1000 iterations

that the performance measure can not reliably be used to compare the performance of the different algorithms. Therefore, the graphs of POF^*s should be plotted and checked to determine whether any algorithm has lost track of the changing POF.

Even though various performance measures were used, the misleading performance measures can play a large enough role to influence the overall ranking of the algorithms. This is shown in Table 4.12. Even though the DVEPSO algorithms ranked the highest with regards to NS, VD and MS, the measures that make use of the HV value affected the ranking in such a way that the DVEPSO algorithms ranked as number 3 and 4 respectively and were outranked by the DNSGA-II algorithms - portraying an incorrect ordering.

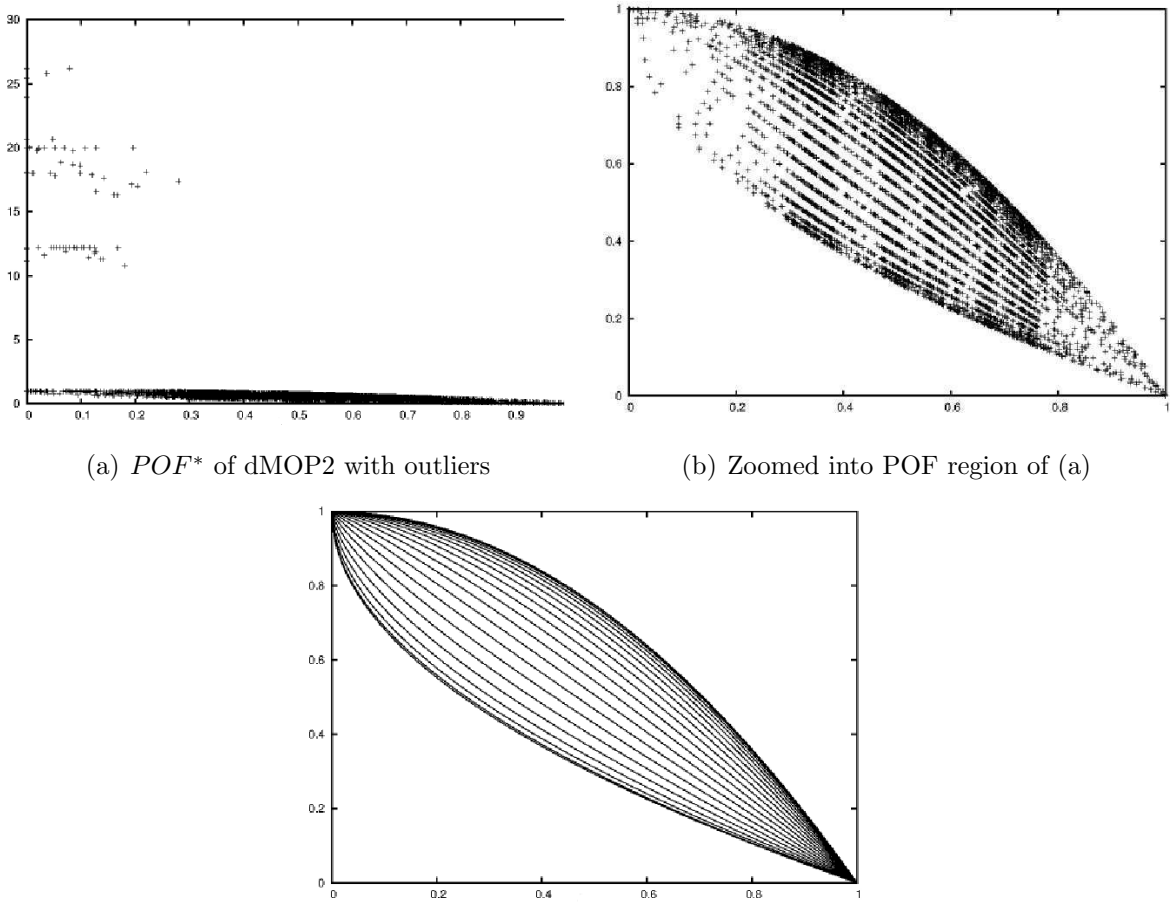
It should be noted that the stability measure, *stab*, proposed by Cámara *et al.* [18] measures the change in the values of the accuracy measure at two consecutive time steps (refer to Section 4.2.3). Under normal circumstances a low *stab* value indicates that the performance of the algorithm is not severely affected by the change in the environment. However, in situations where one or more algorithm(s) lost track of the changing POF, the lowest *stab* value will be obtained by the algorithms that lost track of the changing POF. Therefore, the results obtained with the *stab* performance measure will be misleading. Table 4.12 shows that the NSGA-II algorithms obtained a better *stab* value than the DVEPSO algorithms. Clearly, as indicated by the POFs shown in Figure 4.2, this is not the case.

4.3.2 Outliers in the Pareto Optimal Front

When algorithms solve DMOOPs and the environment changes frequently, the POF^* that has been found by the algorithm for a specific time step may contain outliers. This occurs because the algorithm found non-dominated solutions that are further away from the true POF within the number of iterations or generations available to the algorithm to solve the specific POF. In the time available before the environment changes, the algorithm did not find any solutions closer to the true POF that dominated these outlier solutions. Figure 4.3 illustrates an example POF^* that contains outliers.

Outliers will skew results obtained using:

- distance-based performance measures, such as GD, VD, PL, CS and M_{loc},



(a) POF^* of dMOP2 with outliers

(b) Zoomed into POF region of (a)

(c) POF of dMOP2

Figure 4.3: Example of a POF^* that contains outlier solutions.

- performance measures that measure the spread of the solutions, such as MS, and
- the HV performance measure.

The influence of outlier solutions on the calculation of GD and VD is illustrated in Table 4.13. Due to the large distance between the outliers and POF as shown in Figures 4.3 and 4.4, the resulting GD and VD is much larger with the outliers present compared to when the outliers are not present. However, it should be noted that the severity of the influence of outliers on distance calculations depends on the number of outliers and their distance from POF.

Furthermore, when a performance measure, such as MS of Cámara *et al.* [18], measures the extend or spread of the approximated POF, these outlier solutions may cause the performance measure to be misleading with regards to the performance of the algorithm. In Figure 4.4, the outlier solutions' f_1 and f_2 values will become the $\overline{PF_i^*}$ and $\underline{PF_i^*}$ in the f_2 and f_1 objective in Equation (4.7) respectively. In Figure 4.4, POF^* only contains non-dominated solutions with f_1 values in the range of $[0.2, 0.7]$ and f_2 values in the range of $[0, 0.5]$ without the outlier solutions. However, with the outlier solutions the f_1 values will be calculated as being in the range of $[0, 1.0]$ and f_2 values in the range of $[0.2, 3]$. This will result in the maximum MS value and will not give a true reflection of the diversity of solutions that has been found by the algorithm. The influence of the outliers on the value of MS is shown in Table 4.13.

When solving DMOOPs, many researchers use the HV performance measure, especially when POF is unknown. When comparing various algorithms' POF^*s, the same reference vector is used. HV values that are calculated with different reference vectors cannot be compared against each other. Furthermore, outlier solutions influence the reference vector values that are used to calculate the HV. Typically, the reference vector is chosen as the worst values obtained for each objective. Therefore, for POF^* in Figure 4.4 the reference vector for the HV is $[1.1, 3.1]$ and $[1.1, 1.1]$ with and without the outlier values respectively. This results in much larger HV values when outliers are present, as shown in Table 4.14. From Table 4.14 it is clear that HVR and acc_{alt} provide a more accurate representation of the performance of the algorithm, resulting in a better HVR value without outliers than with the outliers. However, when the HV is used, the POF^* with outliers obtain a better HV value than the POF^* without the outliers. Therefore, if POF is unknown and the HV is used, outlier solutions may lead to misleading results and algorithms being ranked incorrectly.

One approach to manage outliers in POF^* is to remove the outliers from POF^*. However, no consensus exists on the approach that should be followed to decide which non-dominated solutions in POF^* should be classified as outliers.

It should be noted that, as the number of objectives increases, more outlier solutions may become present in POF^*. This is the case, since as the number of objectives increases, more solutions that are found by the algorithm will be non-dominated with

regards to the other solutions in POF^*. Furthermore, outliers in POF^* will cause the same problems when solving static MOOPs. However, since algorithms generally have longer time to converge towards POF with static MOOPs than with DMOOPs where the environment changes, the possibility of the occurrence of outliers increases when solving DMOOPs.

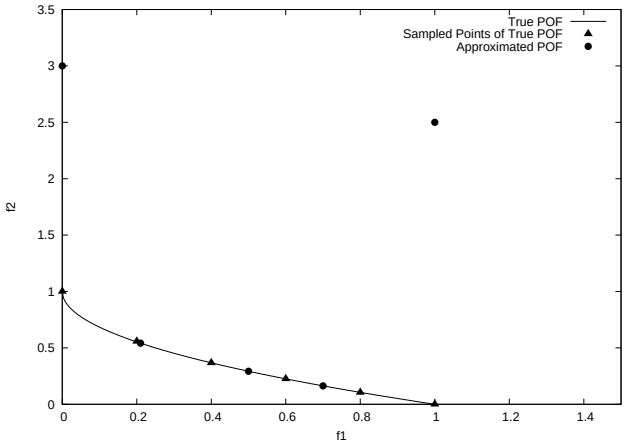


Figure 4.4: POF^* of FDA1 with outlier solutions

Table 4.13: GD, VD and MS values for FDA1

Outliers	**GD**	**VD**	**MS**
Yes	2.05565	4.596574	0.91833
No	0.00942	0.016311	0.4342

Table 4.14: HV, HVR and HVD values for FDA1

Outliers	**HV**	**HVR**	**acc**$_{alt}$
Yes	2.49898	0.84461	0.45974
No	0.69798	0.91994	0.06074

4.3.3 Boundary Constraint Violations

The candidate solutions of certain computational intelligence algorithms tend to move outside the boundary constraints of an optimisation problem while searching for solutions. For example, it has been shown theoretically that most particles in a PSO algorithm [94] leave the bounds within the first few iterations [55, 63]. If a particle finds a better solution outside the bounds, its personal best position is set to this new position. Should this position be better than the current neighbourhood or global best, other particles are also pulled outside of the bounds. Consequently, particles may converge on a solution outside the bounds of the search space. This behaviour of particles was empirically analyzed by Engelbrecht [56].

If a GA [82] uses blending cross-over, such as parent-centric cross-over [44], offspring may be generated outside the boundaries of the search space due to the asymptotic tails of the distributions of the stochastic component of the blending process.

Most evolutionary programming [59] algorithms sample mutational step sizes from zero-mean distributions with tails that asymptotically approach infinity and negative infinity. Consequently, large mutational step sizes can potentially be added to parent individuals, moving offspring outside of the bounds. If such offspring have better fitness than parent individuals, these offspring survive to the next generation with a chance of obtaining a solution that does not lie within the bounds of the search space.

With differential evolution's [142] mutation operator, a weighted difference of two vectors are added to the parameter vector. If this weighted difference is large, it may cause the trial vector to move outside the boundary constraints of the optimisation problem.

Most unconstrained DMOOPs have boundary constraints that limit the search space. However, if an algorithm does not manage boundary constraint violations, infeasible solutions may be added to POF^*. These infeasible solutions may dominate feasible solutions in POF^*, which will cause the feasible solutions to be removed from POF^*. Furthermore, the infeasible solutions may cause misleading results with regards to an algorithm's performance.

Figure 4.5(a) illustrates a POF^* that was found by DVEPSO that did not manage boundary constraint violations (DVEPSO$_u$) when solving dMOP2, DVEPSO that

manages boundary constraint violations (DVEPSO$_c$), and the true POF (POF). From Figure 4.5 it is clear that POF^* of DVEPSO$_u$ has a larger HV value than both POF^* of DVEPSO$_c$ (refer to Figure 4.5(b)) and POF (refer to Figure 4.5(c)). This is confirmed in Table 4.15. This incorrectly indicates that the POF^* that contains solutions that are outside the bounds of the search space to be better. Therefore, when comparing various algorithms with one another, it is important to check that all algorithms manage boundary contraint violations to ensure a fair comparison. It should be noted that the issue of boundary constraint violations is applicable to both SMOO and DMOO.

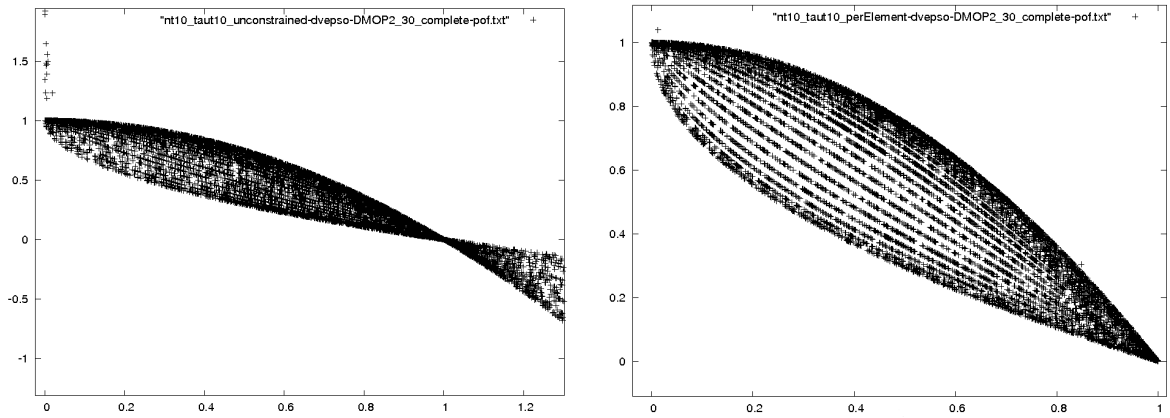
Table 4.15: HVR values for dMOP2

Algorithm	**HVR**
DVEPSO$_u$	1.00181
DVEPSO$_c$	0.99978

4.3.4 Objective Space versus Decision Space

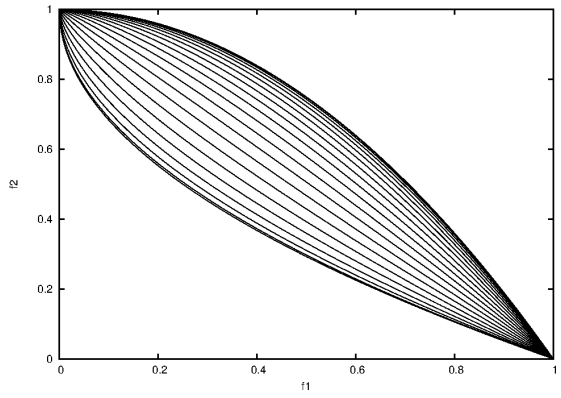
Accuracy measures, such as VD or GD, can be calculated with respect to either the decision or the objective space. Using objective space, VD measures the distance between the non-dominated solutions of POF^* and POF'. Therefore, VD measures the closeness of POF^* to POF. Since one of the goals of solving a DMOOP is to track the changing POF, the accuracy should be measured in the objective space. If the VD measure is calculated in the decision space, the distance between POS^* and POS is calculated. Calculating the VD measure in the decision space may be useful to determine how close POS^* is from POS. However, if for a DMOOP a small change in the POS causes a big change in the POF, it may occur that even though the algorithm's POS^* is very close to POS, POF^* is quite far from POF. This is illustrated with an example DMOOP defined as:

$$\mathsf{DMOOP}_1 = \begin{cases} Minimize : f(\mathsf{x},t) = (f_1(\mathbf{x_I}), g(\mathbf{x_{II}}) \cdot \\ \qquad\qquad h\left(\mathbf{x_{III}}, f_1(\mathbf{x_I}), g\left(\mathbf{x_{II}}\right), t\right)) \\ f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 - \sum_{x_i \in \mathbf{x_{II}}} \sqrt{x_i - G(t)} - \\ \qquad\qquad \sum_{x_j \in \mathbf{x_{III}}} \left(x_j - G(t)\right)^2 \\ h(\mathbf{x_{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ where : \\ G(t) = \sin(0.5\pi t), \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ H(t) = 1.5 + G(t) \\ \mathbf{x_I} \in [0,1]; \;\; \mathbf{x_{II}}, \mathbf{x_{III}} \in [-1,1] \end{cases} \tag{4.30}$$



(a) POF^* of dMOP2 with feasible and infeasible solutions

(b) POF^* of dMOP2 with only feasible solutions



(c) POF of dMOP2

Figure 4.5: Example of a POF^* that contains infeasible solutions due to boundary constraint violations

For DMOOP$_1$, the POS and POF are:

$$POS: x_i = G(t), \forall x_i \in \mathbf{x_{II}}, \mathbf{x_{III}}$$
$$POF: y = 1 - f_1^{H(t)}$$

Let $\mathbf{x_{II}} = \{x_1, x_2, x_3\}$, $\mathbf{x_{III}} = \{x_4, x_5, x_6\}$, $t = 0.1$, $G(t) = 0.156$, $\mathbf{x}_1 = \{0.14, 0.16, 0.16, 0.16, 0.16, 0.16\}$ and $\mathbf{x}_2 = \{0.16, 0.16, 0.16, 0.16, 0.14, 0.16\}$. Then, measuring the distance between the solution and the true POS (i.e. in decision space), $d(\mathbf{x})_{dec}$, $\mathbf{x}_1$ and $\mathbf{x}_2$ obtains the same d_{dec} value. However, $\mathbf{x}_1$ and $\mathbf{x}_2$ produces the following gh values respectively: 0.937512 and 0.93183. The true POF value for $\mathbf{x}_1$ and $\mathbf{x}_2$ are 0.961453 and 0.951914 respectively. The difference between the gh values found by $\mathbf{x}_1$ and $\mathbf{x}_2$ and the true POF values, d_{obj}, are 0.023941 and 0.020084 respectively. Therefore, even though in the decision space the difference between $\mathbf{x}_1$ and $\mathbf{x}_2$ and the true POS produces the same d_{dec} value, their difference in objective space, d_{obj}, is different, with $\mathbf{x}_2$ being closer to the true POF than $\mathbf{x}_1$.

Measuring VD in the decision space will indicate how close the decision variable values are from POS. However, the VD value measured in decision space will not give a true reflection of the accuracy of POF^* with regards to POF. Therefore, measuring VD in decision space to determine the accuracy of the algorithm's found solutions only makes sense for DMOOPs of Type I where the POS changes over time, but the POF remains static. However, for DMOOPs of Type II and III, measuring VD in the decision space will not provide any information with regards to how well the algorithm has tracked the changing POF and therefore for these type of DMOOPs VD should be measured in objective space.

The following papers measured either GD or VD in only the decision space: [67, 66, 100, 117, 146]. In [146] only FDA1, which is a Type I DMOOP, was used and therefore measuring in the decision variable space makes sense. In [100], three DMOOPs of Type I (FDA1, DIMP1 and DIMP2) were used and one DMOOP of Type II (FDA3). For the Type II DMOOP, calculating in the decision space will only provide information with regards to tracking of the changing POS, but not with regards to tracking the changing POF. In [67, 66, 117], DMOOPs of Types I, II and III were used. For the DMOOP of Type III, measuring in the decision space only indicates whether POS^* remains close to

POS, which remains static over time. Therefore, it provides no information with regards to how well the algorithm has tracked the changing POF. The issue of calculating performance measures in either decision or objective space is unique to DMOO, since with SMOO both the POS and POF remain static.

4.3.5 Comparing Performance of Algorithms

When the performance of various algorithms are compared against one another, typically various performance measures are used. Some algorithms will perform very well with regards to certain performance measures and not so well with regards to the other performance measures. Therefore, typically each algorithm will be ranked according to its performance with regards to each performance measure. Then, for each algorithm its average rank is calculated. These averaged ranks are then used to determine how well each algorithm performed with regards to the other algorithms. However, the performance measures that are used for comparing various algorithms should be chosen with care. If the wrong performance measures are selected, it may lead to incorrect ordering as discussed in Section 4.3.1 and illustrated in Table 4.12 and [77]. Therefore, if POF is known, the usage of acc_{alt} is suggested. However, more research is required to determine the best performance measure(s) for cases where POF is unknown.

4.4 Summary

This chapter provided an analysis of performance measures for DMOO. Concepts of outperformance relations, compatibility, monotony and relativity were introduced that have been used to evaluate static MOO performance measures. Performance measures that were used for MOO to measure convergence to the true POF, diversity of the solutions, as well as overall quality of the approximated POF were discussed. Adaptation of the MOO performance measures for DMOO was presented, as well as performance measures that have been introduced specifically for DMOO. Furthermore, problems with current DMOO performance measures were discussed, indicating that algorithms that lose track of the POF, outliers in the found POF, and violation of the boundary constraints can produce misleading results with some performance measures that are currently used to measure the performance of DMOO algorithms.

The first part of the thesis provided background with regards to optimisation. The second part of the thesis discusses CI algorithms used to solve optimisation problems. The next chapter discusses population-based algorithms used to solve SOOPs.

Part II

Computational Intelligence Algorithms

Chapter 5

Population-based Single-objective Algorithms

"One bee makes no swarm." – French proverb

This chapter provides an overview of two CI algorithms that are required as background for the DVEPSO algorithm that is introduced later in the thesis, as well as the algorithms against which DVEPSO are compared to (refer to Chapters 6, 7, 9 and 11). Section 5.1 discusses PSO, while GAs are discussed in Section 5.2.

5.1 Particle Swarm Optimisation

This section discusses the particle swarm optimisation (PSO) algorithm and the various steps of the algorithm. Section 5.1.1 discusses how the swarm is initialised and the conditions that will cause the algorithm to stop running are discussed in Section 5.1.2. The calculation of a particle's velocity is discussed in Section 5.1.3 and Section 5.1.4 discusses the calculation of a particle's position. Section 5.1.5 discusses the calculation of a particle's personal best and the swarm's global best.

Inspired by the social behaviour of bird flocks, Eberhart and Kennedy [94] introduced PSO. The PSO algorithm maintains a swarm of particles, where each particle represents a solution of the optimisation problem under consideration. Each particle moves through

the search space and the particle's position in the search space is updated based on its own experience (cognitive information), as well as the experience of its neighbours (social information). The particle's position that produced the best solution so far is referred to as its personal best or *pbest*. The position that lead to the best overall solution by all particles in a pre-defined neighbourhood, i.e. either the best of the neighbourhood's particles' pbests or the best of the current positions of the neighbourhood's particles, is called the neighbourhood best or *nbest*.

The first PSOs introduced by Eberhart and Kennedy are the global best PSO, or *gbest* PSO, and the local best PSO, or *lbest* PSO. The gbest PSO defines the neighbourhood of each particle as the whole swarm. In this case the neighbourhood best is also referred to as the global best or *gbest*.

The PSO algorithm is described in Algorithm 1. The main steps of the algorithm are described in more detail below.

Algorithm 1 PSO Algorithm

```
1.  create and initialise a swarm
2.      while stopping condition has not been reached
3.          for each particle in swarm do
4.              set pbest using Equation (5.6)
5.          set nbest using Equation (5.7) or Equation (5.8)
6.          for each particle in swarm do
7.              calculate new velocity using Equation (5.2) or Equation (5.4)
8.              calculate new position using Equation (5.5)
```

Before the PSO algorithm can run, certain values have to be set during the intialisation of the algorithm. The next section discusses the initialisation of the PSO.

5.1.1 Initialising the Swarm

The first step of the PSO algorithm initialises each particle's initial position, velocity and *pbest*, and sets swarm size, neighbourhood size, and the control parameters.

When initialising the particles' positions, it should be done in such a way that the particles uniformly cover the search space [55]. Therefore, assuming that the optimum

is located within the domain defined by the vectors $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ (the minimum and maximum range of the decision variables in each dimension), an efficient initialisation method for the particles' positions is [55]:

$$x_j(0) = x_{min,j} + r_j(x_{max,j} - x_{min,j}), \quad \forall j = 1, 2, \ldots, n_x \tag{5.1}$$

where $r_j \sim U(0, 1)$, n_x refers to the dimension of the decision vector $\mathbf{x}$ and x_j is the j-th dimension of $\mathbf{x}$. The random value of r_j should be generated with an uniform number generator to ensure an uniform spread of solutions after initialisation.

The particles obtain both random positions and random moving directions if their positions are randomly initialised (as indicated in Equation (5.1)) [55]. Therefore, the initial velocities are normally set to zero. However, if the velocities are randomly initialised, the velocity values should be chosen carefully, since their values can lead to large position updates causing the particles to move outside the search space within the first few iterations of the run.

The *pbest* of each particle is set to its initial position, i.e. $\mathbf{y}_i(0) = \mathbf{x}_i(0)$. The PSO control parameters are set to values that lead to convergent behaviour [63], for example inertia weight, $w = 0.72$ and $c_1 = c_2 = 1.49$ (refer to Equations (5.2) and (5.4)). However, it should be noted that optimal values for the control parameters that lead to convergent behaviour are problem dependent.

The next section discusses conditions that are used to determine when a PSO algorithm stops running.

5.1.2 Stopping conditions

The PSO algorithm will continue to execute until a specific stopping condition has been reached. The most common stopping conditions used are:

- Running a fixed number of iterations (or function evaluations).
- Stopping when an acceptable solution has been found. If the true optima is known, then for SOOPs a stopping condition can be to stop if the error between the found optima and the true optima is smaller than a specified value. However, for MOOPs and for SOOPs, if the true optima is unknown, a stopping condition can be to stop if the value of a specific performance measure is better than a specified threshold.

5.1.3 Calculating the Velocity

This section provides information about the general method that is used to calculate the velocity of each particle, calculating a particle's velocity using an inertia weight, and clamping the velocities of particles to reduce the step size of the particles.

General Calculation of Velocity

This section discusses the calculation of the particles' velocities. First a general calculation of the velocities are discussed and then modifications to the general calculation of velocity are discussed.

The velocity of a particle is calculated as follows:

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r_1}(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] + c_2\mathbf{r_2}(t)[\hat{\mathbf{y}}_\mathbf{N}(t) - \mathbf{x}_i(t)] \tag{5.2}$$

where $\mathbf{v}_i(t)$ and $\mathbf{x}_i(t)$ are the velocity and position of particle i at time step t respectively; $\hat{\mathbf{y}}_\mathbf{N}(t)$ represents the *nbest* of neighbourhood N (calculated using Equation (5.7) or Equation (5.8)) and $\mathbf{y}_i(t)$ represents the *pbest* (calculated using Equation (5.6)) at time t; $c_1\mathbf{r_1}(t)[\mathbf{y}(t) - \mathbf{x}(t)]$ is the cognitive component of the velocity and $c_2\mathbf{r_2}(t)[\hat{\mathbf{y}}_\mathbf{N}(t) - \mathbf{x}(t)]$ is the social component of the velocity; c_1 and c_2 are positive acceleration coefficients that influence the contributions of the cognitive and social components respectively; and $\mathbf{r_1}, \mathbf{r_2} \sim U(0,1)^{n_x}$ are random values sampled from an uniform distribution with n_x representing the number of decision variables or the dimension of the search space.

One problem with the general calculation of velocity using Equation (5.2) is that the velocity value of a particle can quickly become very large, especially when the particle is exploring an area in the search space that is far away from the particle's pbest or the swarms global best. A large velocity value results in a large position update, which results in the particle moving outside the feasible space. To overcome this problem, various modifications to the basic PSO have been suggested. Two of the modifications that have been made to the general calculation of velocity are discussed in more detail below, namely clamping the velocities of the particles and using inertia weight in the calculation of velocity.

Velocity Clamping

Without any intervention, a particle's velocity may increase in such a way that it may move outside the boundary constraints of the problem. One way of managing this is to clamp the particle's velocity, i.e. if a particle's velocity exceeds a predefined maximum velocity value, the particle's velocity is set to the maximum velocity value [51]. Mathematically, this is described as follows:

$$\mathbf{v}_i(t+1) = \begin{cases} \mathbf{v}'_i(t+1) & \text{if } \mathbf{v}'_i(t+1) < \mathbf{v_{max}} \\ \mathbf{v_{max}}(t+1) & \text{if } |\mathbf{v}'_i(t+1)| \geq \mathbf{v_{max}} \end{cases} \tag{5.3}$$

where $\mathbf{v_{max}}$ is the maximum velocity in each dimension and $\mathbf{v}'_i$ is the velocity of particle i, calculated using Equation (5.4).

Velocity clamping does not prevent a particle from moving outside the boundaries of the feasible space. However, it does restrict the step sizes of the particles. It should be noted that the selection of a $\mathbf{v_{max}}$ value should be carefully chosen and is problem dependent. A large $\mathbf{v_{max}}$ value will increase the swarm's global exploration, but the larger step sizes of the particles may cause the particles to jump over good solutions to continue searching in an area of the search space that does not contain good solutions [55]. A small $\mathbf{v_{max}}$ value will increase the swarm's local exploitation, but a too small $\mathbf{v_{max}}$ value may cause the swarm to only explore local good regions and not other good regions that are further away. Furthermore, with a too small $\mathbf{v_{max}}$ value the swarm may become trapped in local optima [55].

Another approach that is followed to prevent large position updates of particles, is using an inertia weight. The next section discusses how particles' velocities are calculated using an inertia weight and the effect that the inertia weight has on the velocities of the particles.

Calculating the Velocity using Inertia Weight

Shi and Eberhart [136] introduced the concept of an inertia weight to control the influence of previous flight magnitude (step size and direction) on the new velocity, i.e. the momemtum of a particle. The velocity calculation of Equation (5.2) can therefore be adapted as follows:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r_1}(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] + c_2\mathbf{r_2}(t)[\hat{\mathbf{y}}_\mathbf{N}(t) - \mathbf{x}_i(t)] \tag{5.4}$$

where w represents the inertia weight.

The value of w influences the exploration and exploitation ability of the swarm [136, 63]. Shi and Eberhart [136] investigated the effect of w values in the range $[0, 1.4]$ and the study's results indicated that better convergence was achieved with $0.8 \leq w \leq 1.2$, and $w > 1.2$ resulted in more failures in finding the global optimum due to particles leaving the search space.

Let $c_1 = c_2 = 0$ in Equation (5.4). Then, if $w > 1$, the particles' velocities will keep increasing over time, or will keep increasing over time until the maximum velocity is reached if velocity clamping is used. This increase in the velocity will cause the swarm to diverge and therefore, large w values facilitate more exploration. On the other hand, if $w < 1$, the velocities will keep decreasing over time until they reach zero or values close to zero. Therefore, small w values lead to local exploitation, but too small values reduce the swarm's exploration ability.

However, when $c_1, c_2 \neq 0$, the effect of w is not that easy to predict. According to Shi and Eberhart [136] w values close to 1.0 seems preferable. However, according to a study by Van den Bergh and Engelbrecht [152], to ensure convergence, w should be chosen in such a way that it adheres to the following relation: $w > \frac{1}{2}(c_1 + c_2) - 1$. From this relation it is clear that the values of the control parameters w, c_1 and c_2 cannot be selected independently to ensure convergence. Furthermore, the best values for these control parameters are problem dependent.

5.1.4 Calculating the Position

Once the new velocity of a particle has been calculated, its new position can be determined by adding the velocity to its current position as follows:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i + \mathbf{v}_i(t+1) \tag{5.5}$$

When a particle moves outside the boundaries of the search space, the particle's position is calculated differently from Equation (5.5) to pull the particle back into the

search space. Various ways exist to manage boundary constraint violations, as discussed below.

Dealing with Boundary Constraints

Most optimisation problems have boundary constraints (refer to Section 2.1) and therefore a particle should be prevented from moving outside the search space of the problem. If the solution is in close proximity of the bounds, it may be beneficiary to allow a particle to move outside the bounds to enable exploration in the proximity of the optima in the hope that the particle may find the optima. However, once a particle has moved outside the bounds, it should not be allowed to become an attractor (the particle's position should not be selected as either a *pbest* or *nbest*) to ensure that the particle doesn't attract other particles to the area outside the search space.

According to Chu *et al* [25], there are three basic approaches that are widely used to manage boundary constraint violations, namely:

- Random, where if any dimension of a particle's position is outside the search space, a random value from an uniform distribution between the lower and upper boundaries of the violating dimension is assigned to the violating dimension of the particle's position.
- Absorbing, where if a particle moves outside the search space, the dimension that is violating the bounds are set to the boundary of that dimension, so that it seems as though the particle has been absorbed by the boundary.
- Reflection, where if a particle moves outside the search space, the boundary acts like a mirror that reflects the projection of the particle's displacement by flipping the direction of the particle's velocity.

According to Engelbrecht [55], the following approaches can also be used to manage boundary constraint violations:

- Repairing, where if a particle moves outside the search space, it is pulled back into the search space by the *pbest* and *nbest*. For example, if a particle's position in dimension j violates the boundary constraints, the particle's velocity in dimension j, v_j, is set to zero to eliminate the influence of momentum for the j-th dimension, so that this dimension will be pulled back towards dimension j of *pbest* and *nbest*.

However, the particle will only be pulled back within the search space if the *pbest* is still within the search space.

- *pbest* selection, where particles are allowed to cross the boundaries, but if the position of a particle is not within the search space, its position cannot become the particle's *pbest*.

5.1.5 Calculating the *pbest* and *nbest*

This section discusses how the *pbest* and *nbest* is calculated. The equations that are used to update the *pbest* and *nbest* are provided. This section discusses issues that influence the manner in which the *pbest* and *nbest* are updated, namely whether the updates are done in a synchronous or asynchronous way and how the particles in the swarm are connected to each other.

For minimisation problems, the *pbest* at time $t+1$ is calculated as [94]:

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases} \tag{5.6}$$

where n_x represents the number of decision variables and the fitness function is represented by $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$.

The *nbest* at time t can be calculated as [55]:

- The best *pbest* found so far by all particles in the neighbourhood N_j, calculated as:

$$\hat{\mathbf{y}}_{N_j}(t) \in \{N_j \mid f(\hat{\mathbf{y}}_{N_j}(t)) = \min\{f(\mathbf{y}_i(t))\},\ \forall \mathbf{y}_i \in N_j\} \tag{5.7}$$

- The best position of all positions found by the particles in neighbourhood j at a specific time step t, calculated as:

$$\hat{\mathbf{y}}_{N_j}(t) \in \{N_j \mid f(\hat{\mathbf{y}}_{N_j}(t)) = \min\{f(\mathbf{x}_i(t))\},\ \forall \mathbf{x}_i \in N_j\} \tag{5.8}$$

Synchronous and Asynchrounous Updates

The *pbest* and *nbest* values can be updated in either a synchronous or asynchronous manner [21]. *Synchronous* updates of these values are done when all the particles' positions are first calculated and their *pbest* values are updated, and then the *nbest* value

is calculated. Algorithm 1 uses synchronous updates. With synchronous updates, feedback about the best search region is only given once per iteration and all of the particles' knowledge about the *nbest* is the same and updated at the same time. Therefore, Carlisle and Dozier [21] suggest that synchronous updates is suitable for the *gbest* PSO. With *asynchronous* updates, the new *nbest* position is calculated after each particle's position and *pbest* update. Immediate feedback about the best regions in the search space is provided and feedback occurs many times during one iteration. Therefore, according to Carlisle and Dozier [21], asynchronous updates are more suitable for the *lbest* PSO.

Neighbourhood Topologies of Particle Swarm Optimisation

The topology or connections between particles in a swarm influences the communication flow between the various particles. Various topologies have been developed for PSO, but only the topologies used by the *lbest* and *gbest* PSO are discussed in this section. The reader is referred to [26, 55, 95, 126] for more information on the different PSO topologies.

The **star** topology connects all particles to each other and therefore every particle can communicate with all other particles. In this case the neighbourhood of each particle is the whole swarm and therefore the *nbest* is also referred to as the global best or *gbest*. Equations (5.7) and (5.8) are then adapted to calculate the *gbest* as follows: The *gbest* at time t can be calculated as either [55]:

- The best *pbest* found so far by all particles in the swarm, calculated as:

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_p}(t)\} | f(\hat{\mathbf{y}}(t)) = min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_p}(t))\} \tag{5.9}$$

  where n_p is the number of particles in the swarm and $\hat{\mathbf{y}}$ is the *gbest*.
- The best position of all positions found by the particles at a specific time step t, calculated as:

$$\hat{\mathbf{y}}(t) = min\{f(\mathbf{x}_0(t)), \ldots, f(\mathbf{y}_{x_{n_p}}(t))\} \tag{5.10}$$

Since all particles are connected to each other with a star topology, information about the *gbest* can be quickly distributed to all particles, attracting all particles to the best part of the search space, which may lead to faster convergence. However, since a PSO with a star topology has only one attractor (the *gbest*), and the extent of coverage of

the search space is less than with a less connected topology, if the global optima is not close to the *gbest*, the PSO may be trapped in local optima [95, 55]. The *gbest* PSO has a star topology.

A less connected topology than the star topology is the **ring** topology, where each particle communicates directly with only a predefined number of immediate neighbours. It is important to note that the different neighbourhoods in the swarm overlap so that a specific particle can belong to more than one neighbourhood. Allowing particles to belong to more than one neighbourhood enables communication or exchange of information between neighbourhoods. Information about the best solution flows slower through the ring topology than through the star topology, leading to slower convergence. However, with the ring topology a larger area of the search space is covered than with the star toplogy, and it has many attractors (*nbests*), and therefore it is less susceptible to local optima [95, 55]. The *lbest* PSO has a ring toplogy.

Many variations of the original PSO have been developed through the years. However, these variations are beyond the scope of this thesis and the reader is referred to [4, 5, 26, 52, 55, 126] for more information on the various PSO algorithms.

5.2 Genetic Algorithms

The concept of genetic algorithms (GAs) was first described by Holland [82]. GAs are based on concepts of Darwanian evolution. Every living organism consists of cells, where each cell contains the same set of *chromosomes*, i.e. an organised structure of DNA and protein. Each chromosome consists of blocks of DNA referred to as *genes*, where each gene encodes or represents a particular protein or trait, such as the organism's hair colour. The possible values that a gene can have are called *alleles*, for example the eye colour that can be blue, green or brown [121]. Using the metaphor of "survival of the fittest", a GA uses a population of individuals to evolve towards better solutions. When an optimisation problem is solved using a GA, each individual's characteristics are represented by a chromosome. Each chromosome is a combination of the decision variables that have to be optimised, where each decision variable is referred to as a gene.

Algorithm 2 provides a template for a GA.

Algorithm 2 Genetic Algorithm

```
1.  create and initialise the population
2.      while stopping condition has not been reached
3.          for each individual of the population do
4.              evaluate the fitness
5.          select parents for cross-over
6.              perform cross-over on parents to produce offspring (children)
7.          select offspring for mutation
8.              perform mutation on offspring
9.          select a new population for the next generation
```

This section discusses GAs and the various steps of the GA. Section 5.2.1 discusses GA control parameters. Evaluation of the various solutions that are found is discussed in Section 5.2.2 and various selection operators are discussed in Section 5.2.3. Section 5.2.4 discusses the cross-over operator and Section 5.2.5 discusses the mutation operator.

5.2.1 Initialising the Population

The original GA as proposed by Holland used a binary representation for the chromosome. An optimisation problem with a n_x-dimensional search space is then represented by chromosomes that consist of n_x bit-valued strings, i.e. $x_j \in \{0, 1\}$, where each bit string represents a decision variable. If the decision variables are binary variables, the length of the chromosome is n_x bits. However, if the variables have nominal values, the chromosome has n_x bit vectors of length n_d, where each decision variable can have 2^{n_d} values. If the variables have continuous values, a mapping function is used to convert the continuous value to a bit vector. The mapping function is formulated as $\phi : \mathbb{R} \rightarrow \{0, 1\}^{n_d}$. It should be noted that GAs and operators have been developed where floating point presentation can be used directly [81].

As for PSO, chromosomes are initialised to uniformly cover the search space.

5.2.2 Fitness Evaluation

In nature, usually the fittest of the specie survives. Therefore, a GA should use the better solutions to create the new solutions and the best solutions should survive to the next generation. In order to ensure that the GA adheres to this principle, some method of quantifying the quality of a solution represented by a chromosome should be used. Similar to the PSO, this is done using a fitness function that maps the chromosome to a scalar value, expressing the quality of the candidate solution.

5.2.3 Selection Operator

Various approaches exist to select solutions from a specific population, called selection operators. When GAs are used to solve optimisation problems, solutions are selected for the following three steps in the algorithm:

- Solutions from the parent population are selected for cross-over to create new solutions called *offspring* (refer to steps 5 and 6 in Algorithm 2).
- A solution from the offspring population is selected for mutation to create new solutions or offspring (refer to steps 7 and 8 in Algorithm 2).
- Solutions are selected from the parents and offspring for the next generation of the GA (refer to step 9 in Algorithm 2).

Selection operators can be quantified according to their selection pressure (or take-over-time), i.e. the time it takes for the best solution to occupy all but one population slot (individual) if a specific selection operator is repeatedly applied to a population [69]. When a selection operator has a high selection pressure, the population will lose diversity quickly, which may lead to premature convergence to sub-optimal solutions. On the other hand, if the selection pressure is low, the GA's search procedure will behave more like a random search process [40], since the best solutions are not emphasised during selection.

The most common selection operators are tournament selection, proportionate selection, rank-based selection, random selection and elitist selection [40, 55].

Tournament Selection

Tournament selection randomly selects n_{ts} number of individuals from the population (n_{ts} is smaller than the total number of individuals in the population, n_s), compares their fitnesses against each other and then selects the individual with the best fitness. If tournament selection is performed without replacement, the selected individuals are not considered for the next selection. If selection is done for cross-over, tournament selection is performed repeatedly until the required number of parents for cross-over have been selected. If $n_{ts} = 2$, tournament selection is referred to as binary tournament selection.

If n_{ts} is not too large, tournament selection has a small selection pressure, since the best individual has a smaller chance of being selected. However, if n_{ts} is too large, it may occur that the best individual is selected more than once. Furthermore, if n_{ts} is too small, a bad individual may be selected. Therefore, the value of n_{ts} plays a huge roll in the selection pressure of tournament selection.

Proportionate Selection

With proportionate selection, the probability of selecting $\mathbf{x}_i$, $p(\mathbf{x}_i)$, is proportionate to the fitness of $\mathbf{x}_i$, calculated as [55]:

$$p(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)}{\sum_{j=1}^{n_i} f(\mathbf{x}_j)} \tag{5.11}$$

with $p(\mathbf{x_i})$ the probability that individual $\mathbf{x}_i$ will be selected.

Proportionate selection has a large selection pressure and individuals with a good fitness value will have a better chance of being selected, which can decrease the diversity of the population in the next generation. Individuals are selected by sampling the distribution created using Equation (5.11). For example, individuals can be selected according to roulette wheel selection (RWS) where the fitness values are normalised by dividing the fitness value by the maximum fitness value. The probability distribution is then a roulette wheel, where the size of each slice is proportional to the normalised selection probability of an individual. Selecting an individual is then similar to spinning the roulette wheel, recording which slice ends up at the top and then selecting the corresponding individual. With this approach, the wheel has to be spun as many

times as the number of individuals that have to be selected [40, 55]. In other words, random numbers have to be created (simulating a spin) for each individual that has to be selected. However, instead of creating N random numbers for N individuals that have to be selected, stochastic universal sampling (SUS) can be used where only one random number is selected for the entire selection process [40]. If N individuals have to be selected, a set of N equally spaced numbers is created:

$$R = \left\{ r,\ r + \frac{1}{N},\ r + \frac{2}{N}, \ldots,\ r + \frac{N-1}{N} \right\} \%1 \tag{5.12}$$

where $\%$ is the modulus operator.

The corresponding individuals, based on their selection probability, are then selected according to set R.

Rank-based Selection

Rank-based selection ranks the individuals' fitness values and the individuals' selection probability is then calculated based on the rank and not based on the absolute fitness values. Since the selection is independent of the actual fitness value, the best individual will not dominate the selection process. Therefore, rank-based selection has a small selection pressure. The ranking of the fitness values can be done in many ways, of which two are a linear ranking approach or an exponential ranking approach [54, 55].

Random Selection

Random selection selects an individual randomly from the population, and all individuals have an equal chance of being selected. Therefore, random selection has a small selection pressure, since weaker individuals have the same chance of being selected than the best individual.

Elitist Selection

The goal of elitist selection is to ensure that the best individuals of the current population survive to the next generation. The best individuals are added to the new population without performing mutation. Therefore, elitist selection has a large selection pressure.

However, if many individuals survive to the next generation, the population's diversity will decrease.

5.2.4 Cross-Over Operator

The goal of cross-over is to produce offspring that hopefully produce better solutions than their parents. A cross-over operator is applied to a specified number of selected parent solutions and their genetic material or genes are recombined to produce one or more offspring.

When the chromosomes are represented by bit vectors, i.e. the genes have binary representations, uniform- [143], one-point- [82, 92], two-point and n-point [92] cross-over can be used. If the genes have continuous values, other cross-over methods are used, for example linear-, blend- and simulated binary crossover [40]. More information on the various cross-over approaches for binary-values genes and continuous-valued genes can be found in [40, 139, 140].

In order to preserve some good individuals, the cross-over operator is not applied to all parents, but only to a percentage of the parent population. The percentage of the populuation that is selected for cross-over is specified by the cross-over probability, p_c. If p_c is small, only a few new solutions (offspring created with cross-over) are introduced to the selection pool, leading to a smaller area of the search space being searched. However, if p_c is high, many new solutions are created and only a few of the individuals are preserved in the next generation. This will lead to a bigger area of the search space being searched, but good genetic material may get lost in the process.

5.2.5 Mutation Operator

When cross-over is applied, good genetic material may get lost in the process. This problem can be addressed through mutation, since mutation introduces more diversity into the population. When the mutation operator is applied to the selected offspring, the value of randomly selected elements of the chromosome is changed. Each chromosome has a mutation probability p_m of being selected for mutation. If p_m is small, only a small number of genes are mutated, thereby increasing the diversity of the solutions, but

not changing the individuals too drastically. Furthermore, a small p_m may re-introduce genetic material that got lost during cross-over. However, if p_m is large, many genes are changed, almost no genetic material of previous solutions are preserved, and the search process becomes almost similar to a random search [140].

Mutation changes the value of selected genes. In the case of bit-valued genes, the bit value is simply negated. In the case of floating-point values, a mutational step size is sampled from some zero-mean distribution and added to the gene's value [159].

5.3 Summary

This chapter discussed two computational intelligence algorithms, namely PSO and GA, that are required as background for the vector evaluated approaches discussed in Chapter 7, as well as the DMOO algorithms discussed in Chapters 9 and 11.

Section 5.1 provided information about PSO and the various steps of the basic PSO algorithm. The various steps of the basic PSO that were discussed are: the initialisation of the PSO swarm, the conditions under which the PSO algorithm will stop running, the calculation of a particle's position and velocity, and the calculation of a particle's personal best and the swarm's global best.

GAs and the various steps of a GA were discussed in Section 5.2. The various steps of a GA that were discussed, are: how to initialise the population, how to calculate the fitness of an individual, selection of individuals, and cross-over and mutation operators.

The next chapter discusses population-based approaches that were used to solve optimisation problems with more than one objective.

Chapter 6

Population-based Multi-objective Optimisation Algorithms

> *"When it is obvious that the goals cannot be reached, don't adjust the goals, adjust the action steps."* – Confucius

Most problems in real-life have more than one goal or objective that are in conflict with one another - by improving one objective the solutions get worse with regards to the other objective(s). Therefore, the goal of a MOO algorithm is to find the set of optimal trade-off solutions called the POF.

This chapter discusses three MOO algorithms that are required as background for Chapter 8. Chapter 8 discusses versions of these algorithms adapted for DMOO. The performance of the adapted versions of these algorithms are compared against the DMOO algorithm presented in this thesis, namely DVEPSO (refer to Chapter 11).

Section 6.1 provides a short overview of the MOO field. NSGA-II, a multi-population GA, is discussed in Section 6.2. A multi-population PSO, MOPSO is discussed in Section 6.3. Section 6.4 discusses a multi-population cooperative and competitive EA, CCEA. A summary of the chapter is provided in Section 6.5.

6.1 History of Multi-Objective Optimisation

Various CI algorithms have been used to solve MOOPs. This section provides a short summary of the major contributions in the field. The information provided in this section is by no means complete and the reader is referred to [35, 36, 38] for a more detailed discussion of MOO.

David Schaffer's VEGA approach [131] is generally seen as the first implementation of a multi-objective evolutionary algorithm (MOEA) [31]. VEGA is a multi-population approach where each sub-population solves only the one objective assigned to the sub-population. Proportional selection is performed on the sub-population based on only the one objective. All of the selected individuals are then placed together in a new combined population on which cross-over and mutation are applied using a standard GA. VEGA is discussed in more detail in Section 7.1.

However, after Schaffer presented VEGA, researchers mostly used aggregation of objective functions and lexicographic ordering to solve MOOPs [31]. Aggregation methods combine the objective functions into a single objective, which is then used as the fitness function. The objectives can be combined in both a linear and non-linear way. If non-linear aggregation is used, the new objective function, f^*, is created as a weighted sum of the various objective functions, $f_1, f_2, \ldots, f_k$, as follows:

$$f^* = \sum_{j=1}^{k} w_j f_j \tag{6.1}$$

where w_j is the weight of objective function f_j, indicating the relative importance of the specific objective function.

The main problem when using linear aggregation is that the algorithm struggles to generate solutions when the POF is nonconvex [40]. Lexicographic ordering orders the objectives according to importance. The objective that is considered as the most important is then chosen and optimised without considering the other objectives. This process is repeated, optimising all objectives one by one, in the order of importance, until all objectives have been optimised.

The introduction of the concept of *Pareto ranking* by Goldberg [68] changed the way in which EAs are used to solve MOOPs. Pareto ranking is illustrated in Algorithm 3.

Algorithm 3 Pareto ranking

1. set population P_r equal to the whole population P
2. set $i = 0$
3. while P_r is not empty
4. select individuals from P_r that are non-dominated
5. assign rank i to the selected individuals and remove them from P_r
6. increment i

The individuals with rank 1 are similar to non-dominated solutions stored in an external archive.

Even though Goldberg did not implement Pareto ranking himself, most MOEAs that followed after his suggestion, implemented Pareto ranking [60, 83, 141]. The goal of Pareto ranking is to enable an algorithm to converge to the POF.

The early MOO algorithms also faced the problem of preventing the EA from converging to a single solution [31]. To overcome this problem, Goldberg and Richardson suggested the use of a niching technique [45, 70] to increase the diversity of the found non-dominated solutions. A commonly used niching technique is fitness sharing [30, 70], where the fitness of an individual $\mathbf{x}_i$ is degraded by the presence of individuals that are within a specified niche radius (σ_{share}) of $\mathbf{x}_i$.

The introduction of the strength Pareto evolutionary algorithm (SPEA) by Zitzler [172] changed the type of MOEAs that were presented and are still developed today to solve MOO. SPEA incorporates elistim through the use of an archive that stores the non-dominated solutions that have been found so far by the algorithm. The archive is used to ensure that the non-dominated solutions reported at the end of the algorithm run are solutions that are non-dominated with regards to all solutions that have been found by the algorithm throughout the run. It should be noted that elitist selection with regards to GAs refers to ensuring that the best individuals of the current population survive to the next generation (refer to Section 5.2.3). However, in the context of MOO, elitism can be seen as retaining the best solutions obtained by the algorithm throughout the entire run, i.e. solutions that remained non-dominated throughout the entire run of the algorithm.

6.2 Non-dominated Sorting Genetic Algorithm II

NSGA-II, introduced by Deb *et al.* [47, 42], is an improved version of non-dominated sorting genetic algorithm (NSGA) developed by Srinivas and Deb [141]. This section discusses the various steps of NSGA-II. Section 6.2.1 discusses the basic NSGA-II. The fast non-dominated sorting approach used by NSGA-II is discussed in Section 6.2.3. Section 6.2.4 discusses the approach that is followed to maintain the diversity of the non-dominated solutions.

6.2.1 NSGA-II

NSGA-II is a multi-objective genetic algorithm (MOGA) that uses non-domination. The various steps of the algorithm is illustrated in Figure 6.1 and listed in Algorithm 4.

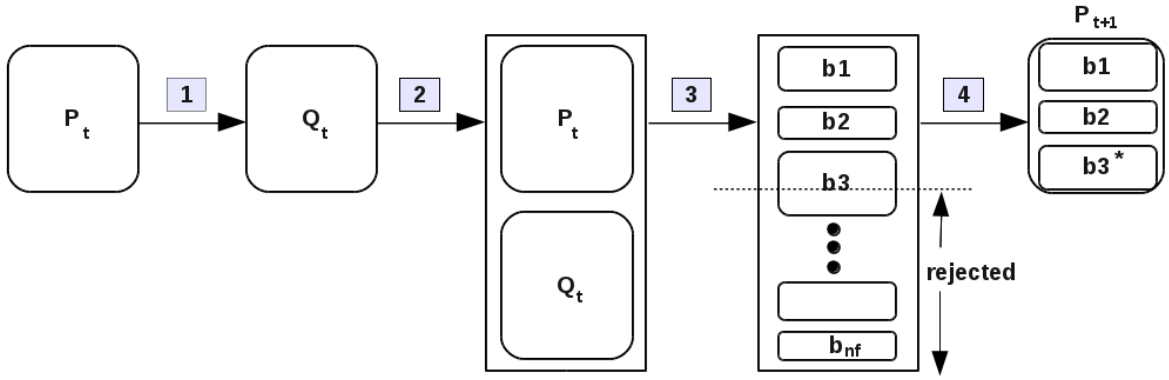


Figure 6.1: Steps of NSGA-II

Algorithm 4 NSGA-II

1. create and initialise a random parent population
2. while stopping condition has not been reached
3. produce offspring (step 1 in Figure 6.1)
5. combine parents and offspring in a combined population R (step 2 in Figure 6.1)
6. sort R according to Pareto-ranking (step 3 in Figure 6.1)
7. select individuals for the new population (step 4 in Figure 6.1)

The steps of NSGA-II highlighted in Figure 6.1 are discussed in more detail below.

6.2.2 Producing Offspring

The first step in Figure 6.1 is to produce offspring. Before the algorithm starts, an initial parent population, P_0, is randomly created. P_0 is then sorted based on non-domination, where each individual is assigned a fitness based on the individual's non-domination level. Binary tournament selection is performed to select parents for cross-over to produce n_p offspring. Once the offspring has been created, mutation is performed on the offspring.

6.2.3 Fast Non-dominated Sorting

After the offspring has been created, the parents and offspring are combined to create a new population, R, of size $2n_p$. This is the second step in Figure 6.1. Pareto-ranking is performed on R, i.e. R is sorted according to the individuals' level of non-domination. This sorting process is the third step in Figure 6.1. In order to speed up the sorting process, Deb *et al.* introduced the fast non-dominated search procedure for NSGA-II [47, 42].

Fast non-dominated search places the first individual of R in a new population P'. Each individual, $\mathbf{x}_i \in R$, is then compared against all individuals in P'. If $\mathbf{x}_i$ dominates any solutions in P', the dominated individuals are removed from P'. If $\mathbf{x}_i$ is dominated by any individual in R, $\mathbf{x}_i$ is ignored and not placed in P'. However, if $\mathbf{x}_i$ is non-dominated with regards to all individuals in R, $\mathbf{x}_i$ is placed in R. After all $\mathbf{x}_i \in R$ have been compared against the individuals in R, the individuals in P' is part of the first front. To determine the second front, all members of the first front are removed from R and not considered. The search process is then repeated. This whole process is repeated until all fronts have been found.

6.2.4 Selecting a New Population

Once all the individuals in R has been assigned a front, a new population, P_{t+1}, is selected for the next generation. This selection process is explained in Algorithm 5.

Algorithm 5 NSGA-II Population Selection

```
1.  if |P_{t+1}| < n_p
2.      for each front b_i
3.          if |b_i| < (n_p − |P_{t+1}|)
5.              add all individuals in b_i to P_{t+1}
6.          else
7.              perform crowded sorting on b_i
8.              while |P_{t+1}| < n_p
9.                  add best individual from sorted b_i
10.                 remove best individual from sorted b_i
```

Crowded sorting [47, 42] is used to ensure diversity of solutions. In order to perform crowded sorting, the individuals' crowding distances are determined. The population is sorted with regards to each objective function in ascending order. Then, for each objective function:

- The boundary individuals, i.e the individuals with the highest and lowest objective function value, are assigned an infinite crowding distance value.
- The other individuals are assigned a crowding distance equal to the distance in objective space between the two adjacent points of the individual, i.e. the adjacent points on each side of the individual in objective space.

Each individual's crowding distance is then calculated as the sum of the individual's crowding distance values for each objective.

After the crowding distances have been calculated, crowded sorting are performed on the individuals. When two individuals are compared using the crowded sorting operator, then

- if two individuals have different non-domination ranks, the individual with the lowest domination rank is selected.
- if both individuals have the same non-domination ranks and therefore belong to the same front, the individual with the largest crowded distance is selected. This ensures that the individual that is located in the less dense area is selected and thereby increases the diversity of the set of non-dominated solutions.

In NSGA-II, NSGA's fitness sharing scheme is replaced with the crowded sorting operator. Another difference between the two algorithms is NSGA-II's faster non-dominating sorting procedure [42, 47]. These changes improves NSGA-II's performance with regards to computational complexity. Furthermore, the performance of NSGA-II is so good, that it has become a benchmark against which other MOOs algorithms are compared against [32].

It is important to note that NSGA-II does not make use of an archive, but preserves elitism through its selection mechanism. NSGA-II uses $(\mu + \lambda)$-selection [9, 134] where μ represents the number of parents and λ the number of offspring produced from the parents. In other words, NSGA-II selects the best μ individuals from both the parents and the offspring for the next generation.

A disadvantage of the crowded sorting operator is that it may cause a Pareto-optimal solution that is in a crowded area (with other non-dominated solutions that are not necessarily Pareto-optimal) to be deleted and non-dominated solutions that are not Pareto-optimal but located in a less crowded area to be preserved [38].

6.3 Multi-objective Particle Swarm Optimisation

The MOPSO algorithm was introduced by Coello Coello and Salazar Lechuga [33] as one of the first PSO algorithms extended for MOO. Algorithm 6 lists the various steps of the MOPSO algorithm.

Before the MOPSO algorithm can be executed, the swarm is initialised as discussed in Section 6.3.1. Section 6.3.2 discusses the calculation of the particles' velocity. The approach used to calculate the swarm's local guide is discussed in Section 6.3.3.

6.3.1 Initialising the Swarm

Similar to PSO, the first step of the MOPSO algorithm initialises each particle's initial position, velocity and *pbest*, and sets swarm size, neighbourhood size, and the control parameters. The particles' initial positions are initialised in such a way that the particles are uniformly spread over the search space. The particles' velocities are initialised to zero and their *pbests* are set to their current positions.

Algorithm 6 MOPSO Algorithm

```
1.  create and initialise a swarm
2.      while stopping condition has not been reached
3.          for each particle in swarm do
4.              calculate new velocity using Equation (5.4)
5.              calculate new position using Equation (5.5)
6.              manage boundary constraint violations
7.          update archive
8.          update the particles' allocation to hypercubes
8.          for each particle in swarm do
9.          update pbest
```

In addition to the PSO initialisation, the particles are evaluated and the positions of the particles that are non-dominated are stored in the archive. Furthermore, the search space that has been explored so far is divided into hypercubes and all particles are placed in a hypercube based on the particle's position in objective space.

6.3.2 Calculation of Velocity

MOPSO uses the velocity equation of PSO (refer to Equation (5.4)) to update the velocity of the particles. However, the *gbest* in Equation (5.4) is a global guide that MOPSO selects from the archive as follows:

- hypercubes containing more than one particle is assigned a fitness equal to $\frac{f(\mathbf{x})}{n_p}$ where $f(\mathbf{x}) > 1$ and n_p is the number of particles in the swarm. This ensures that hypercubes that contain more particles will have a lower fitness value.
- roulette-wheel selection is used on the fitness values to select the hypercube from which the guide is selected.
- a particle is randomly selected from the winning hypercube and selected as the global guide.

6.3.3 Calculation of *pbest*

MOPSO [33] updates each particle's *pbest* (referred to as the local guide) as follows: the new position of the particle is compared to the particle's *pbest*, taking all objective functions into account, and

- if the new position dominates the current *pbest*, the new position is selected as the *pbest*, otherwise
- if the new position is non-dominated with regards to the current *pbest*, the new *pbest* is randomly selected between the particle's position and the current *pbest*.

In contrast to NSGA-II, MOPSO uses an archive to preserve elitism. However, the original version of MOPSO struggled to converge to the true POF in the presence of many local POFs [34]. To overcome this problem, Coello *et al.* [34] introduced an updated version of MOPSO that uses a mutation operator. Initially, the mutation operator is applied to all particles, but then the number of particles being mutated decreases rapidly as the number of iterations increases. The goal of the mutation operator is to increase the swarm's exploration ability. However, the mutation operator is not only applied to the particles, but also to the range of each decision variable of the MOOP. This leads to the whole range of each decision variable to be included in the beginning of the search, but then as the number of iterations increases, the range of each decision variable decreases. Coello *et al.* [34] compared the performance of the MOPSO with the mutation operator against three other MOO algorithms, namely NSGA-II, Micro-GA [29] and pareto archived evolution strategy (PAES) [98] on five constrained MOO benchmark functions. The results of the study indicated that MOPSO with the mutation operator was the only MOO algorithm able to find solutions along the full extend of the POF for all benchmark functions.

6.4 Cooperative-coevolution Evolutionary Algorithm

CCEA, introduced by Tan *et al.* [147], is a multi-population MOO algorithm. The basic CCEA is discussed in Section 6.4.1 and the initialisation of CCEA is discussed in Section 6.4.2. Section 6.4.3 discusses the evaluation of individuals and the calculation of the niche count. Rank assignment of the individuals are discussed in Section 6.4.4.

Various genetic operators performed on the individuals are discussed in Section 6.4.5. Section 6.4.6 discusses the extending operator that is used to increase the diversity of the found non-dominated solutions.

6.4.1 CCEA

CCEA is a co-evolutionary MOO algorithm. Co-evolution can be classified into two main categories, namely competitive co-evolution and cooperative co-evolution [147]. The goal of competitive co-evolution is to obtain more competitive individuals through competitive interaction with one another (similar to the predator versus prey scenario in nature). The goal of cooperative co-evolution is to obtain better individuals through cooperation or collaboration between various independently evolving species or populations. CCEA incorporates both categories of co-evolution. The steps of the basic CCEA are listed in Algorithm 7 and discussed in more detail below.

Algorithm 7 CCEA

```
1.  create and initialise random sub-populations
2.      while stopping condition has not been reached
3.          for each parent sub-population
5.              for each individual in sub-population
6.                  evaluate the individual
7.                  update the archive
8.              for each individual in sub-population
9.                  assign rank to individual
10.                 calculate the niche count of individual
11.             perform genetic operators
12.         perform the extending operator
```

6.4.2 Initialisation

Before the algorithm starts, the sub-populations are created. The number of sub-populations are equal to the number of decision variables of the MOOP. Each sub-

population is assigned one decision variable to optimise.

6.4.3 Evaluation of Individuals

Since each sub-population only optimises one decision variable, each individual in a sub-population is only a sub-component of a solution of the MOOP. Therefore, in order to evaluate the fitness of an individual, the individual is combined with representatives from other sub-populations to obtain a complete solution. The best individual in each population is selected as a representative.

The non-dominated solutions found so far by the algorithm is stored in an archive to preserve elitism. Once an individual is combined with a representative from each of the other sub-populations, the combined vector is evaluated against the solutions in the archive, and

- if the archive is empty, the combined objective vector is placed in the archive.
- if the archive is not empty:
  - if the combined objective vector is dominated by any of the solutions in the archive, the combined objective vector is not added to the archive.
  - if the combined objective vector dominates solutions in the archive, the dominated solutions are removed from the archive and the combined objective vector is added to the archive.
  - if the combined objective vector is non-dominated with regards to all solutions in the archive, and
    - the archive is not full, the combined objective vector is added to the archive.
    - the archive is full, niche count (fitness sharing) is used to determine whether a solution in the archive is replaced with the combined objective vector, and if so, which solution in the archive is replaced with the combined objective vector.

Niche count indicates how many solutions are within a specified distance of an individual. Therefore, a high niche count indicates that the individual is in a crowded area. The solution with the highest niche count will be selected to be replaced by a new solution if the archive is full.

Improper values for the radius, σ_{share}, leads to a bad distribution of solutions. However, it is not a trivial task to determine a good value for σ_{share} if the shape of the POF is unknown. Therefore, Tan *et al.* [147] calculated the niche count in a normalised objective space, where the objective space is normalised at each generation with an estimation of the ranges of the objective space. The ranges of the objective space are estimated according to the objective values in the archive at the specific generation.

6.4.4 Rank Assignment

The combined objective vector cannot be used directly as an individual's fitness in CCEA since the individual's population only optimises one decision variable. However, the combined objective vector can be used to evaluate how well an individual cooperates with other sub-populations to produce good solutions. This evaluation is done with a canonical Pareto-ranking scheme [147], where individuals are ranked according to the number of individuals in the archive that dominates the individual in objective space.

6.4.5 Genetic Operators

Tournament selection is performed in each sub-population to select parents for cross-over to produce offspring. Tournament selection is based on the individuals' rank, and if more than one individual have the best rank, the individual with the lowest niche count wins the tournament. Uniform cross-over is applied to the parents to produce offspring and bit-flip mutation is performed on the offspring.

6.4.6 Extending Operator

In order to improve the diversity of solutions, an extending operator is introduced. In CCEA, the archive member with the lowest niche count, and therefore in the least crowded region of the archive, is cloned and copied to the sub-populations. Copying this archive member to the sub-populations increases the chance that the archive member's components are selected into the mating pool and thereby attracting sub-populations to less explored regions of the search space. The steps of the extending operator are listed in Algorithm 8.

Algorithm 8 CCEA Extending Operator

```
1.  if the archive is full
2.      for each solution in archive
3.          calculate the niche count
5.          find the solution with the lowest niche count, p_{nc_l}
6.              for each sub-population
7.                  for n times
8.                      randomly select an individual
9.                      replace selected individual with cloned copy of p_{nc_l}
```

The number of cloned copies that are used is set to one. Tan *et al.* [147] compared the performance of CCEA against five MOEAs on three MOO benchmark functions. The results indicated that CCEA achieved the best overall performance with regards to converging to the true POF, as well as obtaining an even spread of solutions along the found POF.

One disadvantage of CCEA is that the number of sub-populations increases linearly as the number of decision variables increases. Therefore, Tan *et al.* [148] proposed a distributed CCEA.

6.5 Summary

This chapter provided a brief summary of important contributions to the field of MOO. Furthermore, three MOO algorithms required as background for later chapters in the thesis were discussed. NSGA-II, an improved version of NSGA, is a MOGA that uses non-domination and preserves elitism through a selection procedure. MOPSO is one of the first PSO algorithms extended for MOO and preserves elitism through an archive. CCEA is a multi-population MOO algorithm that uses cooperative co-evolution. Each population only optimises one decision variable of the MOOP and then cooperates with the other sub-populations to produce good solutions.

The next chapter discusses vector-evaluated approaches that were used to solve MOOPs.

Chapter 7

Population-based Multi-objective Vector Evaluated Approaches

"To adapt, is to move ahead." – Byron Pulsifer

This chapter discusses vector evaluated CI approaches that were introduced to solve MOOPs. Section 7.1 discusses VEGA, one of the first algorithms solving MOO without aggregating the objective functions to change the MOOP into a SOOP. VEPSO, based on the idea of VEGA but using PSO, is discussed in Section 7.2. Section 7.3 discusses another modification of VEGA, namely VEDE, where DE is used instead of GA. A hybrid algorithm based on the concept of VEPSO using different types of sub-algorithms is discussed in Section 7.4. Finally, a summary is provided in Section 7.5.

7.1 Vector Evaluated Genetic Algorithm

This section discusses the VEGA algorithm, introduced by Schaffer [131], which can be seen as the first implementation of a MOEA [31, 40]. The main steps of VEGA performed at each iteration when solving a MOOP with n_o objectives, are illustrated in Figure 7.1. These steps are:

1. the population is shuffled, the shuffled individuals are randomly divided into n_o sub-populations, and each sub-population is assigned a different objective function.

2. fitness values are assigned to individuals in each sub-population according to the sub-population's objective function. Individuals from each sub-population are selected for the mating pool using proportionate selection (refer to Section 5.2.3).
3. cross-over and mutation are applied to the mating pool to produce the next generation's population.

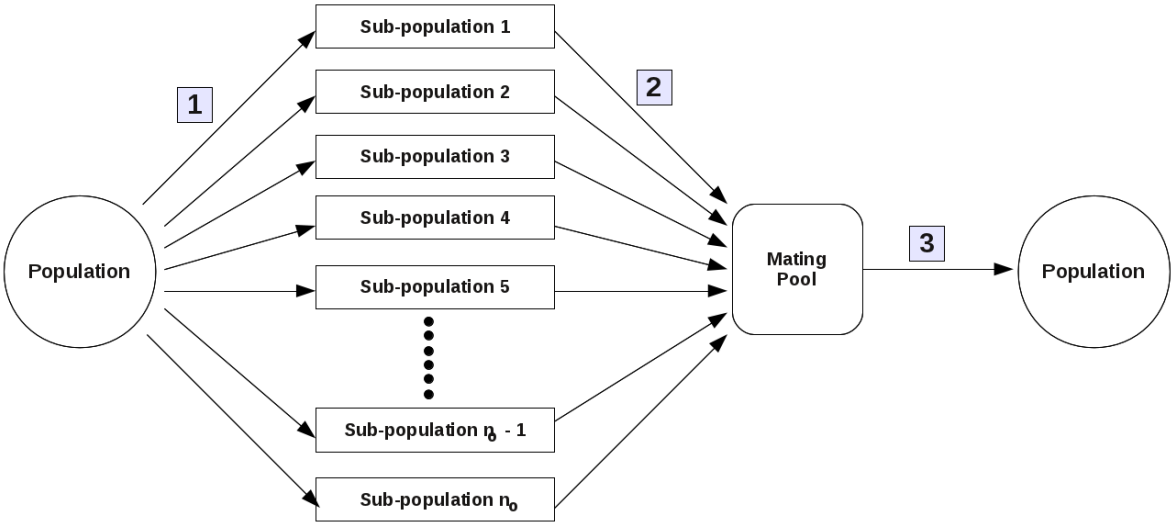


Figure 7.1: Steps of the VEGA algorithm at each generation

Each sub-population optimises one objective function and the knowledge between the different sub-populations are shared through cross-over and mutation applied to the mating pool containing individuals selected from each sub-population. However, since the selection operator is only performed per sub-population, the fitness of the individuals that are selected and placed in the mating pool are only measured based on one objective. This leads to preference of individuals that perform well with regards to the specific objective function assigned to the sub-population that the individual belongs to. Therefore, solutions that do not excel in one particular objective function, but that perform reasonably well for all objective functions (referred to as *middling* solutions), are disregarded during the selection process. This leads to an approximated POF with solutions in only certain areas of the POF. Therefore, the found POF will have a low diversity or spread of solutions. Initially, Schaffer expected that cross-over and

mutation performed on a mating pool that contains solutions from all sub-populations will eliminate this problem. However, results indicated that even when using this mating pool, VEGA still did not find the middling solutions. Therefore, Schaffer [131] proposed two changes to VEGA to overcome this problem, namely:

- using a heuristic selection preference for non-dominated individuals: In stead of only measuring an individual's fitness according to one objective function, all objectives are taken into account and only non-dominated individuals can be selected for the mating pool.
- encouraging cross-breeding amongst the sub-populations by introducing a mate selection heuristic.

More information with regards to these two changes to VEGA can be found in [40].

7.2 Vector Evaluated Particle Swarm Optimisation Algorithm

This section discusses the VEPSO algorithm. The original VEPSO algorithm is discussed in Section 7.2.1. Extensions made to the original VEPSO algorithm for this thesis are discussed in Section 7.2.2.

7.2.1 Original VEPSO Algorithm

The VEPSO algorithm, inspired by VEGA [131], was introduced by Parsopoulos *et al.* [125]. Parsopoulos *et al.* [124] compared the performance of VEGA and VEPSO and found that VEPSO outperforms VEGA.

The VEPSO algorithm consists of two layers, namely a top layer that manages the sub-swarms and a lower layer that contains the sub-swarms. At the top layer the algorithm manages the sharing of knowledge between the swarms. At the lower layer, each sub-swarm optimises its assigned objective function.

Low-level Tasks

This section discusses the task of guide update approaches that are performed by the sub-swarms, i.e. on the lower level of the VEPSO algorithm.

The search process of VEPSO is driven through local and global guides. The local guides, which are actually the personal bests (or pbests), contain information about the particles' own experience with regards to a single objective. On the other hand, the global guides, which are the global bests (or gbests), contain information obtained by a pre-defined neighbourhood of particles with regards to another objective. A knowledge sharing topology determines which objective's gbest is used.

The original VEPSO articles [124, 125] do not indicate whether Pareto-dominance was used for the guide updates. Therefore, it is assumed that the original version of VEPSO updates the guides according to the particles' fitness with regards to only one objective, i.e. the objective that the specific swarm is optimising. To solve DMOOPs, Pareto-dominance was added to the VEPSO algorithm, as discussed in Section 9.3.

Top-level Tasks

This section discusses knowledge exhange between the sub-populations that are performed at the top level of the VEPSO algorithm.

The number of sub-swarms is equal to the number of objectives of the optimisation problem and each swarm optimises only one objective function. Knowledge of best solutions is then shared with the other swarms. This shared knowledge, contained in the global guide of another swarm, is then used to update the velocity of the particles:

$$\begin{aligned} S_k.v_{ij}(t+1) &= wS_k.v_{ij}(t) + c_1 r_{1j}(t)(S_k.y_{ij}(t) - S_k.x_{ij}(t)) \\ &+ c_2 r_{2j}(t)(S_s.\hat{y}_{ij}(t) - S_k.x_{ij}(t)) \end{aligned} \tag{7.1}$$

where $k = 1, \ldots, m$ represents the index of the respective swarm, $v_{ij}(t)$ and $x_{ij}(t)$ represent the j-th dimension of the velocity and position of particle i at time t respectively, w is the inertia weight, $S_s.\hat{\mathbf{y}}_i$ is the global best of the s-th swarm, c_1 and c_2 are respecively the cognitive and social coefficients, $\mathbf{r}_1, \mathbf{r}_2 \in [0, 1]^n$, and n is the dimension of the search space.

The index, s, of the swarm from which knowledge is obtained, is selected based on a knowledge sharing topology. The original VEPSO used a ring topology where s is selected according to Equation (7.2), where

$$s = \begin{cases} M & \text{for } j = 1 \\ j-1 & \text{for } j = 2, \ldots, M \end{cases}$$

Therefore, VEPSO has two topologies, namely:

- the topology of the swarms that is used to exchange knowledge between the different swarms.
- the topology of the particles in each swarm that is used for the global guide update.

Since each swarm optimises only one objective function, an increase in the number of objective functions of a MOOP will result in a linear increase in the number of required swarms, resulting in an increase in computational complexity. To overcome this problem, Parsopoulos and Vrahatis presented a VEPSO design that enables VEPSO to be executed in parallel [124]. Each swarm's velocity update is performed in isolation from the other swarms. Before each parallel execution step, the state of knowledge being shared is recorded and the recorded knowledge is then used to update the particles' velocities.

7.2.2 Extensions to the VEPSO Algorithm

This section discusses extensions to the VEPSO algorithm. Extensions introduced at the sub-swarm level as well as the top level are presented. At the sub-swarm level the management of boundary constraint violations are introduced. At the top level of the algorithm knowledge sharing approaches between the sub-swarms and the management of an archive are introduced.

Low-level Tasks

The original VEPSO articles [124, 125] do not indicate whether boundary constraint violations are managed. Therefore, it is assumed that the original version of VEPSO does not manage the violation of boundary constraints. This section discusses an additional task performed by the sub-swarms, namely managing particles that move outside the search space during the optimisation process.

Some of the approaches that have been used to pull a particle back into the search space and proposed to be used for VEPSO [77] are:

- *Clamping*, where each particle that violates a specific boundary of the search space is placed on or close to the violated boundary of the search space [122]. Clamping is defined as:

$$\begin{aligned} \text{if } \mathbf{x}(t+1) &> \mathbf{x}_{max}, \text{ then } \mathbf{x}(t+1) = \mathbf{x}_{max} - \epsilon \\ \text{if } \mathbf{x}(t+1) &< \mathbf{x}_{min}, \text{ then } \mathbf{x}(t+1) = \mathbf{x}_{min} \end{aligned} \tag{7.2}$$

  with ϵ a very small positive number.

- *Deflection*, where the velocity's direction of the violated dimension is inverted, thereby causing a bouncing effect off the bounds. The deflection approach is defined as:

$$\begin{aligned} \text{if } x_i(t+1) > x^i_{max}, \text{ then } & x_i(t+1) = x^i_{max} - (x_i(t+1) - x^i_{max})\%(x^i_{max} - x^i_{min}) \\ & \text{and } v_i(t+1) = -v_i(t) \\ \text{if } x_i(t+1) < x^i_{min}, \text{ then } & x_i(t+1) = x^i_{min} + (x^i_{min} - x_i(t+1))\%(x^i_{max} - x^i_{min}) \\ & \text{and } v_i(t+1) = -v_i(t) \end{aligned} \tag{7.3}$$

  where x_i, x^i_{min} and x^i_{max} are the i-th dimension of $\mathbf{x}$, $\mathbf{x}_{max}$ and $\mathbf{x}_{min}$ respectively.

- *Per element re-initialisation*, where each dimension of the particle's position that violates the boundary constraint is re-initialised to a random valid value [122]. Therefore, the dimensions of the position that is valid remains the same. Per element re-initialisation is defined as:

$$\begin{aligned} \text{if } x_i(t+1) > x^i_{max}, &\quad \text{then } x_i(t+1) = rand(x^i_{min}, x^i_{max}) \\ \text{if } x_i(t+1) < x^i_{min}, &\quad \text{then } x_i(t+1) = rand(x^i_{min}, x^i_{max}) \end{aligned} \tag{7.4}$$

- *Periodic*, which is similar to the deflection approach, but the violating particle is placed near the lower boundary of the dimension if it violates the upper boundary of the dimension and vice versa [162]. The periodic approach is defined as:

$$\begin{aligned} \text{if } x_i(t+1) > x^i_{max}, \text{ then } x_i(t+1) &= x^i_{min} - (x_i(t+1) - x^i_{max}) \\ &\quad \%(x^i_{max} - x^i_{min}) \\ \text{if } x_i(t+1) < x^i_{min}, \text{ then } x_i(t+1) &= x^i_{max} - (x^i_{min} - x_i(t+1)) \\ &\quad \%(x^i_{max} - x^i_{min}) \end{aligned} \tag{7.5}$$

- *Random*, which re-initialises a particle's position to a valid position within the search space if it violates the boundaries of the search space [79, 162]. Therefore, in contrast to the per element re-initialisation approach, all dimensions are re-initialised and not only the violating dimensions. Random is defined as:

$$\begin{aligned} \text{if } \mathbf{x}(t+1) > \mathbf{x}_{max}, &\quad \text{then } \mathbf{x}(t+1) = rand(\mathbf{x}_{min}, \mathbf{x}_{max}) \\ \text{if } \mathbf{x}(t+1) < \mathbf{x}_{min}, &\quad \text{then } \mathbf{x}(t+1) = rand(\mathbf{x}_{min}, \mathbf{x}_{max}) \end{aligned} \tag{7.6}$$

- *Re-initialisation*, where a particle that violates the bounds of the search space has its position re-initialised to a valid position within the search space, its velocity set to zero, and its local guide set to the particle's new position [122].
- *Unconstrained*, where no clamping is performed and particles are free to move outside the search space. However, only valid positions are selected as the local guide of a particle.

Top-level Tasks

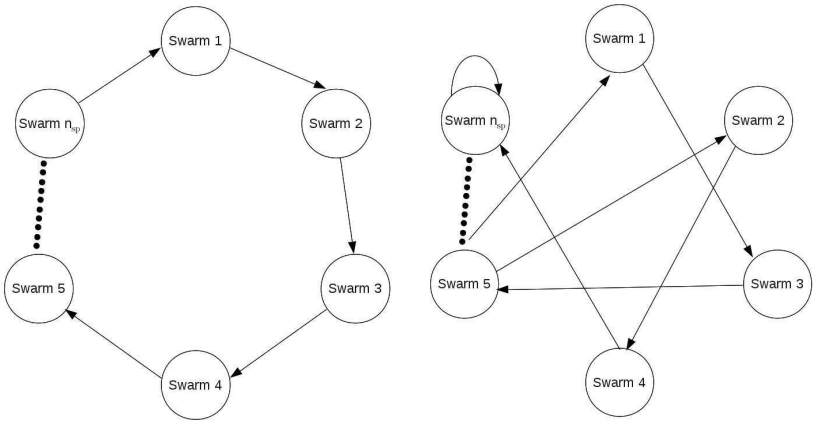
This section discusses additional knowledge exhange approaches and the task of archive management that are performed at the top level of the DVEPSO algorithm.

Knowledge Sharing

The original VEPSO algorithm used the ring topology. Recently, an alternative topology has been proposed, namely a random topology [71][1]. If a random topology is used, s is selected randomly from $[1, m]$. These two VEPSO topologies are illustrated in Figure 7.2, with a ring topology on the left and a random topology on the right. With a random topology, s can be the index of another swarm (for example swarms 1-5 in Figure 7.2

[1] Greeff is the maiden name of M. Helbig.

(b)) or from the swarm itself (for example swarm n_o in Figure 7.2 (b)), i.e. a swarm can use its own *gbest* value to update its particles' velocity. The selection of index s can be done at the beginning of an algorithm run, or after a specified number of iterations. Using a different s every now and again may guide the particles to different parts of the POF since information about another objective is provided. However, if the index of s is changed too often it may slow down the convergence of the algorithm by guiding the particles in a new search direction before optimal solutions were found.



(a) Ring Topology (b) Random Topology

Figure 7.2: Topologies of VEPSO algorithm

Inter- and Intra Swarm Speciation

According to Matthysen and Engelbrecht [116], the pbest update approach used by VEPSO, together with a ring or random topology for the gbest update, may lead to inter- and intra-swarm speciation defined below.

If the gbest is not updated often, the particles' trajectories will end near the gbest. The gbest will cause the particles to converge to a point that is a weighted average between the particle's pbest and the swarm's gbest [27, 63]. This will prevent particles from exploring the entire POF and therefore only sub-regions of the POF will be found. This problem, similar to the problem experienced by VEGA, is referred to as inter-swarm speciation.

One approach to overcome inter-swarm speciation, is to update the gbest more fre-

quently using the random swarm topology (refer to Figure 7.2 (b)). However, the random swarm topology can lead to intra-swarm speciation. Intra-swarm speciation occurs when the randomly selected swarm index results in a swarm using its own gbest frequently to update its particles' velocity. In this situation, the pbest of each particle in the swarm is updated according to only the objective function assigned to the swarm. Then, if the swarm's own gbest is used, the swarm obtains more information about its own objective function being optimised than other swarms that do not use their own gbest. This will lead to exploitation of the swarm's knowledge about its objective function, that may lead to more updates of the swarm's gbest. Even though there is the danger of intra-swarm speciation, the random swarm topology overcomes the problem of inter-swarm speciation and therefore may lead to a more diverse set of solutions being found.

Another approach that can be followed to overcome inter-swarm and intra-swarm speciation is to use information about the other objective functions when updating the pbest and gbest. In this case, each swarm still only optimises one objective function and shares knowledge with each other as discussed above. However, when updating the pbest and gbest, Pareto-dominance is used, taking into account the whole MOOP, and only non-dominated solutions are allowed to become a pbest or gbest.

Archive Management

As the VEPSO algorithm optimises the MOOP, the non-dominated solutions found so far are stored in an archive. Due to limited resources, the size of the archive is normally limited. The following approach is normally used by MOO algorithms to manage the archive:

- if a new non-dominated solution is dominated by any solution in the archive, the new solution is not added to the archive.
- if the new solution dominates any solutions in the archive, the dominated solutions are removed from the archive and the new solution is added to the archive.
- if the new solution is non-dominated with regards to all other solutions in the archive and there is space in the archive, add the new solution to the archive. However, if the archive has reached its maximum size, an approach has to be followed to determine which solutions to remove from the archive. One approach is to remove solutions from the more dense areas of the found POF [6]. This

approach ensures that a diverse set of solutions are kept.

7.3 Vector Evaluated Differential Evolution Algorithm

Inspired by VEGA [131], Parsopoulos *et al.* [123] introduced the vector evaluated differential evolution (VEDE) algorithm. VEDE is a multi-population DE algorithm. DE, introduced by Storn and Price [142], is a direct search method that utilises n_s, n-dimensional vectors. VEDE, similar to VEGA and VEPSO, assigns a single objective function to each sub-population. Therefore, the fitness of the individuals of each sub-population is evaluated against only the objective function that is assigned to that sub-population.

However, contrary to VEGA and VEPSO, VEDE knowledge sharing takes place through not only the usage of the knowledge, but through the immigration of the best individuals from one population to another.

Similar to VEPSO, an increase in the number of objective functions will require a linear increase in the number of populations of the VEDE algorithm. Therefore, Parsopoulos *et al.* [123] presented a parallel implementation of VEDE. After each generation, the best individual of each sub-population is sent to the master node. The master node then sends the migrating individual to the appropriate sub-population, i.e. the next sub-population according to the defined ring topology.

The performance of VEDE was compared against VEGA. Parsopoulos *et al.* [123] found that VEDE outperforms VEGA. However, the study also revealed that VEDE is sensitive to population size, especially when the number of individuals in a sub-population becomes small (less than 20) [123].

7.4 Hybrid Vector Evaluated Algorithm

According to Grobler and Engelbrecht [73], VEDE exploits good solutions at the cost of diversity, while VEPSO explores the search space more than VEDE, resulting in more diverse solutions. Therefore, in order to exploit the good qualities of both VEDE and VEPSO, Grober and Engelbrecht introduced the vector evaluated differential evolution

particle swarm optimisation (VEDEPSO) algorithm as a hybridisation of VEPSO and VEDE.

Similar to VEPSO and VEDE, each sub-population of VEDEPSO is assigned one objective function to optimise. VEDEPSO uses the random topology (refer to Figure 7.2 (b)) for knowledge sharing. For bi-objective problems, one sub-population is a VEPSO and the other sub-population is a VEDE [73]. However, the authors do not specify how to handle more than two sub-populations.

Grobler and Engelbrecht [73] compared VEPSO, VEDE and VEDEPSO on five MOO benchmark functions and found that VEDEPSO outperformed VEPSO and VEDE on four out of the five MOOPs. Furthermore, the results of the study indicated that VEDEPSO performed significantly better on all benchmark functions than the worst performance obtained by either VEPSO or VEDE.

7.5 Summary

This chapter discussed vector evaluated approaches for solving MOOPs. VEGA [131], introduced by Schaffer, can be seen as the first EA used to solve MOO without aggregating the objective functions. Each sub-population of VEGA optimises only one objective function. The knowledge of the various sub-populations are shared through mutation and cross-over on a mating pool consisting of individuals selected from each sub-population. One problem with VEGA is the occurrence of inter-swarm speciation, namely that it favours solutions that excel in one objective. To overcome the problem of speciation, variations to the orignal VEGA algorithm were proposed.

Inspired by VEGA, Parsopoulos *et al.* [125] introduced VEPSO. Similar to VEGA, VEPSO divides the swarm into sub-swarms and each sub-swarm only optimises one objective function. The various sub-swarms share their knowledge through the velocity update of the particles. If the standard PSO pbest update is used together with a ring topology, VEPSO may struggle with inter-swarm speciation, similar to VEGA. However, the problem of inter-swarm speciation can be overcome by using a different knowledge exchange strategy.

Another algorithm was also inspired by VEGA and introduced by Parsopoulos *et*

al. [123], namely VEDE. Similar to VEGA and VEPSO, each sub-population of VEDE optimises only one objective function. However, contrary to VEGA and VEPSO, knowledge sharing takes place through not only the usage of the knowledge, but through the immigration of the best individuals from one sub-population to another.

Since both VEDE and VEPSO have good qualities, Grobler and Engelbrecht [73] introduced an algorithm that hybridises both of these vector evaluated approaches, called VEDEPSO. Each sub-population of VEDEPSO is either a VEDE or VEPSO algorithm. The study by Grobler and Engelbrecht indicated that the hybridised algorithm outperformed the other two vector-evaluated approaches on five MOOPs.

The next chapter discusses population-based algorithms that were introduced to solve dynamic MOOPs.

Chapter 8

Population-based Dynamic Multi-objective Optimisation Algorithms

"The art of life is a constant readjustment to our surroundings."
–K. Okakaura

DMOOPs are optimisation problems with multiple objectives with at least one objective changing over time. The objectives are in conflict with one another and therefore the problem does not have a single solution, as is the case with DSOOP. A DMOOP, similar to a MOOP, has a set of trade-off solutions called the POF. Therefore, in order to solve a DMOOP, an algorithm must be able to track the changing POF over time.

This chapter discusses population-based algorithms proposed to solve DMOOPs. Section 8.1 provides an overview of DMOO algorithms that have been proposed in the literature and a summary is provided in Section 8.2.

8.1 Dynamic Multi-objective Algorithms

This section discusses algorithms that have been proposed for DMOO. Algorithms that aggregate the objective functions of the DMOOP to create a DSOOP are not considered.

All acronyms used in this section for benchmark functions and performance measures were defined in Chapters 3 and 4 respectively.

Section 8.1.1 discusses SMOO algorithms that were adapted to solve DMOOPs. New CI algorithms that were introduced for DMOO are discussed in Section 8.1.2. Section 8.1.3 discusses approaches that are used to convert a DMOOP into multiple static MOOPs (SMOOPs). Generic extensions that can be applied to any DMOO algorithm are discussed in Section 8.1.4. Section 8.1.5 discusses prediction-based approaches where knowledge of previous environments is used to predict the new POS or POF.

8.1.1 Multi-objective Optimisation Algorithms adapted for Dynamic Multi-objective Optimisation

One of the first algorithms proposed to solve DMOOPs without using the weighted sum approach to aggregate the objective functions into a DSOOP, was a hybridised minimal cost evolutionary deterministic algorithm (HMCEDA) introduced by Farina *et al.* [58]. With the hybrid algorithm an (1+1) evolution strategy (ES) is used for global optimization of the DMOOP [57]. An (1+1) ES is an EA where each iteration applies Gaussian mutation to one parent to create one offspring, i.e. a random value from a Gaussian distribution is added to each element of a parent's vector to create an offspring [9]. Once the (1+1) ES starts to converge (determined by comparing the decision variable values from two consecutive iterations), a gradient-based algorithm or a simplex Nelder Mead search algorithm [120] is used. HMCEDA was evaluated on the FDA DMOOPs. For FDA1 and FDA2, the algorithm tracked the changing POF well over time and converged quickly to the new POF after a change in the environment occurred. However, for FDA3, HMCEDA struggled to converge towards the changing POF and struggled to find a diverse set of solutions. For FDA4, the algorithm converged reasonably well to the new POF after each change in the environment. However, for FDA5 where the density of the solutions change over time, HMCEDA struggled to maintain a diverse set of solutions [58]. According to Farina *et al.* [58] the results from the study indicate that HMCEDA should use an EA with better performance, such as NSGA-II, for the global optimization.

Many algorithms used to solve SMOOPs were adapted for DMOO. Avdagić *et al.*

extended MOGA to solve DMOOPs [2]. MOGA was the first multi-objective GA introduced by Fonseca and Flemming [60] and uses Pareto-ranking. The advantages of MOGA is a simple fitness assignment and the easy application of MOGA to other types of optimisation problems (such as combinatorial optimisation problems), since niching takes place in objective space [40]. Furthermore, MOGA maintains a diverse set of solutions. One problem with averaging the fitness values for Pareto ranking is that all solutions in a specific front have the same fitness. For the front with Pareto rank 1 it does not matter that all solutions in the front have the same assigned fitness, since all the solutions are non-dominated. However, for the other fronts, when all solutions have the same assigned fitness, the search may be biased towards unwanted solutions in the search region. If a solution of a lower front (thereby being dominated by less solutions) is located in a crowded region, its niche count will be large and therefore the solution will obtain a lower average fitness than a solution of a higher front (being dominated by more solutions) in a less crowded region. This can cause the search to be biased towards the solution in the higher front, instead of the solution in the lower front [40]. MOGA was evaluated on modified DTLZ functions and its performance was measured using C_o and the HV. MOGA converged well towards POF and found a diverse set of solutions [2].

NSGA-II, one of the most successful MOO algorithms that are a benchmark for MOO research, was adapted for DMOO by Deb *et al.* [46]. A few solutions are randomly selected and re-evaluated and if there is any change in the objective values, it is assumed that a change in the environment has occured. If a change has been detected, the population is re-evaluated. To increase diversity, the following two approaches are used after a change in the environment took place:

- A percentage of the population is replaced with randomly created individuals. This approach is referred to as dynamic NSGA-II (DNSGA-II)-A. Since this approach introduces new solutions, it increases the search space that is covered by the population and may lead to improved performance in environments with severe changes [46]. However, if the environment has small changes, the new individuals may misguide the search to new areas of the search space that is far away from the current optima.
- A percentage of the population is replaced with mutated solutions of randomly

selected existing solutions. This approach is referred to as DNSGA-II-B. Since the new individuals are related to the individuals of the current population, this approach may be beneficial in environments with small changes [46]. However, in environments that change severely, the new individuals may guide the search in new areas that are far away from the new optima. This may cause the algorithm to become stuck in old optima.

Deb *et al.* [46] investigated the performance of the DNSGA-II [47, 42] algorithms on a modified version of FDA2 and measured the algorithms' performance using the HVR. The results indicated that the performance of DNSGA-II deteriorated when the frequency of change increased. Furthermore, when the number of random solutions introduced after an environmental change were increased, the performance of DNSGA-II-A deteriorated. When the number of mutated individuals added to the population after a change occurred was increased, the performance of DNSGA-II-B decreased slightly. However, DNSGA-II-B performed better than DNSGA-II-A. This may be explained by the fact that even though the shape of the POF changed from convex to concave over time, the new POF was not far from the previous POF in objective space. Furthermore, adding a small percentage of either random or mutated individuals after a change in the environment occurred, lead to better performance than not adding any new solutions at all. The study also indicated that almost any percentage of the population can be mutated and NSGA-II-B still performed well. However, with DNSGA-II-A, 20-40% of random individuals after a change lead to good performance [46].

In order to determine the efficiency of various MOO algorithms solving DMOOPs, Mehnen *et al.* [117] compared the performance of three MOEAs, namely NSGA-II [47, 42], SPEA2 (SPEA2) [170] and multiple single objective Pareto sampling (MSOPS) [86]. MSOPS is a stochastic population-based algorithm that does not incorporate Pareto-dominance. MSOPS uses a weighted min-max aggregation of the objectives and a ranking scheme according to weight vectors referred to as targets. However, contrary to conventional linear aggregation approaches, weighted min-max is capable of finding solutions on non-convex parts of the POF. The performance of these three MOEAs were compared solving eight DMOOPs, namely DSW1, DSW2, DTF and the FDA DMOOPs. The results indicated that, when simulated binary crossover (SBX) [43] was used together

with polynomial mutation (PM) [39], all MOEAs were capable of tracking the changing POS when solving the Type I DMOOPs. Furthermore, MSOPS concentrated more on the central region of the POF when solving DMOOPs with a small severity of change. When DE variation operators were used instead of SBX and PM, the population's diversity collapsed and then the operators were no longer capable of creating new solutions. Therefore, the MOEAs got stuck. None of the MOEAs were able to track the changing POF when the environment changed at every iteration. However, MOEAs using SBX and PM converged towards the POF when the environment changed every 25 generations or less frequent than every 25 iterations. NSGA-II converged faster than MSOPS, which in turn converged faster than SPEA2. For FDA4 and FDA5, MSOPS converged quicker towards the POF. NSGA-II and SPEA2 required more function evaluations in order to converge towards the POF. Both NSGA-II and SPEA2 experienced a decrease in selection pressure, since using the Pareto-dominance relation resulted in a number of incomparable solutions. Once again, when optimising FDA4 and FDA5, MSOPS concentrated on certain parts of the POF, where as NSGA-II and SPEA2 obtained a diverse set of solutions. Similar trends were observed with FDA3 and FDA5 where the density of the solutions changed over time. Therefore, Mehnen *et al.* [117] concluded that if a population contains many incomparable individuals, the Pareto dominance based algorithms cannot guarantee further convergence. For FDA2 the POF changes in shape over time, from a convex POF to a non-convex POF. For DTF, the POF changes structure over time, since the number of continuous sections of the disconnected POF vary over time. MSOPS struggled to adapt to a change in either the shape or structure of the POF over time. NSGA-II obtained the best performance for FDA2. On the other hand, SPEA performed well when solving DTF.

An artificial immune system (AIS) algorithm, the clonal-selection algorithm for DMOO (CSADMO), was introduced by Shang *et al.* [135]. CSADMO uses clonal selection [14], a self-adaptive (dynamic) process of the immune system. Non-uniform mutation [118] is used, with the search space being searched uniformly with large jumps during the first few iterations, but during the later generations the search is more local with small jumps. Furthermore, crowding distance [42] is used to increase the diversity of the solutions. The performance of CSADMO was compared against HMCEDA of Farina *et al.* [58] using

FDA3 and FDA5. However, the authors do not mention which frequency (τ_t) and severity of change (n_t) were used for the DMOOPs. The results indicate that both algorithms performed well with regards to convergence and diversity. However, CSADMO outperformed HMCEDA with regards to both convergence and diversity on both DMOOPs. Convergence was measured using GD in the objective space, but diversity was only determined using a graphical representation of the found POFs. Furthermore, the authors used the results of HMCEDA directly from [58]. This may lead to misleading results, since the same set of sampling points for the true POF is not used for the calculation of the GD values for both algorithms.

Zhang [163] also proposed an AIS, MOIA, to solve DMOOPs. Immune operators were proposed to enable the algorithm to adapt to dynamic environments. Furthermore, Zhang proposed an environment recognition rule that is based on previous environmental information. The environment recognition rule classifies the current environment as either a new, similar or an identical environment in comparison to the previous environment. This classification is used to determine which operators to use. MOIA was compared against NSGA-II and SPEA2 using FDA2, FDA4 and a real-world problem. The algorithms' performance was measured using the C-metric and Schott's Spacing measure. The results indicate that NSGA-II outperformed SPEA2 for FDA2 and FDA4, but was outperformed by SPEA2 when solving the real-world problem. MOIA outperformed NSGA-II and SPEA2 on all three problems.

An orthogonal multi-objective EA for DMOO (DOMOEA) [160] was introduced by Zeng *et al.* DOMEA uses two types of cross-over operations, namely orthogonal cross-over that is based on orthogonal design [160] and linear cross-over [158]. One of these cross-over operators are randomly selected and applied to two randomly selected parents to produce offspring. If the number of non-dominated solutions are more than the population size, a clustering technique is used to select individuals for the new population. DOMOEA was evaluated on FDA1, FDA2 and FDA3, with $n_t = 10$ and $\tau_t = 150$. The algorithm's performance was measured using GD and Spread as defined by Deb [42]. The results indicate that DOMOEA successfully converged to the POF and found a diverse set of solutions.

Zhen [165] also proposed a MOEA adapted for DMOO (DMOEA). When a change

is detected in the environment, hypermutation [28] is used to preserve a certain number of elitist individuals, while the rest of the individuals are replaced by randomly created new individuals. Furthermore, DMOEA uses geometrical Pareto-selection to manage the archive. This approach selects an auxiliary point that is far away from POF^*. Each solution in POF^* is connected to the auxiliary point, enabling the calculation of a slope. If two solutions have the same slope, the solution furthest away from the auxiliary point has the best fitness (assuming minimisation). If a new solution is considered to be stored in the archive, the new solution is only compared against the solutions that are located in the same slope region as the new solution. If the new solution has a larger distance to the auxiliary point than one of the solutions in the archive that it is compared against, the new solution replaces the other solution. DMOEA was evaluated on FDA1, modified FDA2, modified FDA3, FDA4 and FDA5 using the HV. For FDA1 to FDA3, $\tau_t = 2000$ and for FDA4 and FDA5, $\tau_t = 5000$. Therefore, the environment changed only every 2000 or 5000 generations. The results indicate that DMOEA successfully tracked the changing POF and found a diverse set of solutions.

A multi-strategy ensemble MOEA, referred to as MS-MOEA, was introduced by Wang and Li [156] to solve DMOOPs. If a change is detected, each individual is either re-initialised to a random new position or re-initialised by selecting a random parent and adding values of a Gaussian distribution to all variables of the parent to create the new individual. During the search, two possible combinations of operators are used to create offspring, namely: (i) SBX and polynomial mutation, or (ii) DE operators. MS-MOEA was compared against an improved version of NSGA-II (INSGA-II) [161], FH-MOEA [154] and MS-MOEADE [156] using the FDA1, FDA2, FDA3, DMZDT functions and WYL. The performance of the algorithms were measured using IGD and the HV. The results indicate that MS-MOEA outperformed the other algorithms on FDA1 and FDA2, and obtained the second highest performance on FDA3. Furthermore, MS-MOEA outperformed the other algorithms on all DMZDT functions and WYL. Only on DMZDT4 with the slowest changing environment setting did MS-MOEA obtain the second highest rank, with INSGA-II obtaining the best performance.

Chen *et al.* [23] introduced the individual diversity multi-objective optimization evolutionary algorithm (IDMOEA) that uses diversity as an additional objective when solv-

ing DMOOPs. The first step of IDMOEA entails checking whether a change in the environment has occurred. If an environment change is detected, a new population is created and the best individuals of the population and the archive are selected for the new population. During the next step of the algorithm crossover is performed on parents (selected through binary tournament selection) to produce offspring, mutation is performed on the offspring and the new population is selected. Finally, the archive is updated by adding non-dominated individuals of the population to the archive. If there are more non-dominated individuals than the size of the archive, the individuals with a better diversity (calculated according to the diversity measure that is used as an additional objective) are added to the archive. IDMOEA was evaluated on FDA1 and FDA5. The algorithm's performance was measured using GD and entropy. The results indicate that the algorithm showed good convergence and maintained a diverse set of solutions.

Recently dynamic multi-objective gradient search (MO-EGS) [65] was adapted for DMOO by Koo *et al.* [100]. The new algorithm is called dynamic MO-EGS (dMO-EGS). A change is detected by re-evaluating solutions in the archive. When a change in the environment occurs, information with regards to the outdated archive (i.e. the archive containing solutions for the previous environment) is represented by the archive centroid and the variance of the archive centroid. The archive centroid is calculated by summing the decision vectors of each solution in the archive and then dividing by the number of archive solutions. A memory item that consists of the centroid and centroid variance is added to the external memory. If the external memory is full, the oldest memory item is replaced.

After a change in the environment has been detected, a truncation operator is used to remove solutions from the outdated archive in such a way that the most diverse portion of solutions are retained. Mutation is then applied to the retained solutions as a form of hypermutation to increase the diversity of the solutions. After mutation has been applied, the retained solutions of the archive are re-evaluated. In addition to mutation, in order to increase the exploration ability of dMO-EGS after a change has occurred, the mutation step size is reset.

dMO-EGS uses a weighted sum average of the objectives to calculate an individual's fitness during the optimisation process. The weights are created randomly and nor-

malised in such a way that the sum of the weights are equal to one. However, Pareto dominance is used to determine which solutions are stored in the archive.

The performance of dMO-EGS was compared against two other DMOO algorithms, namely dCCEA [147] and dPAES [98] (a dynamic version of PAES). The same change response strategies are implemented in dCCEA and dPAES, namely storing and retrieving of memory items, hypermutation and re-evaluation of archive solutions. The three algorithms were tested against four benchmark functions, namely FDA1, FDA3, DIMP1 and DIMP2. For FDA1 and FDA3, both dynamic CCEA (dCCEA) and dMO-EGS performed signficantly better than dPAES with regards to both accuracy of the found POS and the spread of solutions. All three MOEAs had difficulty finding a diverse set of solutions when solving DIMP1. However, dMO-EGS obtained better results for DIMP1 than both dCCEA and dPAES. dMO-EGS obtained the best performance for DIMP2 when the landscape changes were not so severe and the frequency of change was low. When the environment changed severly, dCCEA outperformed dMO-EGS.

8.1.2 New Computational Intelligence Algorithms

A few algorithms introduced for DMOO were not merely adapted SMOO algorithms, but new types of CI algorithms. Amato and Farina [1] proposed an ALife-inspired algorithm to solve DMOOPs. The individuals of the algorithm are coded strings, similar to a GA, but the operators are based on individuals interacting in a population, i.e. the individuals can reproduce, meet each other or fight with each other. Unlike other EAs where selection takes place for the cross-over operator, no selection takes place. Each individual has the same probability to meet another individual. If a meeting does take place, another individual is randomly selected for the meeting. During the meeting the individuals either reproduce or fight. If reproduction occurs, two offspring are produced and added to the population. However, if the two individuals fight, the loser of the fight (the individual that Pareto dominates the other, or if both individuals are non-dominated, the individual with the most neighbours in a specified neighbourhood) is removed from the population. Due to the operators that are performed on the individuals, the population size varies. Since no selection is required, fitness evaluations are only performed when a meeting occurs between individuals, and therefore not necessarily for each individual at

each iteration. For very complex problems, this may lead to a lower computational cost. Furthermore, the algorithm does not have to check for a change in the environment, but automatically tracks the changing POF. However, a major problem with the algorithm is slow convergence and therefore the algorithm will struggle to track the changing POF in a fast changing environment.

Goh and Tan [147] introduced a modified version of CCEA (refer to Section 6.4) to solve DMOOPs. The cooperative EA, CCEA, is extended to incorporate a competitive mechanism and this extended algorithm is referred to as the competitive-cooperative evolutionary algorithm (COEA) [67]. COEA sub-populations optimise only one decision variable. The competitive mechanism is used to determine the most suitable sub-population for each decision variable. The selected sub-population is then used during the cooperative phase of the algorithm to optimise that specific decision variable. The competitive mechanism is implemented as follows: for each decision variable, a representative solution is selected from the associated sub-population and placed in a competition pool along with the competitors that are selected from the other sub-populations. One competitor is selected randomly from each competing sub-population. Once the competition pool has been created, the competitive process is performed on the competition pool. Through the competitive process, the winning sub-population is determined for the specific decision variable and then assigned to optimise the specific decision variable.

Goh and Tan [67] extended the COEA algorithm to detect and respond to changes in the environment. This modified DMOO version of CCEA is called dynamic COEA (dCOEA). A change in the environment is detected by re-evaluating a fixed number of solutions in the archive and checking whether there is a difference between a solution's previous value and the re-evaluated value. If there is a difference, a change is detected and the competitive mechanism is started. Using the competitive mechanism after a change occurs enables the algorithm to apply existing information of the various sub-populations to the new environment. Furthermore, diversity is introduced into the sub-populations through the competitive mechanism, instead of using other diversity mechanisms, such as hypermutation. dCOEA increases diversity through the competitive mechanism by introducing a set of stochastic solutions together with the competitors of other sub-populations into the competition pool. If a stochastic solution emerges as the winning

solution after the competitive process, the associated sub-population is re-initialised in the region that the winning solution has been sampled from. This approach ensures that diversity is only introduced into a sub-population if the added diversity produces better solutions than the sub-population's current individuals. The ratio between the stochastic solutions and the competitors are determined by a pre-defined parameter, SC_{ratio}. dCOEA stores the non-dominated solutions that have been found so far in an external population, also referred to as temporal memory, in addition to an archive. When a change occurs, a fixed number of solutions, R_{size}, of the archive are stored in the temporal memory, before all solutions are removed from the archive. If the temporal memory is full, the oldest set of R_{size} is removed to make place for newer solutions. To ensure that the R_{size} solutions of the archive that are now in the temporal memory do not misguide the optimisation process, these solutions are not re-inserted into the sub-populations during the generation immediately after an environmental change. The value of R_{size} determines how many solutions of each environment are stored in the temporal memory. A small R_{size} value causes that a smaller portion of solutions obtained from more environments are stored in the temporal memory. In this case dCOEA has access to information that ranges across various landscapes.

The performance of dCOEA was compared against adapted versions of a basic MOEA and CCEA [147]. Random restart and temporal memory were incorporated into MOEA and CCEA. dCOEA and the adapted MOEA and CCEA algorithms (referred to as dynamic MOEA (dMOEA) and dCCEA respectively) were evaluated on FDA1, dMOP1, dMOP2 and dMOP3. The study indicates that dCOEA outperformed dMOEA and dCCEA with regards to convergence to the true POF and found a diverse set of solutions for FDA1 and dMOP1. Furthermore, as can be expected, all three algorithms obtained better convergence and diversity for less frequent landscape changes, since the algorithms then had a longer time to converge to the POF before a change in the environment occurred. For high change frequencies where the environment changed every five or ten iterations, dCCEA outperformed dCOEA with regards to convergence to the true POF. However, dCOEA outperformed dMOEA and dCCEA with regards to convergence and diversity when the environment changed less frequently or when the environment changes were less severe. Further investigation revealed that a lower SC_{ratio}

value improved dCOEA's performance in environments that changed frequently. With regards to dMOP3, where the spread of solutions of the true POF changes over time, dCOEA outperformed dMOEA and dCCEA with regards to both convergence and diversity. However, all three algorithms struggled to obtain a diverse set of solutions when the environment changed frequently, i.e. every five or ten iterations.

Recently, Huan *et al.* [84] introduced a DMOO algorithm based on P systems or membrane computing [64], referred to as DMOAP. A membrane structure consists of a main membrane that contains other membranes. A P system is a membrane structure where the membranes contain objects that can evolve. DMOAP consists of a skin membrane and a m_{mid} membrane. The m_{mid} membrane contains $m+1$ membranes or sub-systems. m of these sub-systems optimise only one objective function and the other sub-system optimises all objective functions simultaneously. Each sub-system has its own sub-population and works similar to a single-objective EA. The m_{mid} membrane collects the chromosomes that are produced by the sub-systems and sends the non-dominated solutions to the skin membrane. The skin membrane then selects the best trade-off solution(s). DMOAP is tested on a control problem of a time-varying unstable plant. The results indicate that DMOAP effectively solved the control problem.

Many MOEAs were adapted for DMOO. However, only a few PSO-based algorithms were proposed to solve DMOOPs. Lechuga [102] proposed the first PSO-based DMOO algorithm by extending MOPSO [33] (refer to Section 6.3) for DMOO. To detect changes in the environment, sentry particles [22] are used. A specified number of particles, called sentry particles, are randomly selected and re-evaluated after the algorithm performed the specific iteration, but before the next iteration starts. If the sentry particle's fitness value differs after re-evaluation with more than a specified value, the swarm is notified that a change in the environment has occurred. Once a change has been detected, one of the following approaches are used to react to the change:

- The pbest of the particle is set to its current position if the current position dominates the pbest, referred to as response$_a$.
- The pbest of the particle is set to its current position, referred to as response$_b$.

The performance of the modified MOPSO algorithm using each of the two response approaches is measured using GD and the variance of GD in the objective space. Two

DMOOPs were used to test the modified MOPSO, namely FDA1 and a modified version of FDA2. When solving FDA1, for $n_t = 10$ and $\tau_t = 10$ response$_b$ obtained the best average GD values with a slightly higher standard deviation of GD values. However, for $n_t = 10$ and $\tau_t = 50$, the performance of both approaches were very similar. When solving the modified FDA2 with $n_t = 10$ and $\tau_t = 10$, response$_a$ obtained a better average GD value and a better standard deviation GD value. However, for $n_t = 10$ and $\tau_t = 50$, response$_b$ obtained a better average GD value and a better standard deviation GD value. Therefore, in fast changing environments response$_a$ outperformed response$_b$, but in slower changing environments both approaches performed well. The modified MOPSO algorithm using response$_b$ was compared against the original NSGA-II algorithm that was not adapted for DMOO. FDA1 and the modified FDA2 DMOOPs were used with $n_t = 10$ and $\tau_t = 50$. The results indicate that the modified MOPSO algorithm outperformed NSGA-II on both functions and that there was a statistical significant difference in the GD values with a confidence level of 5% using t-tests [102].

The next PSO algorithm proposed for DMOO was the maximinPSO algorithm [107] extended by Li *et al.* to solve DMOOPs. MaximinPSO evaluates the fitness of an individual with a maximin fitness function that takes into account non-dominance and diversity [107]. When solving DMOOPs, maximinPSO does not test for changes in the environment. At each iteration the particle's pbest is set to its current position. Therefore, only the gbest influences the particle's velocity and new position, i.e. the social-only PSO model is used. According to the empirical results of Kennedy [93], the social-only PSO is more efficient than both the full PSO that uses the pbest and gbest to update the particle's velocity and the cognitive-only PSO that uses only the pbest to update the particle's velocity. The social-only PSO exploits more and therefore converges quicker. However, the social-only PSO is susceptible to local optima [93].

The adapted maximinPSO was tested against the DMOOPs introduced by Jin *et al.* [90]. HVR and IGD were used to measure the performance of the algorithm. The adapted maximinPSO was compared against the original maximinPSO that does not reset the particle's pbest to its current position. The results indicate that for fast changing environments the adapted maximinPSO outperformed the original maximinPSO algorithm.

8.1.3 Converting Dynamic Multi-objective Optimisation Problems into Single Multi-objective Optimisation Problems

Another approach to solve DMOOPs is to convert the DMOOP into various SMOOPs. Wang and Dang [153] proposed a new approach where the time period of the DMOOP is divided into several smaller time intervals. For each time interval a SOOPs is defined by limiting the DMOOP to the specific time interval. Each SMOOP is then transformed into a new two-objective optimisation problem where one objective is to increase the diversity of the solutions and the other objective is to increase the quality of the found non-dominanted solutions. Uniform cross-over is used to avoid cross-over between two individuals that are in close proximity to each other during the first few iterations of the algorithm run. The proposed EA was compared against NSGA-II using FDA1, FDA2 and FDA3. The algorithms' performance was measured using the C-metric and the U-measure. The results indicate that the proposed algorithm successfully tracked the changing POF over time. Furthermore, the proposed algorithm outperformed NSGA-II with regards to both performance measures.

A memetic algorithm (MA) [119] that incorporates a sequential quadradic programming (SQP) solver was proposed by Isaacs *et al.* [87, 88]. Change detection is done by evaluating randomly selected individuals, and if their fitness has changed it indicates a change in the environment. Where EAs use genetic operators (such as cross-over and mutation) to evolve the individuals, the MA uses SQP. The population is randomly initialised within the search space of the DMOOP. The extreme solutions of the POF is calculated by minimising each objective function seperately as a SOOP. Then the range of each objective is sub-divided into small intervals of a specified size. For each of these intervals, a SOOP is solved, where the objective's range is within the smaller interval. This process of solving the DMOOP through a series of SOOPs are referred to as an orthogonal epsilon-constrained formulation of the DMOOP. For each of the SOOPs, an individual of the population is randomly selected as the starting point. The MA was tested against FDA1 and a modified version of FDA2. As expected, an increase in the number of SQP iterations resulted in the found solutions being closer to the POF. However, increasing the number of SQP iterations requires more function evaluations [87]. The MA was also applied to train a neural network for an unmanned aerial vehicle prob-

lem. The results indicate that the MA trained the neural network faster than the online Levenberg-Marquardt algorithm [87].

In another study [88], the MA was compared with an EA that incorporates simulated binary cross-over (SBX) and polonomial mutation. When a change in the environment occurs, the EA re-starts, i.e. all solutions in the population are replaced by new solutions with randomly created positions in the search space. MA and the EA were also tested against FDA1 and a modified version of FDA2. The results indicate that MA obtained a better accuracy than the EA for both functions. The MA framework of Isaacs *et al.* was extended by Ray *et al.* [127]. In the extended version, the MA framework is extended so that either SQP or a sub-EA is used. The sub-EA evolves the population of the MA and uses SBX and polonomial mutation. The performance of the MA and sub-EA versions were compared using FDA1 and a modified version of FDA2, with $n_t = 10$ and $\tau_t = 5$. The results indicate that MA consistently obtained a better accuracy (measured using GD in objective space) than the sub-EA version [127].

Liu and Wang [111, 112] proposed a modified MOEA to solve DMOOPs, referred to as DMEA. The total time of the DMOOP is divided into smaller time intervals. For each time interval the DMOOP is considered as a static MOOP. For each of the MOOPs, an EA is used to optimise the problem. If a change in the environment occurs, the algorithm re-starts by creating a new initial population.

DMEA was tested against two DMOOPs. No performance measures were used, but it did seem as though DMEA was tracking the changing POF according to the POF plots [111]. In another study, DMEA was tested against four DMOOPs. Once again, no performance measures were used and only graphs of the approximated POF were provided. Even though the authors claim that, according to the POF plots DMEA successfully solved the DMOOPs, this is not the case. Closer inspection of the POF plots for DMOOP G3 (FDA2 of Farina *et al.*) reveals that the algorithm lost track of the changing POF.

8.1.4 Generic Extensions for Dynamic Multi-objective Optimisation Algorithms

Various contributions were made to improve dynamic MOEA (DMOEA) algorithms. These contributions are more generic in nature and can therefore easily be incorporated into various DMOO algorithms. Guan *et al.* [74] proposed an inheritance strategy for MOEAs solving DMOOPs. When a change in the environment occurs, the MOEA selects good performing individuals according to the new objectives, and then optimises only the selected individuals until the next change occurs. The inheritance strategy was incoporated into three MOEAs, namely PAES [98], SPEA [172] and NSGA-II [47, 42]. After a change in the environment occurs, PAES re-evaluates the solutions in the archive with regards to the new objectives, the dominated solutions are removed, and only the non-dominated solutions survive. Similar to PAES, SPEA re-evaluates the solutions in the external population and only the non-dominated solutions survive. However, since NSGA-II does not use an archive or an external population to store non-dominated solutions, the whole population is re-evaluated after a change occurs. The whole population is then re-sorted based on Pareto-ranking according to the new objectives. To evaluate the influence of the inheritance strategy on the performance of the MOEAs, the performance of each MOEA without the inheritance strategy was compared against the same MOEA incorporating the inheritance strategy [74]. The MOEAs were tested against three DMOOPs where objective replacement was used to adapt the MOOPs to DMOOPs. The study revealed that the MOEAs with the inheritance strategy found more non-dominated solutions than the MOEAs without an inheritance strategy. MOEAs with the inheritance strategy also converged closer to the true POF than MOEAs without the inheritance strategy. Furthermore, MOEAs with the inheritance strategy found solutions with either a similar or better distribution than MOEAs without the inheritance strategy.

Four re-initialisation strategies that can be applied to any DMOEA when a change in the environment is detected, was proposed by Zhou *et al.* [166]. The four strategies are:

- Randomly re-initialise all new solutions within the bounds of the search space (RND).

- Add Gaussian noise, based on the last time window, to each individual (VAR).
- Sample solutions around predicted locations of the POS (PRE). The last two time windows are used to predict the next location of the POS.
- Create half of the solutions using PRE and the other half using VAR (V&P).

The effectiveness of these four strategies were evaluated on FDA1 and ZJZ. The algorithms' performance were evaluated using GD in decision space and HVD. The results indicate that RND did not perform well, since all previous knowledge were removed by randomly re-initialising all individuals. For FDA1, PRE obtained the best performance independent of the time window size, V&P performed well, VAR performed poorly and RND performed really bad. For ZJZ, the results depended on the time window size. With a time window size of 500, VAR and V&P performed similarly. However, when the time window size increased, V&P and PRE outperformed the other approaches.

When an algorithm solves DMOOPs, one of the issues that should be addressed is how to re-use past knowledge of the found POS when the environment changes. Wang and Li [155] proposed four memory-based approaches, namely:

- a restart scheme, where all individuals are re-initialised to random positions within the search space.
- an explicit memory scheme, where each individual is replaced by either a randomly created solution within the search space or by re-evaluating a randomly selected solution from the archive.
- a local search memory scheme, where each individual is replaced by either a randomly created solution within the search space or by re-evaluating a randomly selected solution from the archive and then performing a local search on the re-evaluated solution.
- a hybrid memory scheme, where each individual is replaced by either a randomly created solution within the search space, or by either re-evaluating a randomly selected solution from the archive or by re-evaluating a randomly selected solution from the archive and then performing a local search on the re-evaluated solution.

These memory schemes were incorporated into a modified NSGA-II algorithm (INSGA-II) that uses an archive. A GA-DE operator is introduced to create offspring, utilising the fast convergence of GAs and DEs' ability to find diverse solutions.

The various approaches were evaluated using FDA1, the DMZDT functions and WYL. The algorithms' performance were measured using IGD in objective space. The results indicate that the explicit, local search and hybrid memory schemes improved INSGA-II's performance. However, as the frequency of changes increased, the effectiveness of the memory schemes decreased. The local search memory scheme was more robust than the other memory schemes. Furthermore, the explicit memory scheme experienced a loss in diversity when the change frequency decreased.

With EAs, immigration schemes enable insertion of new information into the existing genetic pool of the population and therefore increases the diversity of the population. Azevedo and Araújo [3] proposed a new immigration scheme for EAs to solve DMOOPs, called the generalised immigrants-based diversity generator (gIDG). The immigration scheme was incorporated into NSGA-II [47, 42] to investigate whether the new immigration scheme leads to better performance when solving DMOOPs. gIDG is incoporated into NSGA-II in the following way: a specified number of worst individuals, n_w, are replaced with generated immigrants after cross-over and mutation, but before selection of the population for the next generation. This ensures that the immigrants do not influence the generation of offspring, but enable the immigrants to compete with the newly generated offspring for survival. The initial results indicate that a higher quality of front one solutions (non-dominated solutions with Pareto rank one) was maintained over time if the immigrants that were introduced consisted of a mixture of elite-based and random immigrants [3]. Elite-based immigrants are generated by first sorting the population according to dominance and then selecting the first n_w individuals from the sorted population to create an elite population. Individuals of the elite population are then randomly selected and mutated to create the elite-based immigrants. Random-based immigrants are generated by randomly sampling n_w solutions from a probability distribution function. Experiments were conducted using FDA1, FDA2 and a modified version of dMOP3. The algorithm's performance was measured against the offline HV. The results indicated that NSGA-II with gIDG obtained statistically significantly better HV values than NSGA-II without any immigration scheme, NSGA-II with only random-based immigrants, and NSGA-II with only elite-based immigrants [3].

An important aspect when solving DMOOPs is deciding which solution to use. Deb *et*

al. [46] highlighted the need for automated decision making when solving DMOOPs, since a solution has to be selected and implemented before an environment change occurs. Deb *et al.* used a utility measure to automate the decision making process. Another approach is to enable a decision maker to interactively and easily indicate his/her preference [113]. Therefore, Liu *et al.* [113] introduced an artificial immune system algorithm, called the sphere-dominance preference immune-inspired algorithm (SPIA). SPIA introduces a new concept called sphere-dominance that enables a decision maker to interactively define his/her preferences with regards to the solutions of the DMOOP. A sphere (or multiple spheres) with a reference point and a specified radius is defined by the decision maker. If a solution's distance to the reference point is less than the specified radius, it belongs to the sphere. Sphere-dominance then replaces the normal Pareto-dominance relation and is defined as follows:

- a solution x_a sphere-dominates a solution x_b if x_a is a member and x_b is not a member of the sphere respectively.
- a solution x_a sphere-dominates a solution x_b if both solutions are members of the sphere and x_a Pareto-dominates x_b.

This new dominance relation drives the search towards the decision maker's desired area of the found POF. Two hypermutation methods are used, namely Gaussian hypermutation and predictive hypermutation. With predictive hypermutation, a time series is developed for each antibody in the population and the new antibodies are predicted using a linear model. A progress rate that measures the improvement in the antibody quality is calculated. Then, based on the progress rate, one of the hypermuation methods are selected. The performance of SPIA was evaluated using FDA3 and FDA4 with $n_t = 10$ and $\tau_t = 50$. The convergence of SPIA was measured with GD in the objective space. For FDA3, SPIA performed well with regards to GD when searching for the whole POF, one preference region in the middle of the POF, two preference regions at the edges of the POF, and two preference regions away from the edges of the POF. When the decision maker defined two preference regions away from the edges of the POF, SPIA struggled during the first 200 generations to converge towards these regions, but then converged very well between generations 200-500. For FDA4, SPIA performed well with regards to GD when searching for the whole POF and one preference region. However, with two

preference regions SPIA did sometimes struggle to find many solutions and to converge towards the two preference regions. However, since the spread of solutions of FDA5 changes over time, finding solutions in two preference areas of the POF is a difficult task. One problem with the sphere preference approach is that the reference point has to change as the POF changes over time. Therefore, the algorithm enables the decision maker to interactively change his/her preference during the optimisation process.

Many real-world problems are computationally expensive to solve. However, when solving DMOOPs, parallel processing may speed up the execution time of the algorithm. Therefore, Cámara *et al.* [17] introduced a parallel algorithm approach that can be applied to any MOEA. The parallel algorithm has a master process that subdivides the population into sub-populations of equal size and then sends a sub-population to each worker process. Every worker process optimises the MOOP for a fixed number of iterations in the worker process's assigned search space using the assigned MOEA and keeps only the non-dominated solutions. After a fixed number of iterations, each worker process sends the found non-dominated solutions to the master process. The master process adds all non-dominated solutions obtained from the worker processes into a new population and then runs a MOEA on the new population for a specified number of iterations. After the specified number of iterations the master subdivides the population into sub-populations and the whole process is then repeated until the stopping condition has been reached. Cámara *et al.* compared the speedup obtained by the parallel approach by applying the approach to four MOEAs, namely the single front genetic algorithm (SFGA) [20], SFGA2 [20], SPEA2 [170] and NSGA-II [47]. The algorithms were tested against FDA1, $FDA2_{Camara}$ and $FDA3_{Camara}$. The algorithms' performance were measured against the HV, the maximum HV, *acc*, *stab* and *reac*. Using more than one worker processes lead to better performance by the algorithms. Furthermore, the results indicate a small computational time decrease for SFGA and SFGA2, and a huge decrease in computational time for NSGA-II and SPEA2.

8.1.5 Prediction-based Approaches

When solving real-world MOOPs and DMOOPs, function evaluations can be computationally expensive. Therefore, techniques were developed that use prediction to de-

crease the number of function evaluations without decreasing the quality of found solutions [145]. Hatzakis and Wallace [76] introduced a hybrid algorithm that incorporates a forecasting technique with an evolutionary algorithm, referred to as the dynamic queuing multi-objective optimizer (D-QMOO). D-QMOO is an adaptation of the queuing multi-objective optimizer (QMOO) algorithm proposed by Leyland [104] to solve MOOPs. D-QMOO is a steady-state clustering MOEA. Each cluster's population consists of two sub-populations, referred to as the *front* and the *cruft*. The front consists of only non-dominated solutions of Pareto-rank one and the cruft only has dominated solutions of Pareto-rank greater than one. The task of the individuals in the front is to converge to the current true POF. In contrast, the goal of the individuals in the cruft is to increase diversity to enable the algorithm to search and discover a new optimum. If the front has reached its maximum size, individuals are eliminated in such a way that the maximum HV is preserved. D-QMOO preserves elitism by only adding non-dominated individuals to the front. Any dominated individual can be added to the cruft and if the cruft's maximum size has been reached, individuals are removed. Individuals are eliminated from the cruft based on either their age (where the oldest individuals are removed) or crowding in the decision space (individuals in more crowded regions in the decision space are removed).

If D-QMOO incorporates forecasting, a third sub-population (referred to as the *prediction set*) is added when a change in the environment occurs. Forecasting is done using stochastic time series models. The input to the forecasting model is a time series of a specified number of selected points on the POF during the previous time steps. Individuals are then placed on the predicted location of each anchor point and these individuals are then added to the prediction set. The individuals of the prediction set are absorbed by either the front or the cuft during the next step of the optimisation process according to their fitness. Therefore, the prediction set only exists directly after a change in the environment occurs. If the forecasting is accurate, individuals of the prediction set are placed in close vacinity of the new optima, leading to faster convergence towards the new POF. However, the accuracy of the forecasting depends on the accuracy of D-QMOO. If D-QMOO did not converge properly to the true POF in the previous time-steps, the input into the forecasting model will lead to errors in the prediction of the location of

the anchor points. Furthermore, even if the input into the forecasting model is correct, the predicted location of the anchor points may be inaccurate. Therefore, only a small number of individuals should be added to the prediction set. D-QMOO was evaluated on FDA1. The results indicate that the feed-forward prediction strategy lead to better convergence when the environment changed frequently, but did not really improve the performance when the change frequency was low [76].

DMEA [111, 112] is extended by Liu to incorporate an estimation of the next generation's POS [110] to develop the CDDMEA algorithm. The new POS is estimated by calculating the core of the POS at various time steps. The core of a POS is the average solution of the POS, i.e. the mean is calculated for each dimension of the solutions. The difference between the core solutions at time $t-1$ and time $t-2$ is then added to a solution at time t to estimate the solution at time $t+1$. CDDMEA was compared against DNSGA-II-A [46] using a DMOOP of Jin and Sendhoff and FDA2. The performance of the algorithms were measured using the U-measure. The results indicate that CDDMEA outperformed DNSGA-II on both DMOOPs. It should be noted that this prediction approach is based on the POS. Therefore, errors in previously found POSs may cause the algorithm to lose track of the changing POF or POS.

Talukder *et al.* [145] introduced a variation operator for MOEAs and Talukder and Kirley [96] extended the variation operator for DMOEAs. The variation operator approximates the next POF and then inversely maps the POS from the approximated POF using integral transformation [96]. This mapping procedure replaces the standard cross-over and mutation used in MOEAs. Fourier transformation is used as the integral transformation and therefore the dimensions of the input (objective function values) and output values (decision variable values) should be the same. However, with DMOOPs the dimension of the decision variables is usually not the same as the number of objective functions. In order to overcome this problem, each of the design variables and objective function values are considered as seperate input/output values. The variation operator is incorporated into the NSGA-II algorithm after non-dominated sorting to create new individuals from the next predicted POF. The newly created individuals are then combined with the parent population and the remainder of the NSGA-II algorithm steps are performed as usual. The modified NSGA-II algorithm was compared against DNSGA-

II-A [46] using FDA2 and modified versions of FDA3 and FDA5. The performance of the algorithms were measured using the HVR, but with the POF found by DNSGA-II-A used as the approximated front and the POF found by Talukder and Kirley's modified NSGA-II (NSGA-II-TK) used as the known POF. Therefore, an HVR value at time t greater than one indicates that at time t NSGA-II-TK performed better than DNSGA-II-A and vice versa. For FDA2, NSGA-II-TK outperformed DNSGA-II-A between generations 150 and 250, and for FDA3 NSGA-II-TK outperformed DNSGA-II-A for most generations. For FDA5, NSGA-II-TK outperformed DNSGA-II-A for all generations. However, it should be noted that the value with which the POF values decrease between various predicted POFs is problem dependent. Furthermore, the variation operator can only be used for Type II and Type III DMOOPs where the POF changes over time. The variation operator also requires that the population is sorted into non-dominated fronts (Pareto-ranking) and that there is more than one front. Therefore, the variation operator cannot be used by algorithms that store solutions in an archive without ranking solutions in various fronts. An advantage of this predictive approach is that the objective functions do not have to be differentiable.

D-NSGA-II was adapted by Roy and Mehnen [129]. After a change in the environment occurs, the parent population is discarded, only the offspring is re-evaluated and the algorithm is re-started. With this approach, no new individuals are introduced as is the case with DNSGA-II-A and DNSGA-II-B. Furthermore, forecasting or prediction is incorporated into the algorithm by dividing the objective space into a grid of hyper-cubes where each cube represents a section of the POF for a specific time t. At each time t, the means of the coordinates of the points within the cube is calculated to determine representative points. Each representative point in the grid is assigned a two dimensional time series. Then for each objective, a state space model is selected to model the objective's multi-variate time series. The DNSGA-II with forecasting uses the k forecasted values for k iterations after every pre-defined number of iterations. During the k iterations no function evaluations are performed. After the k iterations the DNSGA-II algorithm proceeds in its normal way. Furthermore, the objective functions are transformed according to defined desirability functions to guide the MOEA towards certain sections of the POF that experts with the required knowledge expect to be of higher relevance than

other parts of the POF. The algorithm was tested on a real-world problem where every 40 generations forecasted values are used for 5 iterations. The results indicated that the multivariate analysis of more than four time series at a time resulted in forecasts with poor confidence intervals. Furthermore, the algorithm struggled when there was missing data, for example when the POF was disconnected or the POF had a low density of solutions in certain areas. According to Roy and Mehnen [129] this problem can be overcome by using larger population sizes and shorter intervals between the forecasting periods.

Recently Koo *et al.* [100] adapted MO-EGS [65] for DMOO and referred to the new algorithm as dMO-EGS. An optional prediction strategy can be incorporated into dMO-EGS, which is then referred to as dynamic MO-EGS with prediction gradient (dMO-EGS-PG). If the prediction strategy is incorporated, an archive centroid is used to predict the movement of the POS.

The gradient prediction strategy relies on a POS that changes in a similar way over time. Therefore, if the POS changes randomly, the gradient predition strategy will be of little or no use. Furthermore, since the prediction strategy relies on the previous values of the POS found by the algorithm, the accuracy of the gradient relies on the accuracy of the algorithm during the previous time steps. dMO-EGS also incorporates storing past information in addition to an archive.

In order to determine the effect of the prediction gradient strategy on the performance of dMO-EGS, dMO-EGS and dMO-EGS-PG were compared using FDA1, FDA3, DIMP1 and DIMP2. The results indicate that dMO-EGS-PG outperformed dMO-EGS on all four DMOOPs with regards to both convergence to the true POS and the spread of the solutions. Furthermore, dMO-EGS-PG converged to the true POS of DIMP2 in situations when dMO-EGS failed to converge, i.e. even when there were severe changes to the environment or the frequency of changes was high [100].

8.2 Summary

This chapter discussed population-based algorithms proposed for DMOO. Five main categories of DMOO algorithms were identified, namely adapted SMOO algorithms, new

types of CI algorithms, conversion of DMOOP into multiple SMOOPs, generic extensions for DMOO and prediction-based approaches.

Many extensions were proposed to SMOO algorithms to solve DMOOPs. Most extensions were proposed to EAs. However, PSO and AIS were also adapted for DMOO. Three new types of CI algorithms were proposed to solve DMOOPs, namely ALife, membrane computing and a competitive-cooperative EA.

Generic extensions or strategies were also proposed that can be applied to any DMOO algorithm. These extensions included an inheritance strategy, re-initialisation strategies, memory-based approaches, immigration schemes, the concept of sphere-dominance, and parallel processing.

In order to reduce the number of function evaluations without decreasing the quality of found solutions, prediction based approaches were introduced. These approaches use knowledge of previous environments to predict the new location of either the POS or POF.

Even though many EA algorithms were proposed for DMOO, only two PSO-based algorithms were introduced to solve DMOOPs. The next part of the thesis discusses a new multi-swarm PSO algorithm for DMOO, namely DVEPSO. The next chapter introduces the extensions made to VEPSO for DMOO and investigates the effect of various guide update approaches on the extended algorithm, i.e. DVEPSO.

Chapter 10

Sensitivity Analysis of Dynamic Vector Evaluated Particle Swarm Optimisation Algorithm

"It is not the strongest of the species that survives, nor the most intelligent. It is the one that is the most adaptable to change." – Charles Darwin

A self-adapting DMOO algorithm that does not require the optimisation of parameters is the ideal. However, before a self-adapting DVEPSO algorithm can be developed, the influence of the various parameters on the performance of DVEPSO has to be understood. Therefore, this chapter investigates how knowledge sharing swarm topologies, approaches to manage boundary constraint violations, and approaches to respond to changes in the environment effect the performance of DVEPSO.

Section 10.1 describes the experimental setup for this study. The results of the experiments are presented in Section 10.2. Finally, Section 10.3 provides a summary of this chapter.

10.1 Experimental Setup

This section describes the experimental setup of the experiments discussed in this chapter. The experiments investigate the approaches discussed in Section 7.2.2 to manage

boundary constraints to determine the influence of the approaches on the performance of DVEPSO.

The search process of DVEPSO is driven through guides. Since each swarm only optimises one objective, knowledge about other objectives are obtained through guides from other swarms. The knowledge sharing stategies discussed in Section 7.2.2 were investigated to determine the approaches' effect on the performance of DVEPSO.

After an environment change, DVEPSO has to respond in an appropriate manner to enable tracking of the changing POF or POS. Two response categories are investigated, namely managing the particles and managing the archive. The approaches to manage the particles after a change occurred were discussed in Section 9.2. Section 7.2.2 discussed the archive management approaches that were investigated in this study.

All simulations were run on an Intel Core i7 X990 8-core machine with a 3.47 GHz processor.

Fifteen benchmark functions were used as discussed in Section 9.4.1. Three performance measures were used to quantify the performance of algorithms, as discussed in Section 9.4.1.

The same default configuration of DVEPSO was used for these experiments as discussed in Section 9.4.1. However, p_s-g_r was used for the guide updates.

Statistical analysis of the obtained data was performed as discussed in Section 9.4.1. The three null hypotheses for these experiments were:

1. There are no statistical significant difference between the performance of the various knowledge sharing approaches.
2. There are no statistical significant difference between the performance of the various responses applied to the particles after a change in the environment occurs.
3. There are no statistical significant difference between the performance of the various responses applied to the archive after an environmental change.

The alternative hypothesis for all three cases above is that there is a difference in mean performance.

The next section discusses the results of the experiments.

10.2 Results

This section discusses the results obtained from the experiments. Section 10.2.1 discusses the results of the various strategies that were used to manage particles that moved outside of the search space and therefore violated the boundary constraints of the DMOOP. The approaches used to share knowledge between the sub-swarms of DVEPSO are discussed in Section 10.2.2. Section 10.2.3 discusses responses to a change in the environment applied to either the particles or the archive. Only the tables highlighting interesting trends and that are therefore discussed, are presented in this section. The other wins and losses tables are presented in Appendix D. Only statistical significant values are included in the tables. The p-values obtained for the various Mann-Whitney U tests, as well as the average performance measure values, are presented in Appendix D.

10.2.1 Management of Boundary Constraint Violations

This section discusses the results obtained by various approaches used to manage particles that moved outside the search space. The results are discussed with regards to each performance measure and each n_t-τ_t combination. Results obtained for each of the three DMOOP Types (Type I, II and III) are also presented. Each approach's overall performance is also discussed. Furthermore, general observations with regards to DVEPSO's performance are also highlighted.

The wins and losses of the various boundary constraint management approaches are presented in Tables 10.1 to 10.12. In Tables 10.1 to 10.12, cl, pe, ra and re refer to the clamping, per element re-initialisation, random, and re-initialisation approaches discussed in Section 7.2.2 respectively. The other approaches proposed to manage boundary constraint violations discussed in Section 7.2.2 are not included in the results, since these approaches found solutions so far away from the true POF, that huge reference vectors (larger than 10^{260}) and therefore huge HV values were obtained.

Results with regards to Performance Measures

This section discusses the results obtained by the various approaches used to manage boundary constraint violations for the various performance measures. The wins and

losses obtained by these approaches over all n_t-τ_t combinations for the various performance measures are presented in Table 10.1.

The following observations are made:

- *cl* performed the best with regards to all performance measures. The worst performance for all performance measures was obtained by *pe*.
- With regards to *acc*, both *pe* and *re* obtained more losses than wins.
- For *stab*, all approaches, except *cl*, obtained more losses than wins. Therefore, the other approaches were completely outperformed by *cl*.
- Similar to *stab*, for *NS* only *cl* was awarded more wins than losses.

Table 10.1: Overall wins and losses for various performance measures obtained by various boundary management strategies

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Boundary management strategies**			
				cl	**pe**	**ra**	**re**
all	all	*acc*	Wins	117	30	78	64
all	all	*acc*	Losses	55	104	61	69
all	all	*acc*	Diff	62	-74	17	-5
all	all	*acc*	Rank	**1**	4	2	3
all	all	*stab*	Wins	125	7	37	17
all	all	*stab*	Losses	5	73	55	53
all	all	*stab*	Diff	120	-66	-18	-36
all	all	*stab*	Rank	**1**	4	2	3
all	all	*NS*	Wins	87	24	40	39
all	all	*NS*	Losses	54	53	46	37
all	all	*NS*	Diff	33	-29	-6	2
all	all	*NS*	Rank	**1**	4	3	2

Results with regards to Various Frequencies and Severities of Change

This section discusses the results obtained for the various n_t-τ_t combinations. The wins and losses obtained by the approaches for management of boundary constraint violations are presented in Table 10.2.

Table 10.2: Overall wins and losses for various frequencies and severities of change obtained by various boundary management strategies

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
10	10	all	Wins	72	13	38	31
10	10	all	Losses	22	55	38	39
10	10	all	Diff	50	-42	0	-8
10	10	all	Rank	**1**	4	2	3
10	25	all	Wins	64	12	29	18
10	25	all	Losses	18	43	33	29
10	25	all	Diff	46	-31	-4	-11
10	25	all	Rank	**1**	4	2	3
10	50	all	Wins	59	12	25	15
10	50	all	Losses	28	32	24	27
10	50	all	Diff	31	-20	1	-12
10	50	all	Rank	**1**	4	2	3
1	10	all	Wins	71	13	35	29
1	10	all	Losses	25	54	35	34
1	10	all	Diff	46	-41	0	-5
1	10	all	Rank	**1**	4	2	3
20	10	all	Wins	63	11	28	27
20	10	all	Losses	21	46	32	30
20	10	all	Diff	42	-35	-4	-3
20	10	all	Rank	**1**	4	3	2

From the obtained results, the following observations are made:

- For all n_t-τ_t combinations *cl* obtained the best performance.
- A bad performance was obtained by *pe* and *re*, with more losses than wins for all n_t-τ_t combinations.
- Even though *ra* obtained more losses than wins for only $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$, *ra* achieved only marginally more wins than losses for the other n_t-τ_t combinations.

Results for Various Dynamic Multi-objective Optimisation Problem Types

This section discusses the obtained results with regards to three DMOOP types, namely Type I to III.

Type I DMOOPs

The wins and losses for Type I DMOOPs are presented in Table 10.3.

Table 10.3: Overall wins and losses for various performance measures obtained by various boundary management strategies solving Type I DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
all	all	*acc*	Wins	15	5	15	7
all	all	*acc*	Losses	15	11	6	10
all	all	*acc*	Diff	0	-6	9	-3
all	all	*acc*	Rank	2	4	**1**	3
all	all	*stab*	Wins	15	1	10	1
all	all	*stab*	Losses	3	9	5	10
all	all	*stab*	Diff	12	-8	5	-9
all	all	*stab*	Rank	**1**	3	2	4
all	all	*NS*	Wins	0	5	7	9
all	all	*NS*	Losses	15	4	2	0
all	all	*NS*	Diff	-15	1	5	9
all	all	*NS*	Rank	4	3	2	**1**

The following are observed:

- The best performance for *acc* was obtained by *ra* and *pe* performed the worst. Both *pe* and *re* obtained more losses than wins. Furthermore, an equal number of wins and losses were achieved by *cl*.
- The best rank for *stab* was achieved by *cl* and the worst by *re*. More losses than wins were obtained by *pe* and *re*.
- For *NS*, the best performance was obtained by *pe*. *cl* performed the worst and obtained more losses than wins.

Table 10.4 presents the wins and losses for Type I DMOOPs with regards to the various n_t-τ_t combinations.

With regards to the various types of environments, the following observations are made:

- *ra* performed the best for all environments, except $n_t = 20$ and $\tau_t = 10$. For $n_t = 20$ and $\tau_t = 10$, *ra* obtained the second best rank. For $n_t = 10$ and $\tau_t = 10$, *cl* performed the worst and both *cl* and *pe* were awarded more losses than wins. For $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, *pe* obtained the worst rank. For $n_t = 10$ and $\tau_t = 50$, *re* performed the worst.
- For gradually changing environments ($n_t = 20$ and $\tau_t = 10$), *cl* performed the best obtaining only wins and no losses. In contrast, *pe* and *re* were awarded only losses and no wins. Both *pe* and *re* obtained the lowest rank.

Table 10.4: Overall wins and losses for various frequencies and severities of change obtained by various boundary management strategies solving Type I DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
10	10	all	Wins	6	5	10	9
10	10	all	Losses	15	7	4	4
10	10	all	Diff	-9	-2	6	5
10	10	all	Rank	4	3	**1**	2
10	25	all	Wins	6	2	7	2
10	25	all	Losses	6	5	2	4
10	25	all	Diff	0	-3	5	-2
10	25	all	Rank	2	4	**1**	3
10	50	all	Wins	6	2	5	2
10	50	all	Losses	6	3	2	4
10	50	all	Diff	0	-1	3	-2
10	50	all	Rank	2	3	**1**	4
1	10	all	Wins	6	2	6	4
1	10	all	Losses	6	5	3	4
1	10	all	Diff	0	-3	3	0
1	10	all	Rank	2	4	**1**	2
20	10	all	Wins	6	0	4	0
20	10	all	Losses	0	4	2	4
20	10	all	Diff	6	-4	2	-4
20	10	all	Rank	**1**	3	2	3

The wins and losses for all Type I DMOOPs over all performance measures and all n_t-τ_t combinations are presented in Table 10.5. The best overall rank for Type I DMOOPs was obtained by *ra*, with *pe* obtaining the worst rank. Only *ra* scored more wins than losses, while the other three approaches were awarded more losses than wins.

Table 10.5: Overall wins and losses obtained by various boundary management strategies solving Type I DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
all	all	all	Wins	30	11	32	17
all	all	all	Losses	33	24	13	20
all	all	all	Diff	-3	-13	19	-3
all	all	all	Rank	2	4	**1**	2

Type II DMOOPs

Table 10.6 presents the wins and losses with regards to the various performance measures

for the boundary constraint management approaches solving Type II DMOOPs.

Table 10.6: Overall wins and losses for various performance measures obtained by various boundary management strategies solving Type II DMOOPs

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Boundary management strategies**			
				cl	**pe**	**ra**	**re**
all	all	*acc*	Wins	72	12	55	28
all	all	*acc*	Losses	16	73	29	49
all	all	*acc*	Diff	56	-61	26	-21
all	all	*acc*	Rank	**1**	4	2	3
all	all	*stab*	Wins	69	4	27	15
all	all	*stab*	Losses	1	51	34	29
all	all	*stab*	Diff	68	-47	-7	-14
all	all	*stab*	Rank	**1**	4	2	3
all	all	NS	Wins	24	8	20	2
all	all	NS	Losses	15	16	10	13
all	all	NS	Diff	9	-8	10	-11
all	all	NS	Rank	2	3	**1**	4

The following observations are made:

- The best performance for *acc* was achieved by *cl* and the worst by *pe*. More losses than wins were obtained by *pe* and *re*.
- With regards to *stab*, *cl* performed the best and *pe* the worst. Only one loss was obtained by *cl*, where as the other three approaches were all awarded more losses than wins.
- For NS the highest rank was obtained by *ra* and the worst rank by *re*. Both *pe* and *re* performed poorly, obtaining more losses than wins.

The wins and losses with regards to the various n_t-τ_t combinations are presented in Table 10.7.

With regards to the various environment types, the following are observed:

- For all environment types, *cl* performed the best and *pe* the worst.
- More losses than wins were awarded to *pe* and *re* for all n_t-τ_t combinations.

Table 10.7: Overall wins and losses for various frequencies and severities of change obtained by various boundary management strategies solving Type II DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
10	10	all	Wins	40	4	26	13
10	10	all	Losses	4	36	18	25
10	10	all	Diff	36	-32	8	-12
10	10	all	Rank	**1**	4	2	3
10	25	all	Wins	32	5	20	8
10	25	all	Losses	5	27	17	16
10	25	all	Diff	27	-22	3	-8
10	25	all	Rank	**1**	4	2	3
10	50	all	Wins	28	7	17	4
10	50	all	Losses	13	19	11	13
10	50	all	Diff	15	-12	6	-9
10	50	all	Rank	**1**	4	2	3
1	10	all	Wins	35	6	23	11
1	10	all	Losses	7	33	15	20
1	10	all	Diff	28	-27	8	-9
1	10	all	Rank	**1**	4	2	3
20	10	all	Wins	30	2	16	9
20	10	all	Losses	3	25	12	17
20	10	all	Diff	27	-23	4	-8
20	10	all	Rank	**1**	4	2	3

Table 10.8 presents the wins and losses for all Type II DMOOPs over all performance measures and all n_t-τ_t combinations. For Type II DMOOPs, *cl* obtained the best overall rank and *pe* the worst rank. Similar to the results for Type I DMOOPs, *pe* and *re* performed poorly, being awarded more losses than wins. All other approaches were completely outperformed by *cl*, with *cl* obtaining 133 more wins than losses and *ra* that ranked second obtained only 29 more wins than losses.

Table 10.8: Overall wins and losses obtained by various boundary management strategies solving Type II DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
all	all	all	Wins	165	24	102	45
all	all	all	Losses	32	140	73	91
all	all	all	Diff	133	-116	29	-46
all	all	all	Rank	**1**	4	2	3

Type III DMOOPs

Table 10.9 presents the wins and losses with regards to the various performance measures for the boundary management approaches solving Type III DMOOPs.

The following observations are made:

- With regards to *acc*, *re* performed the best and *ra* performed the worst. Furthermore, both *pe* and *ra* obtained more losses than wins.
- For *stab*, the highest rank was obtained by *cl* and the lowest rank by *ra*. All approaches, except *cl*, were awarded more losses than wins.
- The best performance for *NS* was achieved by *cl* and *pe* performed the worst. Both *pe* and *ra* scored more losses than wins.

Table 10.9: Overall wins and losses for various performance measures obtained by various boundary management strategies solving Type III DMOOP

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Boundary management strategies**			
				cl	**pe**	**ra**	**re**
all	all	*acc*	Wins	30	13	8	29
all	all	*acc*	Losses	24	20	26	10
all	all	*acc*	Diff	6	-7	-18	19
all	all	*acc*	Rank	2	3	4	**1**
all	all	*stab*	Wins	41	2	0	1
all	all	*stab*	Losses	1	13	16	14
all	all	*stab*	Diff	40	-11	-16	-13
all	all	*stab*	Rank	**1**	2	4	3
all	all	*NS*	Wins	63	11	13	28
all	all	*NS*	Losses	24	33	34	24
all	all	*NS*	Diff	39	-22	-21	4
all	all	*NS*	Rank	**1**	4	3	2

The wins and losses with regards to the various environment types for Type III DMOOPs are presented in Table 10.10.

Observations made with regards to the various n_t-τ_t combinations are:

- *cl* performed the best in all environments, and *ra* the worst. In addition, for $n_t = 1$ and $\tau_t = 10$, *pe* ranked the lowest together with *ra*.
- *cl* was the only approach that obtained more wins than losses in all environments. *pe* and *ra* performed poorly, obtaining more losses than wins for all environments.
- *re* obtained the second best rank in all environments, and obtained more losses

than wins for only the slowly changing environments ($n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$).

Table 10.10: Overall wins and losses for various frequencies and severities of change obtained by various boundary management strategies solving Type III DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
10	10	all	Wins	26	7	5	14
10	10	all	Losses	12	13	17	10
10	10	all	Diff	14	-6	-12	4
10	10	all	Rank	**1**	3	4	2
10	25	all	Wins	26	5	2	8
10	25	all	Losses	7	11	14	9
10	25	all	Diff	19	-6	-12	-1
10	25	all	Rank	**1**	3	4	2
10	50	all	Wins	25	3	3	9
10	50	all	Losses	9	10	11	10
10	50	all	Diff	16	-7	-8	-1
10	50	all	Rank	**1**	3	4	2
1	10	all	Wins	30	5	6	14
1	10	all	Losses	12	16	17	10
1	10	all	Diff	18	-11	-11	4
1	10	all	Rank	**1**	3	3	2
20	10	all	Wins	27	6	5	13
20	10	all	Losses	9	16	17	9
20	10	all	Diff	18	-10	-12	4
20	10	all	Rank	**1**	3	4	2

Table 10.11 presents the wins and losses for Type III DMOOPs measured over all performance measures and all n_t-τ_t combinations. The best overall performance for Type III DMOOPs was obtained by *cl*, with *ra* performing the worst.

Table 10.11: Overall wins and losses obtained by various boundary management strategies solving Type III DMOOPs

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
all	all	all	Wins	134	26	21	58
all	all	all	Losses	49	66	76	48
all	all	all	Diff	85	-40	-55	10
all	all	all	Rank	**1**	3	4	2

Overall Performance

This section discusses the overall performance of the various approaches used to manage boundary constraint violations. The overall wins and losses over all DMOOPs, n_t-τ_t combinations and performance measures are presented in Table 11.19.

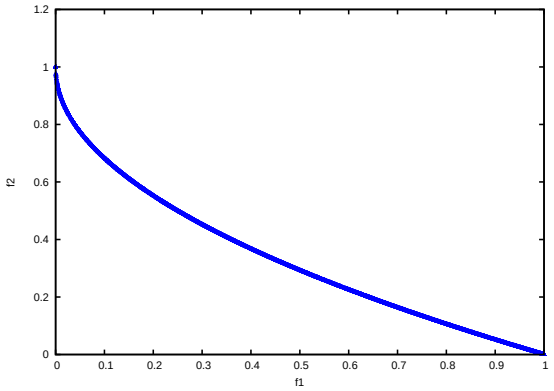
Table 10.12: Overall wins and losses by various boundary management strategies

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
all	all	all	Wins	329	61	155	120
all	all	all	Losses	114	230	162	159
all	all	all	Diff	215	-169	-7	-39
all	all	all	Rank	**1**	4	2	3

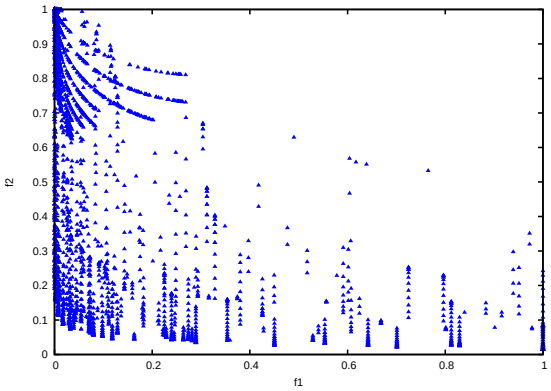
The following are observed:

- The best performance was achieved by *cl*, completely outperforming the other approaches with 215 more wins than losses.
- All approaches, except *cl*, were awarded more losses than wins.
- The lowest rank was obtained by *pe*, with 169 more losses than wins.

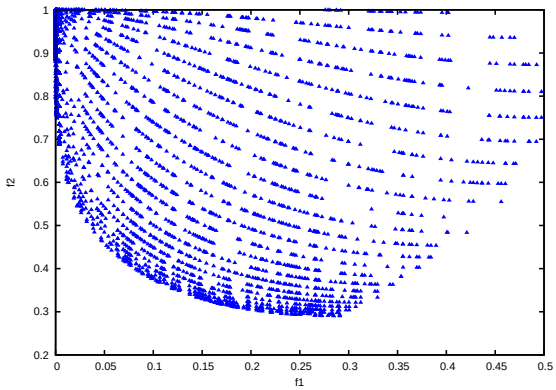
The POF^*s found by *cl* for $n_t = 10$ and $\tau_t = 10$ are illustrated in Figures 10.1 to 10.3. The Figures indicate that DVEPSO successfully tracked the changing POF for $FDA1_{Zhou}$, $FDA2_{Camara}$, FDA3, $FDA3_{Camara}$, DIMP2, dMOP2, $dMOP2_{iso}$ and $dMOP2_{dec}$. Even though DVEPSO struggled with FDA2, it did manage to find some of the POFs, but with a bad spread of solutions. For dMOP3, DVEPSO found solutions close to the true POF, but also quite a few solutions that are a bit further away. With the discontinous function HE1, DVEPSO struggled to find the different continuous sections of the discontinous POF. However, for the discontinous function HE2, DVEPSO managed to find solutions for all the continuous parts of the POF, but the found solutions were not very close to the true POF. For HE6 and HE7 DVEPSO did manage to find some of the POFs, but did not always track the POF successfully with a good spread of solutions. However, for HE9 DVEPSO failed to track the changing POF.



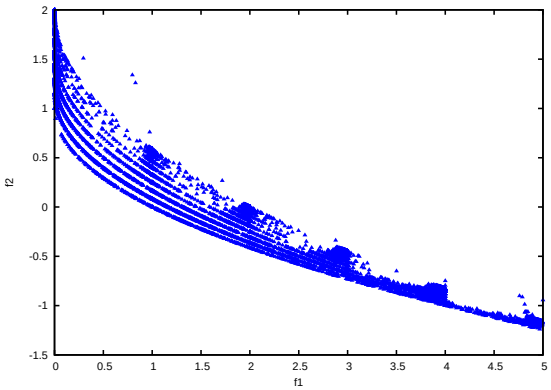
(a) POF^* for FDA1$_{Zhou}$



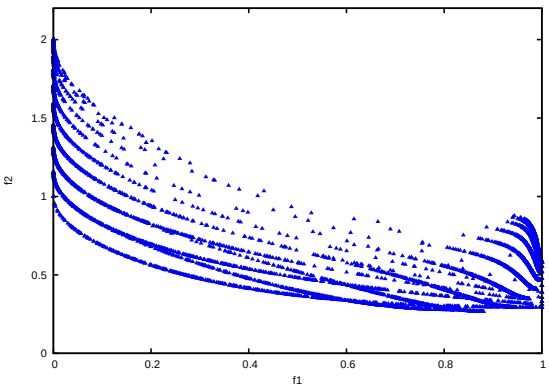
(b) POF^* for FDA2



(c) POF^* for FDA2$_{Camara}$

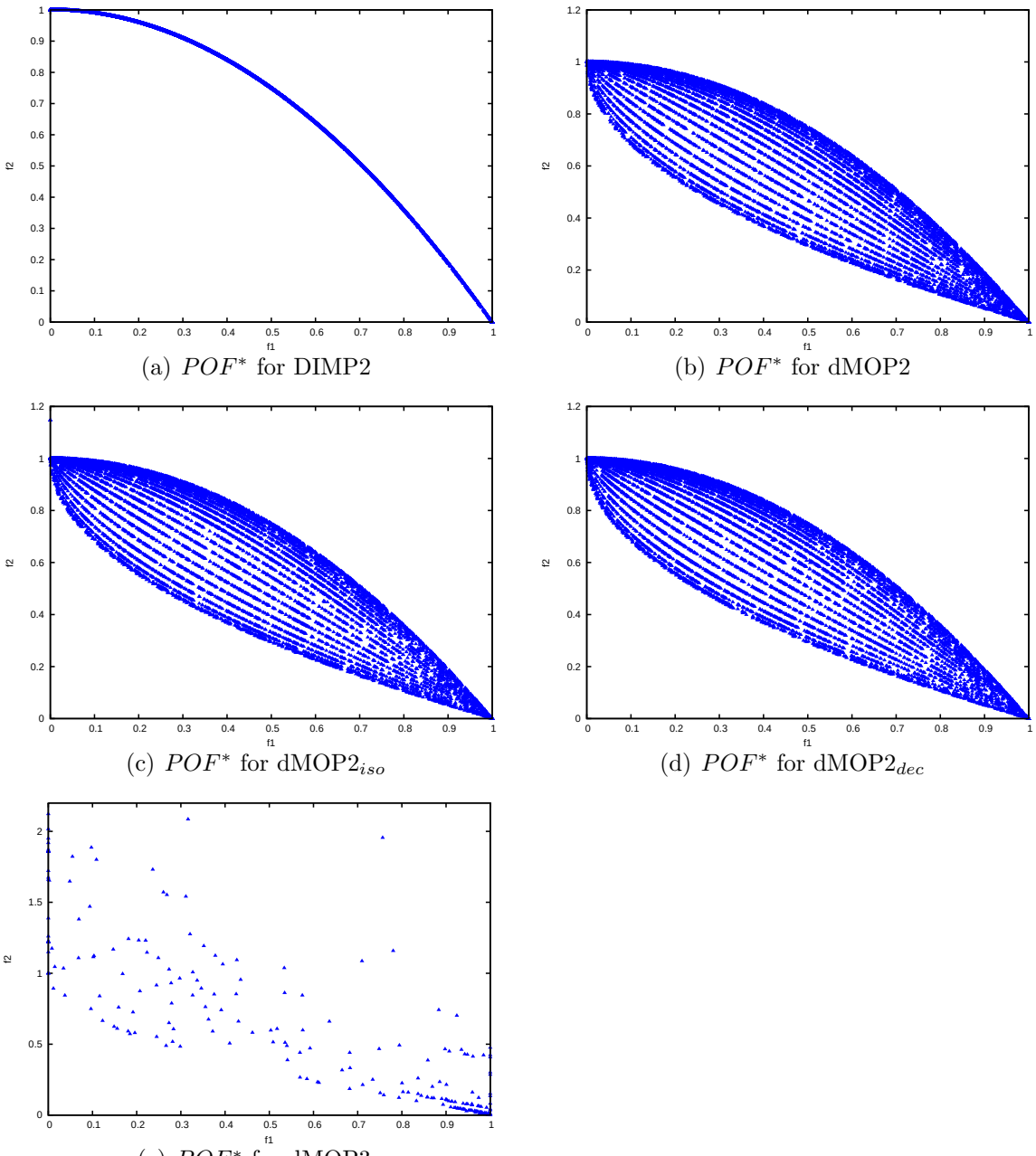


(d) POF^* for FDA3



(e) POF^* for FDA3$_{Camara}$

Figure 10.1: POF^* for the FDA functions of DVEPSO using cl for $n_t = 10$ and $\tau_t = 10$



(a) POF^* for DIMP2

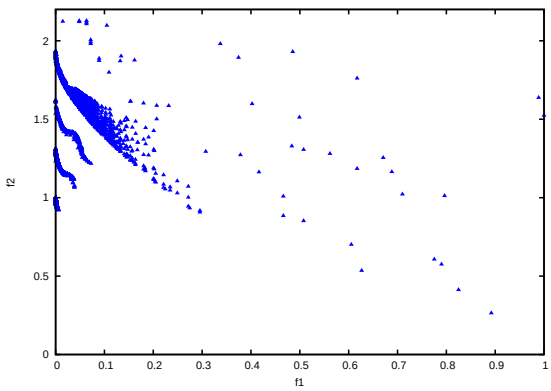
(b) POF^* for dMOP2

(c) POF^* for dMOP2$_{iso}$

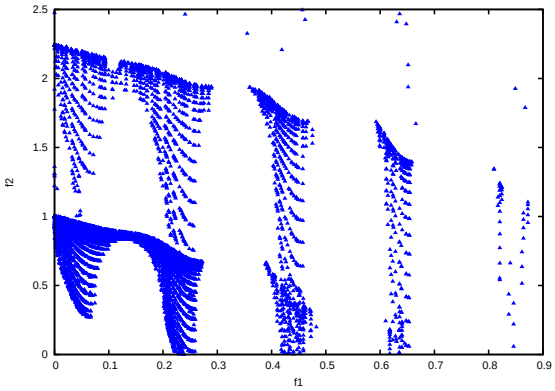
(d) POF^* for dMOP2$_{dec}$

(e) POF^* for dMOP3

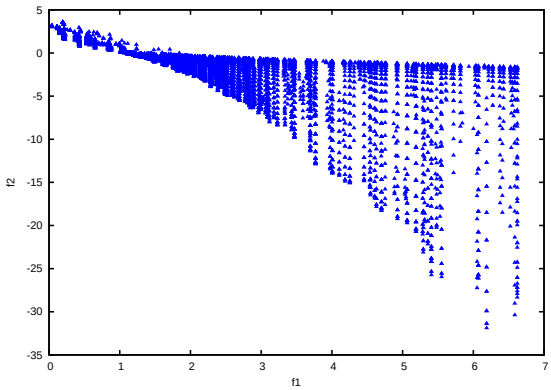
Figure 10.2: POF^* for DIMP2 and the dMOP functions of DVEPSO using cl for $n_t = 10$ and $\tau_t = 10$



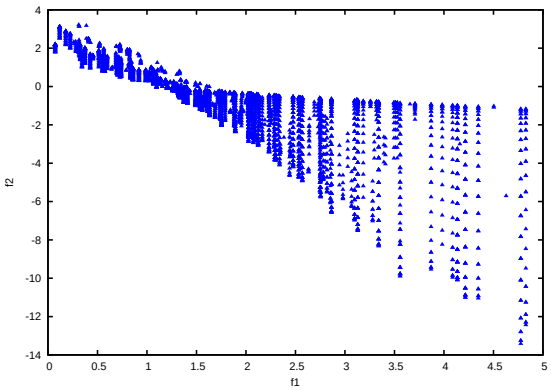
(a) POF^* for HE1



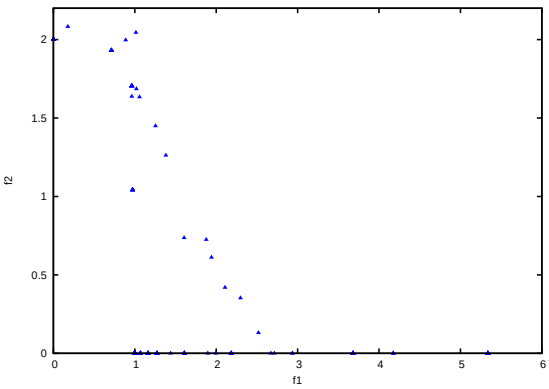
(b) POF^* for HE2



(c) POF^* for HE6



(d) POF^* for HE7



(e) POF^* for HE9

Figure 10.3: POF^* for the HE functions of DVEPSO using cl for $n_t = 10$ and $\tau_t = 10$

General Observations

This section discusses general observations that were made about the performance of the boundary management approaches.

Even though *cl* completely outperformed the other boundary management approaches with regards to the overall wins and losses, there were a few DMOOPs where *cl* did not perform that well, namely FDA2, dMOP3, HE6 and HE9.

The wins and losses for FDA2 are presented in Table 10.13. *cl* and *re* obtained more losses than wins and *re* performed the worst. For the various performance measures, *cl* obtained the second lowest rank for *acc* and *stab* and the second best rank for NS. However, the difference between the overall wins and losses for FDA2 caused *cl* to obtain the lowest overall rank. For the various n_t-τ_t combinations, *cl* performed poorly for $n_t = 10$ and $\tau_t = 25$, obtaining the worst rank. In addition, *cl* obtained the second lowest rank for all other environments. A similar trend was observed for dMOP3. However, when solving dMOP3, *cl* obtained the worst rank for *acc* and *stab*, but the best rank for NS. Furthermore, *cl* performed the best in the slowly changing environments and the second best for $n_t = 1$ and $\tau_t = 10$. However, for $n_t = 10$ and $\tau_t = 10$, *cl* obtained the second lowest rank and for $n_t = 20$ and $\tau_t = 10$, *cl* performed the worst.

Table 10.13: Wins and Losses of FDA2 for various boundary management strategies

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
10	10	*acc*	Wins	0	2	3	0
10	10	*acc*	Losses	2	1	0	2
10	10	*acc*	Diff	-2	1	3	-2
10	10	*acc*	Rank	3	2	**1**	3
10	25	*acc*	Wins	0	2	3	0
10	25	*acc*	Losses	2	1	0	2
10	25	*acc*	Diff	-2	1	3	-2
10	25	*acc*	Rank	3	2	**1**	3
10	50	*acc*	Wins	0	2	3	0
10	50	*acc*	Losses	2	1	0	2
10	50	*acc*	Diff	-2	1	3	-2
10	50	*acc*	Rank	3	2	**1**	3
1	10	*acc*	Wins	0	2	3	0
1	10	*acc*	Losses	2	1	0	2
1	10	*acc*	Diff	-2	1	3	-2
1	10	*acc*	Rank	3	2	**1**	3

Continued on next page

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
20	10	*acc*	Wins	0	2	3	0
20	10	*acc*	Losses	2	1	0	2
20	10	*acc*	Diff	-2	1	3	-2
20	10	*acc*	Rank	3	2	**1**	3
all	all	*acc*	Wins	0	10	15	0
all	all	*acc*	Losses	10	5	0	10
all	all	*acc*	Diff	-10	5	15	-10
all	all	*acc*	Rank	3	2	**1**	3
10	10	*stab*	Wins	0	1	0	1
10	10	*stab*	Losses	0	0	2	0
10	10	*stab*	Diff	0	1	-2	1
10	10	*stab*	Rank	3	**1**	4	**1**
10	25	*stab*	Wins	0	1	1	1
10	25	*stab*	Losses	1	0	2	0
10	25	*stab*	Diff	-1	1	-1	1
10	25	*stab*	Rank	3	**1**	3	**1**
10	50	*stab*	Wins	0	1	0	0
10	50	*stab*	Losses	0	0	1	0
10	50	*stab*	Diff	0	1	-1	0
10	50	*stab*	Rank	2	**1**	4	2
1	10	*stab*	Wins	0	1	0	0
1	10	*stab*	Losses	0	0	1	0
1	10	*stab*	Diff	0	1	-1	0
1	10	*stab*	Rank	2	**1**	4	2
all	all	*stab*	Wins	0	4	1	2
all	all	*stab*	Losses	1	0	6	0
all	all	*stab*	Diff	-1	4	-5	2
all	all	*stab*	Rank	3	**1**	4	2
10	10	*NS*	Wins	1	3	0	2
10	10	*NS*	Losses	2	0	3	1
10	10	*NS*	Diff	-1	3	-3	1
10	10	*NS*	Rank	3	**1**	4	2
10	25	*NS*	Wins	1	1	0	1
10	25	*NS*	Losses	0	0	3	0
10	25	*NS*	Diff	1	1	-3	1
10	25	*NS*	Rank	**1**	**1**	4	**1**
10	50	*NS*	Wins	1	1	0	1
10	50	*NS*	Losses	0	0	3	0
10	50	*NS*	Diff	1	1	-3	1
10	50	*NS*	Rank	**1**	**1**	4	**1**
1	10	*NS*	Wins	1	1	0	1
1	10	*NS*	Losses	0	0	3	0
1	10	*NS*	Diff	1	1	-3	1
1	10	*NS*	Rank	**1**	**1**	4	**1**
20	10	*NS*	Wins	1	1	0	2

Continued on next page

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	pe	ra	re
20	10	NS	Losses	1	0	3	0
20	10	NS	Diff	0	1	-3	2
20	10	NS	Rank	3	2	4	**1**
all	all	NS	Wins	3	0	15	1
all	all	NS	Losses	5	7	0	7
all	all	NS	Diff	-2	-7	15	-6
all	all	NS	Rank	2	4	**1**	3
10	10	all	Wins	2	3	6	2
10	10	all	Losses	3	4	2	4
10	10	all	Diff	-1	-1	4	-2
10	10	all	Rank	2	2	**1**	4
10	25	all	Wins	0	3	7	1
10	25	all	Losses	4	2	2	3
10	25	all	Diff	-4	1	5	-2
10	25	all	Rank	4	2	**1**	3
10	50	all	Wins	0	3	6	0
10	50	all	Losses	3	2	1	3
10	50	all	Diff	-3	1	5	-3
10	50	all	Rank	3	2	**1**	3
1	10	all	Wins	0	3	6	0
1	10	all	Losses	3	2	1	3
1	10	all	Diff	-3	1	5	-3
1	10	all	Rank	3	2	**1**	3
20	10	all	Wins	1	2	6	0
20	10	all	Losses	3	2	0	4
20	10	all	Diff	-2	0	6	-4
20	10	all	Rank	3	2	**1**	4
all	all	all	Wins	3	14	31	3
all	all	all	Losses	16	12	6	17
all	all	all	Diff	-13	2	25	-14
all	all	all	Rank	3	2	**1**	4

The *cl* approach obtained a mixed performance with two of the HE functions where the decision variables each have their own POS and the POSs are defined by non-linear functions. Table 10.14 presents the wins and losses for HE6. For *acc*, there was no statistical significant difference between the performance measure values of the various approaches for all n_t-τ_t combinations, except $n_t = 1$ and $\tau_t = 10$. Furthermore, for *stab* there was no statistical significant difference for all n_t-τ_t combinations. However, for NS there was a statistical significant difference, and *cl* performed the best with regards to NS. Therefore, *cl* obtained the best overall rank for HE6. The second best rank was obtained by *re*. However, *cl* was the only approach that obtained more wins than losses.

Table 10.14: Wins and Losses of HE6 for various boundary management strategies

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
1	10	*acc*	Wins	0	1	1	1
1	10	*acc*	Losses	3	0	0	0
1	10	*acc*	Diff	-3	1	1	1
1	10	*acc*	Rank	4	**1**	**1**	**1**
all	all	*acc*	Wins	0	1	1	1
all	all	*acc*	Losses	3	0	0	0
all	all	*acc*	Diff	-3	1	1	1
all	all	*acc*	Rank	4	**1**	**1**	**1**
10	10	*NS*	Wins	3	0	0	1
10	10	*NS*	Losses	0	1	2	1
10	10	*NS*	Diff	3	-1	-2	0
10	10	*NS*	Rank	**1**	3	4	2
10	25	*NS*	Wins	3	0	0	0
10	25	*NS*	Losses	0	1	1	1
10	25	*NS*	Diff	3	-1	-1	-1
10	25	*NS*	Rank	**1**	2	2	2
10	50	*NS*	Wins	3	0	0	0
10	50	*NS*	Losses	0	1	1	1
10	50	*NS*	Diff	3	-1	-1	-1
10	50	*NS*	Rank	**1**	2	2	2
1	10	*NS*	Wins	3	0	0	1
1	10	*NS*	Losses	0	1	2	1
1	10	*NS*	Diff	3	-1	-2	0
1	10	*NS*	Rank	**1**	3	4	2
20	10	*NS*	Wins	3	0	0	0
20	10	*NS*	Losses	0	1	1	1
20	10	*NS*	Diff	3	-1	-1	-1
20	10	*NS*	Rank	**1**	2	2	2
all	all	*NS*	Wins	15	0	0	2
all	all	*NS*	Losses	0	5	7	5
all	all	*NS*	Diff	15	-5	-7	-3
all	all	*NS*	Rank	**1**	3	4	2
10	10	all	Wins	3	0	0	1
10	10	all	Losses	0	1	2	1
10	10	all	Diff	3	-1	-2	0
10	10	all	Rank	**1**	3	4	2
10	25	all	Wins	3	0	0	0
10	25	all	Losses	0	1	1	1
10	25	all	Diff	3	-1	-1	-1
10	25	all	Rank	**1**	2	2	2
10	50	all	Wins	3	0	0	0
10	50	all	Losses	0	1	1	1
10	50	all	Diff	3	-1	-1	-1

Continued on next page

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
10	50	all	Rank	**1**	2	2	2
1	10	all	Wins	3	1	1	2
1	10	all	Losses	3	1	2	1
1	10	all	Diff	0	0	-1	1
1	10	all	Rank	2	2	4	**1**
20	10	all	Wins	3	0	0	0
20	10	all	Losses	0	1	1	1
20	10	all	Diff	3	-1	-1	-1
20	10	all	Rank	**1**	2	2	2
all	all	all	Wins	15	1	1	3
all	all	all	Losses	3	5	7	5
all	all	all	Diff	12	-4	-6	-2
all	all	all	Rank	**1**	3	4	2

The wins and losses for HE9 are presented in Table 10.15. The poorest performance with regards to *acc* was obtained by *cl*. However, for *stab* and NS, *cl* performed the best. However, for the various environment types, *cl* performed the best for all n_t-τ_t combinations.

Table 10.15: Wins and Losses of HE9 for various boundary management strategies

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
10	10	*acc*	Wins	0	1	1	1
10	10	*acc*	Losses	3	0	0	0
10	10	*acc*	Diff	-3	1	1	1
10	10	*acc*	Rank	4	**1**	**1**	**1**
10	25	*acc*	Wins	0	1	1	1
10	25	*acc*	Losses	3	0	0	0
10	25	*acc*	Diff	-3	1	1	1
10	25	*acc*	Rank	4	**1**	**1**	**1**
10	50	*acc*	Wins	0	1	1	1
10	50	*acc*	Losses	3	0	0	0
10	50	*acc*	Diff	-3	1	1	1
10	50	*acc*	Rank	4	**1**	**1**	**1**
1	10	*acc*	Wins	0	1	1	1
1	10	*acc*	Losses	3	0	0	0
1	10	*acc*	Diff	-3	1	1	1
1	10	*acc*	Rank	4	**1**	**1**	**1**
20	10	*acc*	Wins	0	1	1	1
20	10	*acc*	Losses	3	0	0	0
20	10	*acc*	Diff	-3	1	1	1
20	10	*acc*	Rank	4	**1**	**1**	**1**
all	all	*acc*	Wins	0	5	5	5

Continued on next page

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
all	all	*acc*	Losses	15	0	0	0
all	all	*acc*	Diff	-15	5	5	5
all	all	*acc*	Rank	4	**1**	**1**	**1**
10	10	*stab*	Wins	3	0	0	0
10	10	*stab*	Losses	0	1	1	1
10	10	*stab*	Diff	3	-1	-1	-1
10	10	*stab*	Rank	**1**	2	2	2
10	25	*stab*	Wins	3	0	0	0
10	25	*stab*	Losses	0	1	1	1
10	25	*stab*	Diff	3	-1	-1	-1
10	25	*stab*	Rank	**1**	2	2	2
10	50	*stab*	Wins	3	0	0	0
10	50	*stab*	Losses	0	1	1	1
10	50	*stab*	Diff	3	-1	-1	-1
10	50	*stab*	Rank	**1**	2	2	2
1	10	*stab*	Wins	3	0	0	0
1	10	*stab*	Losses	0	1	1	1
1	10	*stab*	Diff	3	-1	-1	-1
1	10	*stab*	Rank	**1**	2	2	2
20	10	*stab*	Wins	3	0	0	0
20	10	*stab*	Losses	0	1	1	1
20	10	*stab*	Diff	3	-1	-1	-1
20	10	*stab*	Rank	**1**	2	2	2
all	all	*stab*	Wins	15	0	0	0
all	all	*stab*	Losses	0	5	5	5
all	all	*stab*	Diff	15	-5	-5	-5
all	all	*stab*	Rank	**1**	2	2	2
10	10	*NS*	Wins	3	1	1	0
10	10	*NS*	Losses	0	1	1	3
10	10	*NS*	Diff	3	0	0	-3
10	10	*NS*	Rank	**1**	2	2	4
10	25	*NS*	Wins	3	0	0	0
10	25	*NS*	Losses	0	1	1	1
10	25	*NS*	Diff	3	-1	-1	-1
10	25	*NS*	Rank	**1**	2	2	2
10	50	*NS*	Wins	3	0	1	0
10	50	*NS*	Losses	0	1	1	2
10	50	*NS*	Diff	3	-1	0	-2
10	50	*NS*	Rank	**1**	3	2	4
1	10	*NS*	Wins	3	0	0	0
1	10	*NS*	Losses	0	1	1	1
1	10	*NS*	Diff	3	-1	-1	-1
1	10	*NS*	Rank	**1**	2	2	2
20	10	*NS*	Wins	3	0	1	1
20	10	*NS*	Losses	0	3	1	1

Continued on next page

n_t	τ_t	PM	Results	Boundary management strategies			
				cl	**pe**	**ra**	**re**
20	10	NS	Diff	3	-3	0	0
20	10	NS	Rank	**1**	4	2	2
all	all	NS	Wins	15	1	3	1
all	all	NS	Losses	0	7	5	8
all	all	NS	Diff	15	-6	-2	-7
all	all	NS	Rank	**1**	3	2	4
10	10	all	Wins	6	2	2	1
10	10	all	Losses	3	2	2	4
10	10	all	Diff	3	0	0	-3
10	10	all	Rank	**1**	2	2	4
10	25	all	Wins	6	1	1	1
10	25	all	Losses	3	2	2	2
10	25	all	Diff	3	-1	-1	-1
10	25	all	Rank	**1**	2	2	2
10	50	all	Wins	6	1	2	1
10	50	all	Losses	3	2	2	3
10	50	all	Diff	3	-1	0	-2
10	50	all	Rank	**1**	3	2	4
1	10	all	Wins	6	1	1	1
1	10	all	Losses	3	2	2	2
1	10	all	Diff	3	-1	-1	-1
1	10	all	Rank	**1**	2	2	2
20	10	all	Wins	6	1	2	2
20	10	all	Losses	3	4	2	2
20	10	all	Diff	3	-3	0	0
20	10	all	Rank	**1**	4	2	2
all	all	all	Wins	30	6	8	6
all	all	all	Losses	15	12	10	13
all	all	all	Diff	15	-6	-2	-7
all	all	all	Rank	**1**	3	2	4

The results discussed above indicate that possible future work should include the development of a hyper-heuristic approach that selects the best boundary mechanism on the fly.

The next section discusses results obtained by various knowledge sharing approaches.

10.2.2 Knowledge Sharing Swarm Topologies

This section discusses results obtained by various approaches used to share knowledge between the sub-swarms of DVEPSO. The results are discussed with regards to each performance measure and each n_t-τ_t combination. Results obtained for DMOOPs of

Type I, II and III are also presented. The performance of each knowledge sharing approach measured over all performance measures, n_t-τ_t combinations and DMOOPs are discussed to determine the approach's overall performance. Furthermore, general observations are also highlighted. The wins and losses of the various knowledge sharing strategies are presented in Tables 10.16 to 10.27. In Tables 10.16 to 10.27, *ra* and *ri* indicates either a random or ring topology, and *g* or *t* indicates whether the guide is selected as the gbest of the selected sub-swarm or through tournament selection applied to the selected sub-swarm's particles' positions.

Results with regards to Performance Measures

This section discusses the results with regards to the performance measures obtained by the various approaches used to share knowledge between the sub-swarms. The wins and losses obtained by the approaches over all n_t-τ_t combinations for the various performance measures are presented in Table 10.16.

Table 10.16: Overall wins and losses for various performance measures obtained by various knowledge sharing strategies

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	*acc*	Wins	64	93	23	87
all	all	*acc*	Losses	70	31	132	34
all	all	*acc*	Diff	-6	62	-109	53
all	all	*acc*	Rank	3	**1**	4	2
all	all	*stab*	Wins	60	70	7	76
all	all	*stab*	Losses	43	19	127	24
all	all	*stab*	Diff	17	51	-120	52
all	all	*stab*	Rank	3	2	4	**1**
all	all	*NS*	Wins	62	56	42	62
all	all	*NS*	Losses	38	45	102	37
all	all	*NS*	Diff	24	11	-60	25
all	all	*NS*	Rank	2	3	4	**1**

From the obtained results, the following observations are made:

- The best performance for *acc* was achieved by *ra-t*, while *ri-g* performed the worst. Both approaches that use tournament selection outperformed the approaches using the gbest values.

- For *stab*, *ri-t* performed the best and *ri-g* performed significantly the worst. Once again, both tournament selection approaches outperformed the gbest approaches. A really poor performance was achieved by *ri-g*, obtaining 120 more losses than wins.
- Measured against NS, *ri-t* performed the best and *ri-g* performed significantly the worst. All approaches, except *ri-g*, were awarded more wins than losses.

Results with regards to Various Frequencies and Severities of Change

This section discusses the results obtained with regards to the various n_t-τ_t combinations. The wins and losses obtained by the knowledge sharing approaches are presented in Table 10.17.

Table 10.17: Overall wins and losses for various frequencies and severities of change obtained by various knowledge sharing strategies

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**knowledge sharing strategies**			
				ra-g	**ra-t**	**ri-g**	**ri-t**
10	10	all	Wins	3	4	0	7
10	10	all	Losses	2	2	9	1
10	10	all	Diff	1	2	-9	6
10	10	all	Rank	3	2	4	**1**
10	25	all	Wins	3	4	0	7
10	25	all	Losses	2	2	9	1
10	25	all	Diff	1	2	-9	6
10	25	all	Rank	3	2	4	**1**
10	50	all	Wins	3	4	0	8
10	50	all	Losses	3	2	9	1
10	50	all	Diff	0	2	-9	7
10	50	all	Rank	3	2	4	**1**
1	10	all	Wins	4	3	0	8
1	10	all	Losses	2	4	9	0
1	10	all	Diff	2	-1	-9	8
1	10	all	Rank	2	3	4	**1**
20	10	all	Wins	3	3	0	7
20	10	all	Losses	2	2	9	0
20	10	all	Diff	1	1	-9	7
20	10	all	Rank	2	2	4	**1**

The following observations are made with regards to the various environment types:

- For all environments the best performance was obtained by *ri-t* and the worst by

ri-g. Furthermore, the tournament selection approaches outperformed the gbest approaches.

Results for Various Dynamic Multi-objective Optimisation Problem Types

The results obtained for DMOOPs of Type I, II and III are discussed in this section.

Type I DMOOPs

This section discusses the results obtained for Type I DMOOPs. The wins and losses obtained by the various knowledge sharing approaches for the performance measures measured over all n_t-τ_t combinations are presented in Table 10.16.

Table 10.18: Overall wins and losses for various performance measures obtained by various knowledge sharing strategies solving Type I DMOOPs

$\mathbf{n_t}$	$\tau_\mathbf{t}$	**PM**	**Results**	**knowledge sharing strategies**			
				ra-g	**ra-t**	**ri-g**	**ri-t**
all	all	*acc*	Wins	5	5	0	15
all	all	*acc*	Losses	5	5	15	0
all	all	*acc*	Diff	0	0	-15	15
all	all	*acc*	Rank	2	2	4	**1**
all	all	*stab*	Wins	5	5	0	15
all	all	*stab*	Losses	5	5	15	0
all	all	*stab*	Diff	0	0	-15	15
all	all	*stab*	Rank	2	2	4	**1**
all	all	*NS*	Wins	6	8	0	7
all	all	*NS*	Losses	1	2	15	3
all	all	*NS*	Diff	5	6	-15	4
all	all	*NS*	Rank	2	**1**	4	3

From the obtained results, the following observations are made:

- For *acc* and *stab* the highest rank was obtained by *ri-t*. The worst rank was obtained by *ri-g*, with zero wins.
- Measured against *NS*, *ra-t* ranked the best and *ri-g* the worst. Furthermore, *ri-g* was the only approach that obtained more losses than wins.

Table 10.19 presents the wins and losses with regards to the various n_t-τ_t combinations.

Table 10.19: Overall wins and losses for various frequencies and severities of change obtained by various knowledge sharing strategies solving Type I DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
10	10	all	Wins	3	4	0	7
10	10	all	Losses	2	2	9	1
10	10	all	Diff	1	2	-9	6
10	10	all	Rank	3	2	4	**1**
10	25	all	Wins	3	4	0	7
10	25	all	Losses	2	2	9	1
10	25	all	Diff	1	2	-9	6
10	25	all	Rank	3	2	4	**1**
10	50	all	Wins	3	4	0	8
10	50	all	Losses	3	2	9	1
10	50	all	Diff	0	2	-9	7
10	50	all	Rank	3	2	4	**1**
1	10	all	Wins	4	3	0	8
1	10	all	Losses	2	4	9	0
1	10	all	Diff	2	-1	-9	8
1	10	all	Rank	2	3	4	**1**
20	10	all	Wins	3	3	0	7
20	10	all	Losses	2	2	9	0
20	10	all	Diff	1	1	-9	7
20	10	all	Rank	2	2	4	**1**

With regards to the various n_t-τ_t combinations, the following observations are made:

- For all n_t-τ_t combinations the best performance was achieved by *ri-t* and *ri-g* performed the worst.
- *ri-g* performed poorly, obtaining more losses than wins for all environments.

The wins and losses measured over all performance measures and n_t-τ_t for Type I DMOOPs are presented in Table 10.20.

Table 10.20: Overall wins and losses obtained by various knowledge sharing strategies solving Type I DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	all	Wins	16	18	0	37
all	all	all	Losses	11	12	45	3
all	all	all	Diff	5	6	-45	34
all	all	all	Rank	3	2	4	**1**

The best overall rank for Type I DMOOPs was obtained by *ri-t*, with *ri-g* obtaining the worst rank. Only *ri-g* was awarded more losses than wins for Type I DMOOPs.

Type II DMOOPs

This section discusses results obtained for Type II DMOOPs. Table 10.21 presents the wins and losses with regards to the various performance measures over all n_t-τ_t combinations for Type II DMOOPs.

Table 10.21: Overall wins and losses for various performance measures obtained by various knowledge sharing strategies solving Type II DMOOPs

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**knowledge sharing strategies**			
				ra-g	**ra-t**	**ri-g**	**ri-t**
all	all	*acc*	Wins	28	43	12	62
all	all	*acc*	Losses	44	20	79	2
all	all	*acc*	Diff	-16	23	-67	60
all	all	*acc*	Rank	3	2	4	**1**
all	all	*stab*	Wins	37	56	0	49
all	all	*stab*	Losses	29	5	94	14
all	all	*stab*	Diff	8	51	-94	35
all	all	*stab*	Rank	3	**1**	4	2
all	all	*NS*	Wins	27	31	29	35
all	all	*NS*	Losses	24	22	59	17
all	all	*NS*	Diff	3	9	-30	18
all	all	*NS*	Rank	3	2	4	**1**

The following observations are made:

- The best performance with regards to *acc* was obtained by *ri-t* and the worst performance by *ri-g*. Both gbest approaches performed poorly, obtaining more losses than wins.
- For *stab*, *ra-t* achieved the best rank and the worst rank was obtained by *ri-g*. Once again, the tournament approaches outperformed the gbest approaches.
- Measured against *NS*, *ri-t* performed the best and *ri-g* performed the worst. Similar to *acc* and *stab*, the tournament approaches outperformed the gbest approaches with regards to *NS*.

Table 10.22 presents the wins and losses of the knowledge sharing approaches with regards to the various n_t-τ_t combinations for Type II DMOOPs.

Table 10.22: Overall wins and losses for various frequencies and severities of change obtained by various knowledge sharing strategies solving Type II DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
10	10	all	Wins	22	35	9	33
10	10	all	Losses	25	9	56	9
10	10	all	Diff	-3	26	-47	24
10	10	all	Rank	3	**1**	4	2
10	25	all	Wins	17	24	8	28
10	25	all	Losses	17	7	47	6
10	25	all	Diff	0	17	-39	22
10	25	all	Rank	3	2	4	**1**
10	50	all	Wins	17	26	10	32
10	50	all	Losses	25	12	42	6
10	50	all	Diff	-8	14	-32	26
10	50	all	Rank	3	2	4	**1**
1	10	all	Wins	19	25	5	32
1	10	all	Losses	18	10	48	5
1	10	all	Diff	1	15	-43	27
1	10	all	Rank	3	2	4	**1**
20	10	all	Wins	17	20	9	21
20	10	all	Losses	12	9	39	7
20	10	all	Diff	5	11	-30	14
20	10	all	Rank	3	2	4	**1**

With regards to the various n_t-τ_t combinations, the following observations are made:

- The approach that performed the worst for all n_t-τ_t combinations was *ri-g*.
- *ri-t* performed the best for all environments.
- For $n_t = 10$ and $\tau_t = 10$, the tournament approaches outperformed the gbest approaches. *ri-g* performed poorly, obtaining more losses than wins.
- In a slow changing environment ($\tau_t = 25$ and $\tau_t = 50$)the gbest approaches was completely outperformed by the tournament approaches.

The wins and losses of the knowledge sharing approaches for Type II DMOOPs calculated over all performance measures and n_t-τ_t combinations are presented in Table 10.23. For Type II DMOOPs, *ri-t* obtained the best overall rank and *ri-g* performed the worst. Both tournament approaches performed really well, outperforming the gbest approaches.

Table 10.23: Overall wins and losses btained by various knowledge sharing strategies solving Type II DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	all	Wins	92	130	41	146
all	all	all	Losses	97	47	232	33
all	all	all	Diff	-5	83	-191	113
all	all	all	Rank	3	2	4	**1**

Type III DMOOPs

Table 10.24 presents the wins and losses with regards to the various performance measures for the knowledge sharing approaches solving Type III DMOOPs.

Table 10.24: Overall wins and losses for various performance measures obtained by various knowledge sharing strategies solving Type III DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	*acc*	Wins	31	45	11	10
all	all	*acc*	Losses	21	6	38	32
all	all	*acc*	Diff	10	39	-27	-22
all	all	*acc*	Rank	2	**1**	4	3
all	all	*stab*	Wins	18	9	7	12
all	all	*stab*	Losses	9	9	18	10
all	all	*stab*	Diff	9	0	-11	2
all	all	*stab*	Rank	**1**	3	4	2
all	all	NS	Wins	29	17	13	20
all	all	NS	Losses	13	21	28	17
all	all	NS	Diff	16	-4	-15	3
all	all	NS	Rank	**1**	3	4	2

The following observations are made:

- For *acc*, the best performance was obtained by *ra-t* and *ri-g* performed the worst. Both ring approaches performed poorly, with more losses than wins.
- The best performance with regards to *stab* was achieved by *ra-g* and the worst performance was achieved by *ri-g*.
- With regards to NS, *ra-g* ranked the best and *ri-g* ranked the worst.

The wins and losses for Type III DMOOPs with regards to the various environment types are presented in Table 10.25.

Table 10.25: Overall wins and losses for various frequencies and severities of change obtained by various knowledge sharing strategies solving Type III DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
10	10	all	Wins	19	19	8	9
10	10	all	Losses	9	10	21	15
10	10	all	Diff	10	9	-13	-6
10	10	all	Rank	**1**	2	4	3
10	25	all	Wins	13	14	4	6
10	25	all	Losses	7	4	17	9
10	25	all	Diff	6	10	-13	-3
10	25	all	Rank	2	**1**	4	3
10	50	all	Wins	14	7	4	8
10	50	all	Losses	6	4	15	8
10	50	all	Diff	8	3	-11	0
10	50	all	Rank	**1**	2	4	3
1	10	all	Wins	18	16	11	8
1	10	all	Losses	11	10	14	18
1	10	all	Diff	7	6	-3	-10
1	10	all	Rank	**1**	2	3	4
20	10	all	Wins	14	15	4	11
20	10	all	Losses	10	8	17	9
20	10	all	Diff	4	7	-13	2
20	10	all	Rank	2	**1**	4	3

With regards to the various n_t-τ_t combinations the following observations are made:

- For all n_t-τ_t combinations, except $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$, the best performance was obtained by *ra-g*.
- For $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$, *ra-t* performed the best.
- The worst rank was achieved by *ri-g* for all n_t-τ_t combinations, except $n_t = 1$ and $\tau_t = 10$.
- The worst performing approach for $n_t = 1$ and $\tau_t = 10$ was *ri-t*.

Table 10.26 presents the wins and losses for Type III DMOOPs measured over all performance measures and all n_t-τ_t combinations. When solving Type III DMOOPs, the best overall performance was obtained by both *ra-g* and *ra-t*, with *ri-g* performing the worst. The random approaches outperformed the ring approaches on the Type III DMOOPs. Both ring approaches performed poorly, being awarded more losses than wins.

Table 10.26: Overall wins and losses obtained by various knowledge sharing strategies solving Type III DMOOPs

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	all	Wins	78	71	31	42
all	all	all	Losses	43	36	84	59
all	all	all	Diff	35	35	-53	-17
all	all	all	Rank	**1**	**1**	4	3

Overall Performance

This section discusses the overall performance of the knowledge sharing approaches. The overall wins and losses over all DMOOPs, n_t-τ_t combinations and performance measures are presented in Table 10.27.

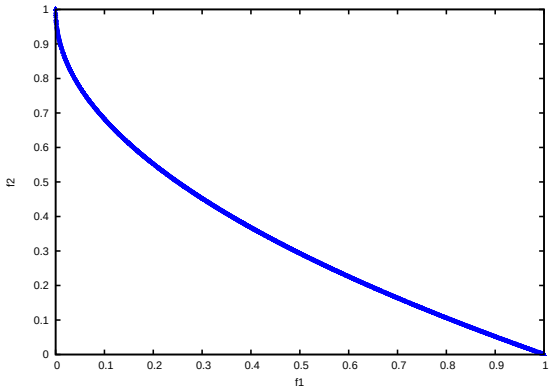
Table 10.27: Overall wins and losses obtained by various knowledge sharing strategies

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
all	all	all	Wins	186	219	72	225
all	all	all	Losses	151	95	361	95
all	all	all	Diff	35	124	-289	130
all	all	all	Rank	3	2	4	**1**

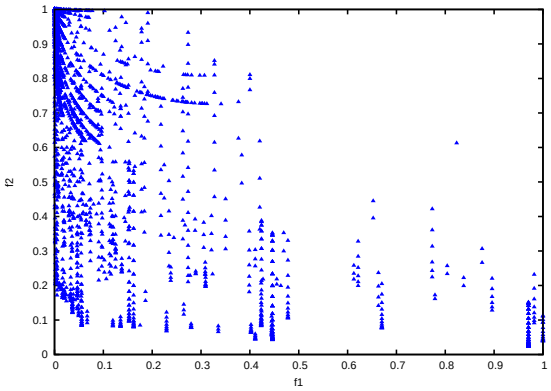
The following are observed:

- The best overall rank was achieved by *ri-t* and the worst performance was obtained by *ri-g*. Both *ri-t* and *ra-t* performed well, obtaining 130 and 124 more wins than losses respectively.
- The tournament approaches completely outperformed the gbest approaches.
- *ra-g* obtained only more wins than losses. In contrast, *ri-g* was awarded 289 more losses than wins.

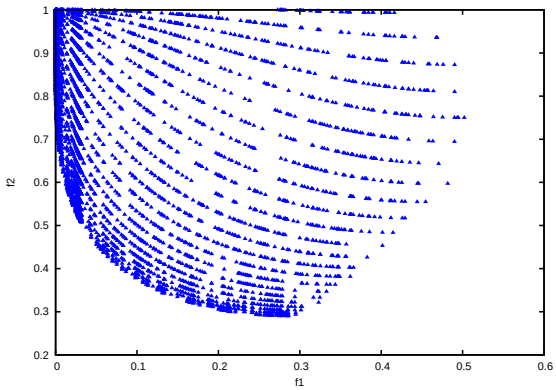
Figures 10.4 to 10.6 illustrate the POF^*s for $n_t = 10$ and $\tau_t = 10$ found by *ra-t*. The POF^*s found by *ra-t* followed the same trend as the POF^*s found by *cl* discussed in Section 10.2.1. However, *ra-t* found a better spread of solutions for FDA3 and FDA3_{Camara}. Furthermore, *ra-t* found solutions closer to the true POF of dMOP3.



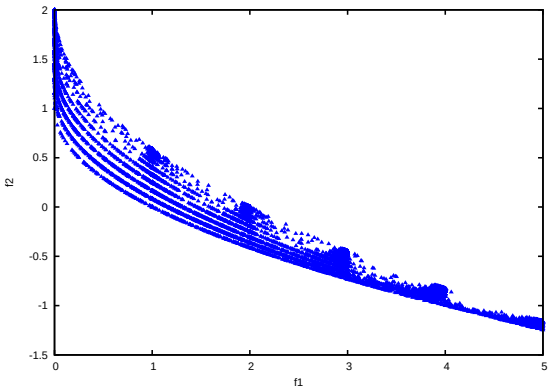
(a) POF^* for FDA1$_{Zhou}$



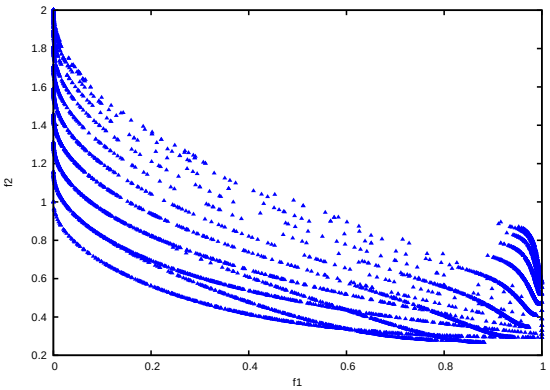
(b) POF^* for FDA2



(c) POF^* for FDA2$_{Camara}$

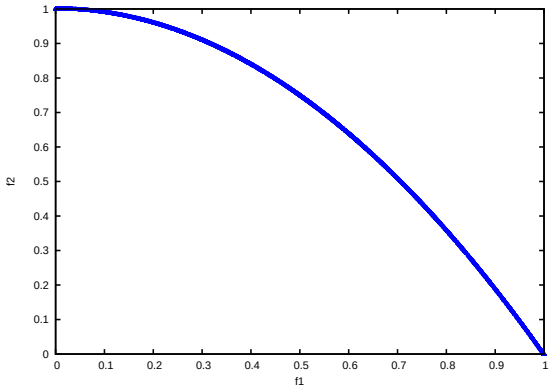


(d) POF^* for FDA3

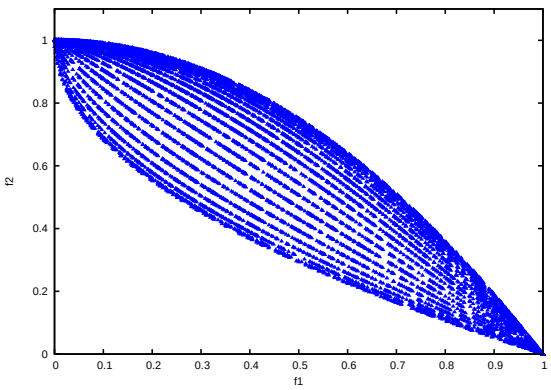


(e) POF^* for FDA3$_{Camara}$

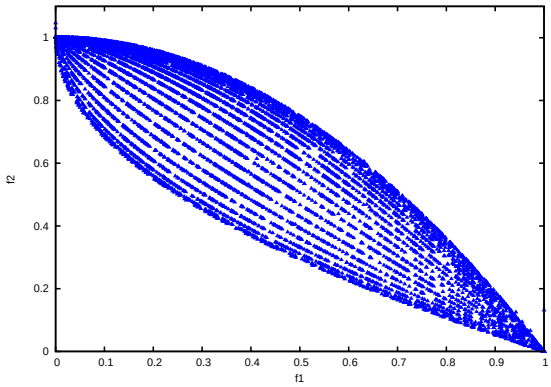
Figure 10.4: POF^* for FDA functions of DVEPSO using *ra-t* for $n_t = 10$ and $\tau_t = 10$



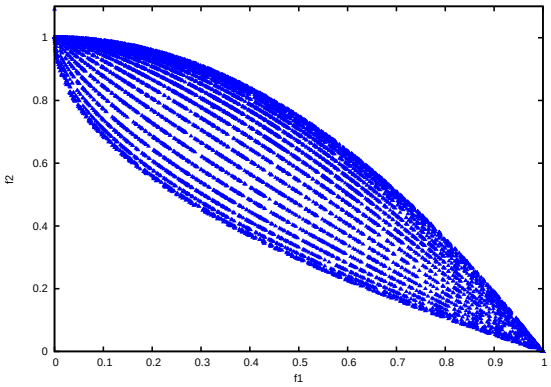
(a) POF^* for DIMP2



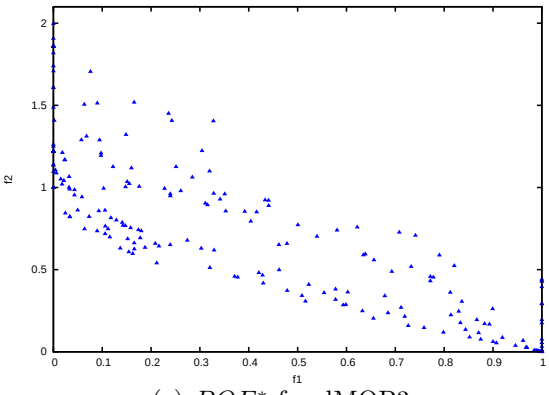
(b) POF^* for dMOP2



(c) POF^* for dMOP2$_{iso}$

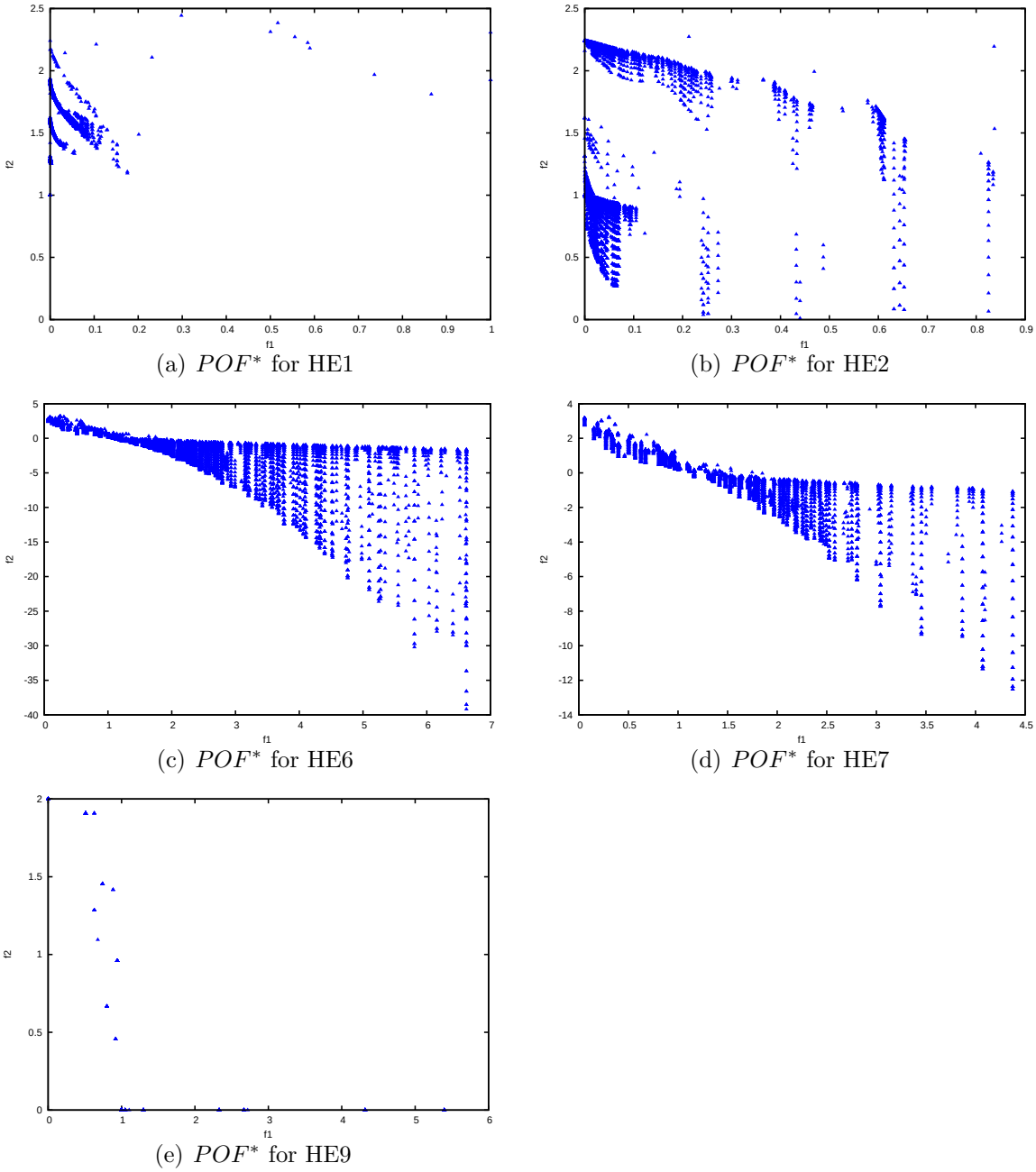


(d) POF^* for dMOP2$_{dec}$



(e) POF^* for dMOP3

Figure 10.5: POF^* for DIMP2 and dMOP2 functions of DVEPSO using *ra-t* for $n_t = 10$ and $\tau_t = 10$



(a) POF^* for HE1

(b) POF^* for HE2

(c) POF^* for HE6

(d) POF^* for HE7

(e) POF^* for HE9

Figure 10.6: POF^* for HE functions of DVEPSO using *ra-t* for $n_t = 10$ and $\tau_t = 10$

General Observations

This section discusses general observations that were made with regards to the performance of the knowledge sharing approaches.

The second best overall performing knowledge sharing approach, *ra-t*, generally performed well for all DMOOPs. However, it did struggle with DIMP2 and FDA3. The wins and losses for DIMP2 are presented in Table 10.28. When solving DIMP2, *ra-t* performed well with regards to *acc* and *stab*. However, for NS it obtained the lowest rank. The number of losses awarded for NS to *ra-t*, caused *ra-t* to obtain the second lowest overall rank for DIMP2.

Table 10.28: Wins and Losses of DIMP2 for various knowledge sharing strategies

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
10	10	*acc*	Wins	1	1	0	3
10	10	*acc*	Losses	1	1	3	0
10	10	*acc*	Diff	0	0	-3	3
10	10	*acc*	Rank	2	2	4	**1**
10	25	*acc*	Wins	1	1	0	3
10	25	*acc*	Losses	1	1	3	0
10	25	*acc*	Diff	0	0	-3	3
10	25	*acc*	Rank	2	2	4	**1**
10	50	*acc*	Wins	1	1	0	3
10	50	*acc*	Losses	1	1	3	0
10	50	*acc*	Diff	0	0	-3	3
10	50	*acc*	Rank	2	2	4	**1**
1	10	*acc*	Wins	1	1	0	3
1	10	*acc*	Losses	1	1	3	0
1	10	*acc*	Diff	0	0	-3	3
1	10	*acc*	Rank	2	2	4	**1**
20	10	*acc*	Wins	1	1	0	3
20	10	*acc*	Losses	1	1	3	0
20	10	*acc*	Diff	0	0	-3	3
20	10	*acc*	Rank	2	2	4	**1**
all	all	*acc*	Wins	5	5	0	15
all	all	*acc*	Losses	5	5	15	0
all	all	*acc*	Diff	0	0	-15	15
all	all	*acc*	Rank	2	2	4	**1**
10	10	*stab*	Wins	1	1	0	3
10	10	*stab*	Losses	1	1	3	0
10	10	*stab*	Diff	0	0	-3	3
10	10	*stab*	Rank	2	2	4	**1**
10	25	*stab*	Wins	1	1	0	3

Continued on next page

n_t	τ_t	PM	Results	knowledge sharing strategy			
				ra-g	ra-t	ri-g	ri-t
10	25	*stab*	Losses	1	1	3	0
10	25	*stab*	Diff	0	0	-3	3
10	25	*stab*	Rank	2	2	4	**1**
10	50	*stab*	Wins	1	1	0	3
10	50	*stab*	Losses	1	1	3	0
10	50	*stab*	Diff	0	0	-3	3
10	50	*stab*	Rank	2	2	4	**1**
1	10	*stab*	Wins	1	1	0	3
1	10	*stab*	Losses	1	1	3	0
1	10	*stab*	Diff	0	0	-3	3
1	10	*stab*	Rank	2	2	4	**1**
20	10	*stab*	Wins	1	1	0	3
20	10	*stab*	Losses	1	1	3	0
20	10	*stab*	Diff	0	0	-3	3
20	10	*stab*	Rank	2	2	4	**1**
all	all	*stab*	Wins	5	5	0	15
all	all	*stab*	Losses	5	5	15	0
all	all	*stab*	Diff	0	0	-15	15
all	all	*stab*	Rank	2	2	4	**1**
10	10	*NS*	Wins	1	2	0	1
10	10	*NS*	Losses	0	0	3	1
10	10	*NS*	Diff	1	2	-3	0
10	10	*NS*	Rank	2	**1**	4	3
10	25	*NS*	Wins	1	2	0	1
10	25	*NS*	Losses	0	0	3	1
10	25	*NS*	Diff	1	2	-3	0
10	25	*NS*	Rank	2	**1**	4	3
10	50	*NS*	Wins	1	2	0	1
10	50	*NS*	Losses	0	0	3	1
10	50	*NS*	Diff	1	2	-3	0
10	50	*NS*	Rank	2	**1**	4	3
1	10	*NS*	Wins	1	1	0	1
1	10	*NS*	Losses	0	0	3	0
1	10	*NS*	Diff	1	1	-3	1
1	10	*NS*	Rank	**1**	**1**	4	**1**
20	10	*NS*	Wins	1	1	0	1
20	10	*NS*	Losses	0	0	3	0
20	10	*NS*	Diff	1	1	-3	1
20	10	*NS*	Rank	**1**	**1**	4	**1**
all	all	*NS*	Wins	5	8	0	5
all	all	*NS*	Losses	0	0	15	3
all	all	*NS*	Diff	5	8	-15	2
all	all	*NS*	Rank	2	**1**	4	3
10	10	all	Wins	3	4	0	7
10	10	all	Losses	2	2	9	1

Continued on next page

n_t	τ_t	PM	Results	knowledge sharing strategy			
				ra-g	ra-t	ri-g	ri-t
10	10	all	Diff	1	2	-9	6
10	10	all	Rank	3	2	4	**1**
10	25	all	Wins	3	4	0	7
10	25	all	Losses	2	2	9	1
10	25	all	Diff	1	2	-9	6
10	25	all	Rank	3	2	4	**1**
10	50	all	Wins	3	4	0	7
10	50	all	Losses	2	2	9	1
10	50	all	Diff	1	2	-9	6
10	50	all	Rank	3	2	4	**1**
1	10	all	Wins	3	3	0	7
1	10	all	Losses	2	2	9	0
1	10	all	Diff	1	1	-9	7
1	10	all	Rank	2	2	4	**1**
20	10	all	Wins	3	3	0	7
20	10	all	Losses	2	2	9	0
20	10	all	Diff	1	1	-9	7
20	10	all	Rank	2	2	4	**1**
all	all	all	Wins	15	18	0	35
all	all	all	Losses	10	10	45	3
all	all	all	Diff	5	8	-45	32
all	all	all	Rank	3	2	4	**1**

Another DMOOP that *ra-t* struggled with, is FDA3. The wins and losses obtained by the various knowledge sharing strategies for FDA3 are presented in Table 10.29. The worst rank was obtained by *ra-g* and *ra-t* for *acc*. For *stab*, *ra-t* was awarded the second lowest rank. With regards to NS, both *ra-g* and *ra-t* performed the best. However, the overall wins and losses for FDA3 measured over all performance measures and all n_t-τ_t combinations, lead to *ra-t* obtaining the best rank for FDA3.

Table 10.29: Wins and Losses of FDA3 for various knowledge sharing strategies

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
10	10	*acc*	Wins	0	1	0	2
10	10	*acc*	Losses	2	1	0	0
10	10	*acc*	Diff	-2	0	0	2
10	10	*acc*	Rank	4	2	2	**1**
10	25	*acc*	Wins	0	0	0	2
10	25	*acc*	Losses	1	1	0	0
10	25	*acc*	Diff	-1	-1	0	2
10	25	*acc*	Rank	3	3	2	**1**

Continued on next page

n_t	τ_t	PM	Results	knowledge sharing strategy			
				ra-g	ra-t	ri-g	ri-t
10	50	*acc*	Wins	1	0	2	2
10	50	*acc*	Losses	2	3	0	0
10	50	*acc*	Diff	-1	-3	2	2
10	50	*acc*	Rank	3	4	**1**	**1**
1	10	*acc*	Wins	1	1	0	1
1	10	*acc*	Losses	0	0	3	0
1	10	*acc*	Diff	1	1	-3	1
1	10	*acc*	Rank	**1**	**1**	4	**1**
all	all	*acc*	Wins	2	2	2	7
all	all	*acc*	Losses	5	5	3	0
all	all	*acc*	Diff	-3	-3	-1	7
all	all	*acc*	Rank	3	3	2	**1**
10	10	*stab*	Wins	1	1	0	1
10	10	*stab*	Losses	0	0	3	0
10	10	*stab*	Diff	1	1	-3	1
10	10	*stab*	Rank	**1**	**1**	4	**1**
10	25	*stab*	Wins	1	1	0	1
10	25	*stab*	Losses	0	0	3	0
10	25	*stab*	Diff	1	1	-3	1
10	25	*stab*	Rank	**1**	**1**	4	**1**
10	50	*stab*	Wins	1	2	0	2
10	50	*stab*	Losses	2	0	3	0
10	50	*stab*	Diff	-1	2	-3	2
10	50	*stab*	Rank	3	**1**	4	**1**
20	10	*stab*	Wins	1	1	0	1
20	10	*stab*	Losses	0	0	3	0
20	10	*stab*	Diff	1	1	-3	1
20	10	*stab*	Rank	**1**	**1**	4	**1**
all	all	*stab*	Wins	4	5	0	5
all	all	*stab*	Losses	2	0	12	0
all	all	*stab*	Diff	2	5	-12	5
all	all	*stab*	Rank	3	**1**	4	**1**
10	10	*NS*	Wins	0	0	3	0
10	10	*NS*	Losses	1	1	0	1
10	10	*NS*	Diff	-1	-1	3	-1
10	10	*NS*	Rank	2	2	**1**	2
10	25	*NS*	Wins	0	0	3	0
10	25	*NS*	Losses	1	1	0	1
10	25	*NS*	Diff	-1	-1	3	-1
10	25	*NS*	Rank	2	2	**1**	2
10	50	*NS*	Wins	0	0	3	0
10	50	*NS*	Losses	1	1	0	1
10	50	*NS*	Diff	-1	-1	3	-1
10	50	*NS*	Rank	2	2	**1**	2
1	10	*NS*	Wins	0	0	3	0

Continued on next page

n_t	τ_t	PM	Results	knowledge sharing strategy			
				ra-g	ra-t	ri-g	ri-t
1	10	NS	Losses	1	1	0	1
1	10	NS	Diff	-1	-1	3	-1
1	10	NS	Rank	2	2	**1**	2
20	10	NS	Wins	0	0	3	0
20	10	NS	Losses	1	1	0	1
20	10	NS	Diff	-1	-1	3	-1
20	10	NS	Rank	2	2	**1**	2
all	all	NS	Wins	10	10	0	0
all	all	NS	Losses	0	0	10	10
all	all	NS	Diff	10	10	-10	-10
all	all	NS	Rank	**1**	**1**	3	3
10	10	all	Wins	3	4	0	3
10	10	all	Losses	2	1	5	2
10	10	all	Diff	1	3	-5	1
10	10	all	Rank	2	**1**	4	2
10	25	all	Wins	3	3	0	3
10	25	all	Losses	1	1	5	2
10	25	all	Diff	2	2	-5	1
10	25	all	Rank	**1**	**1**	4	3
10	50	all	Wins	4	4	2	4
10	50	all	Losses	4	3	5	2
10	50	all	Diff	0	1	-3	2
10	50	all	Rank	3	2	4	**1**
1	10	all	Wins	3	3	0	1
1	10	all	Losses	0	0	5	2
1	10	all	Diff	3	3	-5	-1
1	10	all	Rank	**1**	**1**	4	3
20	10	all	Wins	3	3	0	1
20	10	all	Losses	0	0	5	2
20	10	all	Diff	3	3	-5	-1
20	10	all	Rank	**1**	**1**	4	3
all	all	all	Wins	16	17	2	12
all	all	all	Losses	7	5	25	10
all	all	all	Diff	9	12	-23	2
all	all	all	Rank	2	**1**	4	3

Table 10.30 presents the wins and losses for HE6. It is interesting to note that for HE6 there was no statistical significant difference in the performance of the various knowledge sharing approaches for *acc* and *stab* for most n_t-τ_t combinations. Furthermore, there was no statistical significant difference for NS for most of the n_t-τ_t combinations. A similar trend was observed for HE7.

Table 10.30: Wins and Losses of HE6 for various knowledge sharing strategies

n_t	τ_t	PM	Results	knowledge sharing strategies			
				ra-g	ra-t	ri-g	ri-t
1	10	NS	Wins	1	0	0	0
1	10	NS	Losses	0	0	0	1
1	10	NS	Diff	1	0	0	-1
1	10	NS	Rank	**1**	2	2	4
all	all	NS	Wins	0	0	0	1
all	all	NS	Losses	1	0	0	0
all	all	NS	Diff	-1	0	0	1
all	all	NS	Rank	4	2	2	**1**
1	10	all	Wins	0	0	0	1
1	10	all	Losses	1	0	0	0
1	10	all	Diff	-1	0	0	1
1	10	all	Rank	4	2	2	**1**
all	all	all	Wins	0	0	0	1
all	all	all	Losses	1	0	0	0
all	all	all	Diff	-1	0	0	1
all	all	all	Rank	4	2	2	**1**

The next section discusses results obtained by various responses to changes in the environment.

10.2.3 Responses to Change

This section investigates the influence of various responses to environmental changes on the performance of DVEPSO. When a change occurrs, both the particles in the sub-swarms and the archive that stores the non-dominated solutions have to respond to the change in an appropriate manner.

Change Response Strategies applied to Particles

This section discusses results obtained by various responses to changes in the environment applied to particles of the sub-swarms. The results are discussed with regards to each performance measure and each n_t-τ_t combination. Results obtained for DMOOPs of Type I, II and III are also presented. Furthermore, the overall performance of each response is discussed and general observations are highlighted. The wins and losses of the various approaches to respond to environment changes are presented in Tables 10.31 to 10.47. In Tables 10.31 to 10.47, ri and re indicate re-initialisation or re-evaluation of particles respectively, c and a indicate whether the response is applied to only the

sub-swarm(s) whose objective function changed or all sub-swarms, and 10, 20 and 30 indicate the percentage of particles that is re-initialised.

Results with regards to Performance Measures

This section discusses the results obtained by the responses applied to the particles for the various performance measures. The wins and losses with regards to the performance measures over all n_t-τ_t combinations are presented in Table 10.31.

Table 10.31: Overall wins and losses for various performance measures obtained by various change response strategies applied to the particles

$\mathbf{n_t}$	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	*acc*	Wins	163	103	115	112	112	112	43	69	42	71
all	all	*acc*	Losses	24	88	71	73	78	74	139	134	150	111
all	all	*acc*	Diff	139	15	44	39	34	38	-96	-65	-108	-40
all	all	*acc*	Rank	**1**	6	2	3	5	4	9	8	10	7
all	all	*stab*	Wins	113	63	39	56	30	53	29	22	35	33
all	all	*stab*	Losses	6	58	51	24	61	35	46	75	53	64
all	all	*stab*	Diff	107	5	-12	32	-31	18	-17	-53	-18	-31
all	all	*stab*	Rank	**1**	4	5	2	8	3	6	10	7	8
all	all	*NS*	Wins	108	99	90	89	109	68	70	96	85	110
all	all	*NS*	Losses	68	90	105	98	50	104	128	83	109	89
all	all	*NS*	Diff	40	9	-15	-9	59	-36	-58	13	-24	21
all	all	*NS*	Rank	2	5	7	6	**1**	9	10	4	8	3

The following are observed:

- The best performance for *acc* was obtained by *ri-c*-30 and the worst performance by *reu-c*.
- For *stab*, *ri-c*-30 again obtained the best rank and *re-a* obtained the worst rank.
- Measured against *NS*, *ri-c*-10 performed the best and *re-c* performed the worst.
- Both *re-c* and *reu-c* were awarded more losses than wins for all performance measures. Three other approaches, *ri-c*-20, *re-a* and *reu-a*, obtained more losses than wins for two of the three performance measures.

Results with regards to Various Frequencies and Severities of Change

The results obtained by the responses applied to the particles for the various environment types are discussed in this section. Table 10.32 presents the wins and losses for the various

n_t-τ_t combinations.

Table 10.32: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	all	Wins	92	70	47	61	57	49	25	48	41	51
10	10	all	Losses	26	55	59	40	39	62	80	60	71	49
10	10	all	Diff	66	15	-12	21	18	-13	-55	-12	-30	2
10	10	all	Rank	**1**	4	6	2	3	8	10	6	9	5
10	25	all	Wins	72	32	35	43	46	34	22	25	21	23
10	25	all	Losses	5	40	39	18	26	33	53	49	51	39
10	25	all	Diff	67	-8	-4	25	20	1	-31	-24	-30	-16
10	25	all	Rank	**1**	6	5	2	3	4	10	8	9	7
10	50	all	Wins	59	32	47	34	31	30	21	23	21	27
10	50	all	Losses	12	21	21	41	23	32	49	38	44	44
10	50	all	Diff	47	11	26	-7	8	-2	-28	-15	-23	-17
10	50	all	Rank	**1**	3	2	6	4	5	10	7	9	8
1	10	all	Wins	95	73	67	56	84	71	47	58	44	63
1	10	all	Losses	28	80	57	67	51	53	70	85	86	81
1	10	all	Diff	67	-7	10	-11	33	18	-23	-27	-42	-18
1	10	all	Rank	**1**	5	4	6	2	3	8	9	10	7
20	10	all	Wins	66	58	48	63	33	49	27	33	35	50
20	10	all	Losses	27	40	51	29	50	33	61	60	60	51
20	10	all	Diff	39	18	-3	34	-17	16	-34	-27	-25	-1
20	10	all	Rank	**1**	3	6	2	7	4	10	9	8	5

With regards to the various environment types, the following observations are made:

- For all n_t-τ_t combinations*ri-c*-30 performed the best.
- *re-c* performed the worst in all environments, except severely changing environments ($n_t = 1$ and $\tau_t = 10$).
- For severely changing environments, *reu-c* obtained the worst rank.
- Three approaches, *re-c*, *re-a* and *reu-c*, performed poorly, obtaining more losses than wins for all n_t-τ_t combinations.

Results for Various Dynamic Multi-objective Optimisation Problem Types

This section discusses the results obtained by the various response approaches for solving DMOOPs of Type I, II or III respectively.

Type I DMOOPs

This section discusses the results obtained for Type I DMOOPs. The wins and losses

obtained by the various change response approaches for the performance measures over all n_t-τ_t combinations are presented in Table 10.33.

Table 10.33: Overall wins and losses for various performance measures obtained by various change response strategies applied to the particles solving Type I DMOOPs

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Particle response stategies**									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	*acc*	Wins	10	13	10	14	9	10	3	4	0	1
all	all	*acc*	Losses	0	0	4	0	6	0	19	15	17	13
all	all	*acc*	Diff	10	13	6	14	3	10	-16	-11	-17	-12
all	all	*acc*	Rank	3	2	5	**1**	6	3	9	7	10	8
all	all	*stab*	Wins	13	13	11	12	11	11	0	0	0	0
all	all	*stab*	Losses	0	0	1	0	0	0	18	18	21	13
all	all	*stab*	Diff	13	13	10	12	11	11	-18	-18	-21	-13
all	all	*stab*	Rank	**1**	**1**	6	3	4	4	8	8	10	7
all	all	*NS*	Wins	1	0	3	0	8	1	24	15	21	16
all	all	*NS*	Losses	16	19	10	17	7	14	0	6	0	0
all	all	*NS*	Diff	-15	-19	-7	-17	1	-13	24	9	21	16
all	all	*NS*	Rank	8	10	6	9	5	7	**1**	4	2	3

The following observations are made:

- The best performing approach for *acc* was *ri-a*-20 and *reu-c* performed the worst.
- The best performance for *stab* was obtained by both *ri-c*-30 and *ri-a*-30. The worst performing approach was *reu-c*.
- With regards to *NS*, *re-c* obtained the best rank and *ri-a*-30 the worst rank.
- Four approaches, *re-c*, *re-a*, *reu-c* and *reu-a*, obtained more losses than wins for two performance measures. All of these four approaches re-evaluate the particles after a change in the environment occured. Therefore, the re-evaluation approaches performed poorly. On the other hand, all re-initialisation approaches performed well.

Table 10.34 presents the wins and losses for the n_t-τ_t combinations.
With regards to the various environment types, the following are observed:

- The best performance for $n_t = 10$ and $\tau_t = 10$, and $n_t = 10$ and $\tau_t = 50$ was obtained by *ri-c*-10. The worst performance for $n_t = 10$ and $\tau_t = 10$ was achieved by *reu-a*.
- Three approaches, *ri-c*-10, *re-c* and *reu-a*, performed the best for $n_t = 10$ and $\tau_t = 50$. The worst rank was obtained by *ri-c*-30 and *ri-a*-20.

- For $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, the best performing approach was *ri-c*-30, with *rea-a* and *reau-a* performing the worst respectively.
- In gradually changing environments, there was almost no difference in the performance of the various response approaches. The best performing approach was *ri-a*-20 and the worst performing approaches were *ri-c*-10 and *ri-c*-20.
- More losses than wins were awarded to *re-a* and *reu-c* for three of the five n_t-τ_t combinations. Therefore, similar to the wins and losses with regards to the performance measures, the re-evaluation approaches performed poorly for most of the n_t-τ_t combinations.

Table 10.34: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the particles solving Type I DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	all	Wins	13	16	13	15	13	13	13	10	11	4
10	10	all	Losses	7	7	9	7	3	7	24	19	20	18
10	10	all	Diff	6	9	4	8	10	6	-11	-9	-9	-14
10	10	all	Rank	4	2	6	3	**1**	4	9	7	7	10
10	25	all	Wins	4	3	2	3	2	2	1	0	0	0
10	25	all	Losses	0	0	0	0	0	0	6	7	3	1
10	25	all	Diff	4	3	2	3	2	2	-5	-7	-3	-1
10	25	all	Rank	**1**	2	4	2	4	4	9	10	8	7
10	50	all	Wins	0	1	3	0	5	0	5	0	3	5
10	50	all	Losses	5	5	0	5	0	3	0	4	0	0
10	50	all	Diff	-5	-4	3	-5	5	-3	5	-4	3	5
10	50	all	Rank	9	7	4	9	**1**	6	**1**	7	4	**1**
1	10	all	Wins	7	6	6	6	8	7	8	9	7	8
1	10	all	Losses	4	7	5	5	9	4	7	9	15	7
1	10	all	Diff	3	-1	1	1	-1	3	1	0	-8	1
1	10	all	Rank	**1**	8	3	3	8	**1**	3	7	10	3
20	10	all	Wins	0	0	0	2	0	0	0	0	0	0
20	10	all	Losses	0	0	1	0	1	0	0	0	0	0
20	10	all	Diff	0	0	-1	2	-1	0	0	0	0	0
20	10	all	Rank	2	2	9	**1**	9	2	2	2	2	2

The wins and losses of the response approaches measured over all Type I DMOOPs, all performance measures and all n_t-τ_t combinations are presented in Table 10.35.

Table 10.35: Overall wins and losses obtained by various change response strategies applied to the particles solving Type I DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	ri-a-30	ri-c-20	ri-a-20	ri-c-10	ri-a-10	re-c	re-a	reu-c	reu-a
all	all	all	Wins	24	26	24	26	28	22	27	19	21	17
all	all	all	Losses	16	19	15	17	13	14	37	39	38	26
all	all	all	Diff	8	7	9	9	15	8	-10	-20	-17	-9
all	all	all	Rank	4	6	2	2	**1**	4	8	10	9	7

Measuring the approaches' performance over all performance measures and all n_t-τ_t combinations for Type I DMOOPs, the best rank was obtained by *ri-c*-10, with *rea-a* obtaining the worst rank. All re-evaluation approaches performed poorly and were outperformed by the re-initialisation approaches.

Type II DMOOPs

This section discusses the results for Type II DMOOPs that were obtained by the various response approaches applied to the particles. The wins and losses obtained by the various response approaches for the performance measures over all n_t-τ_t combinations are presented in Table 10.36.

Table 10.36: Overall wins and losses for various performance measures obtained by various change response strategies applied to the particles solving Type II DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	ri-a-30	ri-c-20	ri-a-20	ri-c-10	ri-a-10	re-c	re-a	reu-c	reu-a
all	all	*acc*	Wins	108	76	60	79	51	73	19	25	7	37
all	all	*acc*	Losses	8	27	43	21	49	38	85	87	106	71
all	all	*acc*	Diff	100	49	17	58	2	35	-66	-62	-99	-34
all	all	*acc*	Rank	**1**	3	5	2	6	4	9	8	10	7
all	all	*stab*	Wins	94	49	18	41	14	36	25	9	25	19
all	all	*stab*	Losses	4	24	47	15	54	30	23	55	29	49
all	all	*stab*	Diff	90	25	-29	26	-40	6	2	-46	-4	-30
all	all	*stab*	Rank	**1**	3	7	2	9	4	5	10	6	8
all	all	*NS*	Wins	53	37	39	50	41	42	5	24	13	38
all	all	*NS*	Losses	18	34	33	13	17	21	70	40	65	31
all	all	*NS*	Diff	35	3	6	37	24	21	-65	-16	-52	7
all	all	*NS*	Rank	2	7	6	**1**	3	4	10	8	9	5

The following observations are made:

- The best performance for *acc* was obtained by *ri-c*-30 and the worst by *reu-c*. The re-evaluation approaches were awarded the four worst ranks.

- For *stab*, *ri-c*-30 performed the best and *re-a* obtained the worst performance.
- With regards to NS, *ri-a*-20 obtained the best performance and *re-c* the worst.
- More losses than wins were awarded to *re-a* for all performance measures.

Table 10.37 presents the wins and losses with regards to the various n_t-τ_t combinations. With regards to the various environment types, the following are observed:

- In all environments *ri-c*-30 performed the best.
- The worst performance for $n_t = 10$ and $\tau_t = 10$ was achieved by *reu-c*.
- For $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$, *re-a* and *reu-c* performed the worst respectively.
- In a severely changing environment *reu-c* obtained the worst rank. However, for $n_t = 20$ and $\tau_t = 10$ causing a gradually changing environment *re-c* performed the worst.
- Three approaches obtained more losses than wins for all n_t-τ_t combinations, namely *re-c*, *re-a* and *reu-c*.

Table 10.37: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the particles solving Type II DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	all	Wins	60	44	28	39	23	32	10	11	8	33
10	10	all	Losses	10	21	28	13	28	32	43	44	47	22
10	10	all	Diff	50	23	0	26	-5	0	-33	-33	-39	11
10	10	all	Rank	**1**	3	5	2	7	5	8	8	10	4
10	25	all	Wins	51	27	13	31	16	23	14	12	15	16
10	25	all	Losses	1	25	32	6	25	19	29	33	26	22
10	25	all	Diff	50	2	-19	25	-9	4	-15	-21	-11	-6
10	25	all	Rank	**1**	4	9	2	6	3	8	10	7	5
10	50	all	Wins	44	16	15	18	15	23	7	7	3	11
10	50	all	Losses	4	6	14	8	10	11	25	26	36	19
10	50	all	Diff	40	10	1	10	5	12	-18	-19	-33	-8
10	50	all	Rank	**1**	3	6	3	5	2	8	9	10	7
1	10	all	Wins	52	44	35	42	34	40	14	22	9	22
1	10	all	Losses	7	23	27	17	31	19	43	41	53	53
1	10	all	Diff	45	21	8	25	3	21	-29	-19	-44	-31
1	10	all	Rank	**1**	3	5	2	6	3	8	7	10	9
20	10	all	Wins	48	31	26	40	18	33	4	6	10	12
20	10	all	Losses	8	10	22	5	26	8	38	38	38	35
20	10	all	Diff	40	21	4	35	-8	25	-34	-32	-28	-23
20	10	all	Rank	**1**	4	5	2	6	3	10	9	8	7

Table 10.38 presents the wins and losses of the response approaches' performance over all performance measures and all n_t-τ_t combinations solving Type II DMOOPs. From the data, the following are observed:

- The best overall rank was obtained by *ri-c*-30 and the worst by *reu-c*.
- All re-evaluation approaches performed badly, obtaining more losses than wins.
- An average or poor performance was obtained by all *ri-c* approaches, except *ri-c*-30 that obtained the best overall rank. In contrast, all *ri-a* approaches performed well.

Table 10.38: Overall wins and losses obtained by various change response strategies applied to the particles solving Type II DMOOPs

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Particle response stategies**									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	all	Wins	255	162	117	170	106	151	49	58	45	94
all	all	all	Losses	30	85	123	49	120	89	178	182	200	151
all	all	all	Diff	225	77	-6	121	-14	62	-129	-124	-155	-57
all	all	all	Rank	**1**	3	5	2	6	4	9	8	10	7

Type III DMOOPs

This section discusses the results that were obtained by the various response approaches applied to the particles for Type III DMOOPs. The wins and losses obtained by the various response approaches for the performance measures measured over all n_t-τ_t combinations are presented in Table 10.39.

Table 10.39: Overall wins and losses for various performance measures obtained by various change response strategies applied to the particles solving Type II DMOOPs

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Particle response stategies**									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	*acc*	Wins	45	14	45	19	52	29	21	40	35	33
all	all	*acc*	Losses	16	61	24	52	23	36	35	32	27	27
all	all	*acc*	Diff	29	-47	21	-33	29	-7	-14	8	8	6
all	all	*acc*	Rank	**1**	10	3	9	**1**	7	8	4	4	6
all	all	*stab*	Wins	6	1	10	3	5	6	4	13	10	14
all	all	*stab*	Losses	2	34	3	9	7	5	5	2	3	2
all	all	*stab*	Diff	4	-33	7	-6	-2	1	-1	11	7	12
all	all	*stab*	Rank	5	10	3	9	8	6	7	2	3	**1**
all	all	*NS*	Wins	54	62	48	39	60	25	41	57	51	56
all	all	*NS*	Losses	34	37	62	68	26	69	58	37	44	58
all	all	*NS*	Diff	20	25	-14	-29	34	-44	-17	20	7	-2
all	all	*NS*	Rank	3	2	7	9	**1**	10	8	3	5	6

The following are observed with regards to the various performance measures:

- The best performance for *acc* were obtained by *ri-c*-30 and *ri-c*-10. The worst rank was achieved by *ri-a*-30. With the exception of *re-c*, all re-evaluation approaches performed well.
- For *stab* the best performing approach was *reu-a* and *ri-c*-10 performed the worst. All re-evaluation approaches obtained the top ranks, except *re-c* which performed poorly.
- With regards to NS, the best performance was obtained by *ri-c*-10 and the worst by *ri-a*-10.

Table 10.40 presents the wins and losses for the n_t-τ_t combinations.

Table 10.40: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the particles solving Type III DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	all	Wins	24	16	11	14	26	9	8	32	27	18
10	10	all	Losses	13	31	25	24	9	27	25	4	12	15
10	10	all	Diff	11	-15	-14	-10	17	-18	-17	28	15	3
10	10	all	Rank	4	8	7	6	2	10	9	**1**	3	5
10	25	all	Wins	17	2	20	9	28	9	7	13	6	7
10	25	all	Losses	4	15	7	12	1	14	18	9	22	16
10	25	all	Diff	13	-13	13	-3	27	-5	-11	4	-16	-9
10	25	all	Rank	2	9	2	5	**1**	6	8	4	10	7
10	50	all	Wins	15	15	29	16	11	7	9	16	15	11
10	50	all	Losses	3	10	7	28	13	18	24	8	8	25
10	50	all	Diff	12	5	22	-12	-2	-11	-15	8	7	-14
10	50	all	Rank	2	5	**1**	8	6	7	10	3	4	9
1	10	all	Wins	36	23	26	8	42	24	25	27	28	33
1	10	all	Losses	17	50	25	45	11	30	20	35	18	21
1	10	all	Diff	19	-27	1	-37	31	-6	5	-8	10	12
1	10	all	Rank	2	9	6	10	**1**	7	5	8	4	3
20	10	all	Wins	13	21	17	14	10	11	17	22	20	34
20	10	all	Losses	15	26	25	20	22	21	11	15	14	10
20	10	all	Diff	-2	-5	-8	-6	-12	-10	6	7	6	24
20	10	all	Rank	5	6	8	7	10	9	3	2	3	**1**

With regards to the different environments, the following are observed:

- For $n_t = 10$ and $\tau_t = 10$, *re-a* performed the best and *ri-a*-10 the worst.
- For both $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, *ri-c*-10 obtained the best

rank, with *reu-c* and *ri-a*-20 performing the worst respectively.

- In slowly changing environments ($\tau_t = 50$), *ri-c*-20 performed the best, and *re-c* the worst.
- In gradually changing environments ($n_t = 20$), *reu-a* obtained the best rank and *ri-c*-20 the worst rank.

The wins and losses for Type III benchmark functions are presented in Table 10.41. For Type III DMOOPs the best performance measured over all performance measures and n_t-τ_t combinations was obtained by *ri-c*-10. The worst overal rank was obtained by *ri-a*-20.

Table 10.41: Overall wins and losses obtained by various change response strategies applied to the particles solving Type III DMOOPs

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	all	Wins	105	77	103	61	117	60	66	110	96	103
all	all	all	Losses	52	132	89	129	56	110	98	71	74	87
all	all	all	Diff	53	-55	14	-68	61	-50	-32	39	22	16
all	all	all	Rank	2	9	6	10	**1**	8	7	3	4	5

Overall Performance

This section discusses the overall performance of the responses applied to the particles. The wins and losses over all DMOOPs, n_t-τ_t combinations and performance measures are presented in Table 10.47.

Table 10.42: Overall wins and losses obtained by various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	all	Wins	384	265	244	257	251	233	142	187	162	214
all	all	all	Losses	98	236	227	195	189	213	313	292	312	264
all	all	all	Diff	286	29	17	62	62	20	-171	-105	-150	-50
all	all	all	Rank	**1**	4	6	2	2	5	10	8	9	7

The following observations are made:

- The approach that obtained the best overall rank was *ri-c*-30, completely outperforming the other approaches with 286 more wins than losses. The approach that

ranked second obtained 62 more wins than losses.

- With the exception of *ri-c*-10, all re-initialisation approaches performed well. On the other hand, all re-evaluation approaches performed poorly.

The POF^*s found by *ri-c*-30 for $n_t = 10$ and $\tau_t = 10$ are illustrated in Figures 10.7 to 10.9. The same trend as the POF^*s found by *cl* was observed.

General Observations

This section discusses general observations that were made with regards to the performance of the responses applied to the particles of DVEPSO's sub-swarms.

In general, the re-initialisation approaches outperformed the re-evaluation approaches. The re-evaluation approaches performed well for DIMP2, FDA2_{Camara}, FDA3, FDA1_{Zhou}, HE1 and HE2. However, for DIMP2 and the FDA DMOOPs there was no statistical significant difference in the performance measure values of the various approaches for most of the performance measures. The wins and losses for DIMP2 are presented in Table 10.43.

Table 10.43: Wins and Losses of DIMP2 for various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	*acc*	Wins	0	1	0	0	0	0	1	1	0	0
10	10	*acc*	Losses	0	0	3	0	0	0	0	0	0	0
10	10	*acc*	Diff	0	1	-3	0	0	0	1	1	0	0
10	10	*acc*	Rank	4	**1**	10	4	4	4	**1**	**1**	4	4
10	25	*acc*	Wins	0	0	0	0	0	0	1	0	0	0
10	25	*acc*	Losses	0	0	0	0	0	0	0	1	0	0
10	25	*acc*	Diff	0	0	0	0	0	0	1	-1	0	0
10	25	*acc*	Rank	2	2	2	2	2	2	**1**	10	2	2
10	50	*acc*	Wins	0	1	0	0	0	0	0	0	0	0
10	50	*acc*	Losses	0	0	0	0	0	0	0	1	0	0
10	50	*acc*	Diff	0	1	0	0	0	0	0	-1	0	0
10	50	*acc*	Rank	2	**1**	2	2	2	2	2	10	2	2
20	10	*acc*	Wins	0	0	0	2	0	0	0	0	0	0
20	10	*acc*	Losses	0	0	1	0	1	0	0	0	0	0
20	10	*acc*	Diff	0	0	-1	2	-1	0	0	0	0	0
20	10	*acc*	Rank	2	2	9	**1**	9	2	2	2	2	2
all	all	*acc*	Wins	0	2	0	2	0	0	2	1	0	0
all	all	*acc*	Losses	0	0	4	0	1	0	0	2	0	0
all	all	*acc*	Diff	0	2	-4	2	-1	0	2	-1	0	0

Continued on next page

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	*acc*	Rank	4	**1**	10	**1**	8	4	**1**	8	4	4
10	10	*stab*	Wins	0	1	0	0	0	0	0	0	0	0
10	10	*stab*	Losses	0	0	1	0	0	0	0	0	0	0
10	10	*stab*	Diff	0	1	-1	0	0	0	0	0	0	0
10	10	*stab*	Rank	2	**1**	10	2	2	2	2	2	2	2
all	all	*stab*	Wins	0	1	0	0	0	0	0	0	0	0
all	all	*stab*	Losses	0	0	1	0	0	0	0	0	0	0
all	all	*stab*	Diff	0	1	-1	0	0	0	0	0	0	0
all	all	*stab*	Rank	2	**1**	10	2	2	2	2	2	2	2
10	10	all	Wins	0	2	0	0	0	0	1	1	0	0
10	10	all	Losses	0	0	4	0	0	0	0	0	0	0
10	10	all	Diff	0	2	-4	0	0	0	1	1	0	0
10	10	all	Rank	4	**1**	10	4	4	4	2	2	4	4
10	25	all	Wins	0	0	0	0	0	0	1	0	0	0
10	25	all	Losses	0	0	0	0	0	0	0	1	0	0
10	25	all	Diff	0	0	0	0	0	0	1	-1	0	0
10	25	all	Rank	2	2	2	2	2	2	**1**	10	2	2
10	50	all	Wins	0	1	0	0	0	0	0	0	0	0
10	50	all	Losses	0	0	0	0	0	0	0	1	0	0
10	50	all	Diff	0	1	0	0	0	0	0	-1	0	0
10	50	all	Rank	2	**1**	2	2	2	2	2	10	2	2
20	10	all	Wins	0	0	0	2	0	0	0	0	0	0
20	10	all	Losses	0	0	1	0	1	0	0	0	0	0
20	10	all	Diff	0	0	-1	2	-1	0	0	0	0	0
20	10	all	Rank	2	2	9	**1**	9	2	2	2	2	2
all	all	all	Wins	0	3	0	2	0	0	2	1	0	0
all	all	all	Losses	0	0	5	0	1	0	0	2	0	0
all	all	all	Diff	0	3	-5	2	-1	0	2	-1	0	0
all	all	all	Rank	4	**1**	10	2	8	4	2	8	4	4

For the discontinuous functions, HE1 and HE2, the re-evaluation approaches performed well. For HE1, all re-evaluation approaches performed well, except *re-c*. For HE2, all re-evaluation approaches performed well, except *re-a*. The best overall rank for HE2 was obtained by *reu-a*, and three of the re-evaluation approaches obtained a rank in the top six for HE2. Tables 10.44 and 10.45 present the wins and losses for HE1 and HE2 respectively.

Table 10.44: Wins and Losses of HE1 for various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	*acc*	Wins	5	0	5	2	3	5	1	7	7	2
10	10	*acc*	Losses	3	9	1	5	4	2	7	0	0	6

Continued on next page

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	10	*acc*	Diff	2	-9	4	-3	-1	3	-6	7	7	-4
10	10	*acc*	Rank	5	10	3	7	6	4	9	**1**	**1**	8
10	25	*acc*	Wins	1	1	3	1	8	6	1	3	0	1
10	25	*acc*	Losses	2	2	1	2	0	0	4	1	9	4
10	25	*acc*	Diff	-1	-1	2	-1	8	6	-3	2	-9	-3
10	25	*acc*	Rank	5	5	3	5	**1**	2	8	3	10	8
10	50	*acc*	Wins	5	1	8	0	1	1	2	1	1	1
10	50	*acc*	Losses	0	2	0	9	3	2	1	1	1	2
10	50	*acc*	Diff	5	-1	8	-9	-2	-1	1	0	0	-1
10	50	*acc*	Rank	2	6	**1**	10	9	6	3	4	4	6
1	10	*acc*	Wins	5	0	2	0	9	8	5	3	6	4
1	10	*acc*	Losses	3	8	7	8	0	1	2	6	2	5
1	10	*acc*	Diff	2	-8	-5	-8	9	7	3	-3	4	-1
1	10	*acc*	Rank	5	9	8	9	**1**	2	4	7	3	6
20	10	*acc*	Wins	2	0	7	1	2	4	2	7	2	7
20	10	*acc*	Losses	3	9	0	8	3	3	4	0	4	0
20	10	*acc*	Diff	-1	-9	7	-7	-1	1	-2	7	-2	7
20	10	*acc*	Rank	5	10	**1**	9	5	4	7	**1**	7	**1**
all	all	*acc*	Wins	18	2	25	4	23	24	11	21	16	15
all	all	*acc*	Losses	11	30	9	32	10	8	18	8	16	17
all	all	*acc*	Diff	7	-28	16	-28	13	16	-7	13	0	-2
all	all	*acc*	Rank	5	9	**1**	9	3	**1**	8	3	6	7
10	10	*stab*	Wins	1	1	1	1	1	1	0	1	1	1
10	10	*stab*	Losses	0	8	0	0	0	0	1	0	0	0
10	10	*stab*	Diff	1	-7	1	1	1	1	-1	1	1	1
10	10	*stab*	Rank	**1**	10	**1**	**1**	**1**	**1**	9	**1**	**1**	**1**
1	10	*stab*	Wins	2	0	2	1	1	2	2	7	2	7
1	10	*stab*	Losses	1	9	2	2	6	2	2	0	2	0
1	10	*stab*	Diff	1	-9	0	-1	-5	0	0	7	0	7
1	10	*stab*	Rank	3	10	4	8	9	4	4	**1**	4	**1**
all	all	*stab*	Wins	3	1	3	2	2	3	2	8	3	8
all	all	*stab*	Losses	1	17	2	2	6	2	3	0	2	0
all	all	*stab*	Diff	2	-16	1	0	-4	1	-1	8	1	8
all	all	*stab*	Rank	3	10	4	7	9	4	8	**1**	4	**1**
10	10	*NS*	Wins	6	9	0	3	5	0	1	7	8	3
10	10	*NS*	Losses	3	0	8	5	4	7	7	2	1	5
10	10	*NS*	Diff	3	9	-8	-2	1	-7	-6	5	7	-2
10	10	*NS*	Rank	4	**1**	10	6	5	9	8	3	2	6
10	25	*NS*	Wins	6	1	5	1	8	1	3	8	5	0
10	25	*NS*	Losses	2	5	3	6	0	6	5	0	2	9
10	25	*NS*	Diff	4	-4	2	-5	8	-5	-2	8	3	-9
10	25	*NS*	Rank	3	7	5	8	**1**	8	6	**1**	4	10
10	50	*NS*	Wins	2	2	2	0	1	2	0	2	2	0
10	50	*NS*	Losses	0	0	0	6	0	0	0	0	0	7
10	50	*NS*	Diff	2	2	2	-6	1	2	0	2	2	-7
10	50	*NS*	Rank	**1**	**1**	**1**	9	7	**1**	8	**1**	**1**	10
1	10	*NS*	Wins	7	8	2	0	8	5	6	4	0	3
1	10	*NS*	Losses	2	0	7	8	0	4	3	5	8	6

Continued on next page

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	ri-a-30	ri-c-20	ri-a-20	ri-c-10	ri-a-10	re-c	re-a	reu-c	reu-a
1	10	*NS*	Diff	5	8	-5	-8	8	1	3	-1	-8	-3
1	10	*NS*	Rank	3	**1**	8	9	**1**	5	4	6	9	7
20	10	*NS*	Wins	0	9	5	4	1	1	5	4	1	8
20	10	*NS*	Losses	9	0	2	2	6	6	2	4	6	1
20	10	*NS*	Diff	-9	9	3	2	-5	-5	3	0	-5	7
20	10	*NS*	Rank	10	**1**	3	5	7	7	3	6	7	2
all	all	*NS*	Wins	21	29	14	8	23	9	15	25	16	14
all	all	*NS*	Losses	16	5	20	27	10	23	17	11	17	28
all	all	*NS*	Diff	5	24	-6	-19	13	-14	-2	14	-1	-14
all	all	*NS*	Rank	4	**1**	7	10	3	8	6	2	5	8
10	10	all	Wins	12	10	6	6	9	6	2	15	16	6
10	10	all	Losses	6	17	9	10	8	9	15	2	1	11
10	10	all	Diff	6	-7	-3	-4	1	-3	-13	13	15	-5
10	10	all	Rank	3	9	5	7	4	5	10	2	**1**	8
10	25	all	Wins	7	2	8	2	16	7	4	11	5	1
10	25	all	Losses	4	7	4	8	0	6	9	1	11	13
10	25	all	Diff	3	-5	4	-6	16	1	-5	10	-6	-12
10	25	all	Rank	4	6	3	8	**1**	5	6	2	8	10
10	50	all	Wins	7	3	10	0	2	3	2	3	3	1
10	50	all	Losses	0	2	0	15	3	2	1	1	1	9
10	50	all	Diff	7	1	10	-15	-1	1	1	2	2	-8
10	50	all	Rank	2	5	**1**	10	8	5	5	3	3	9
1	10	all	Wins	14	8	6	1	18	15	13	14	8	14
1	10	all	Losses	6	17	16	18	6	7	7	11	12	11
1	10	all	Diff	8	-9	-10	-17	12	8	6	3	-4	3
1	10	all	Rank	2	8	9	10	**1**	2	4	5	7	5
20	10	all	Wins	2	9	12	5	3	5	7	11	3	15
20	10	all	Losses	12	9	2	10	9	9	6	4	10	1
20	10	all	Diff	-10	0	10	-5	-6	-4	1	7	-7	14
20	10	all	Rank	10	5	2	7	8	6	4	3	9	**1**
all	all	all	Wins	42	32	42	14	48	36	28	54	35	37
all	all	all	Losses	28	52	31	61	26	33	38	19	35	45
all	all	all	Diff	14	-20	11	-47	22	3	-10	35	0	-8
all	all	all	Rank	3	9	4	10	2	5	8	**1**	6	7

Table 10.45: Wins and Losses of HE2 for various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	ri-a-30	ri-c-20	ri-a-20	ri-c-10	ri-a-10	re-c	re-a	reu-c	reu-a
10	10	*acc*	Wins	6	0	2	1	6	1	1	6	2	5
10	10	*acc*	Losses	0	9	3	6	0	4	4	0	4	0
10	10	*acc*	Diff	6	-9	-1	-5	6	-3	-3	6	-2	5
10	10	*acc*	Rank	**1**	10	5	9	**1**	7	7	**1**	6	4
10	25	*acc*	Wins	1	0	4	0	4	0	0	0	0	0
10	25	*acc*	Losses	0	0	0	0	0	3	2	2	2	0
10	25	*acc*	Diff	1	0	4	0	4	-3	-2	-2	-2	0

Continued on next page

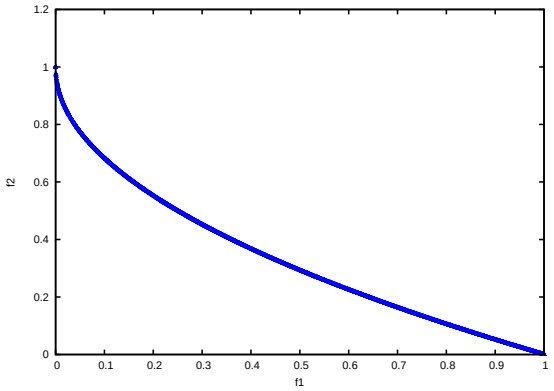
n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
10	25	*acc*	Rank	3	4	**1**	4	**1**	10	7	7	7	4
10	50	*acc*	Wins	1	6	4	6	0	0	1	0	2	1
10	50	*acc*	Losses	2	0	0	0	7	3	3	4	0	2
10	50	*acc*	Diff	-1	6	4	6	-7	-3	-2	-4	2	-1
10	50	*acc*	Rank	5	**1**	3	**1**	10	8	7	9	4	5
1	10	*acc*	Wins	4	0	5	1	9	3	4	2	4	4
1	10	*acc*	Losses	2	9	1	8	0	6	1	7	1	1
1	10	*acc*	Diff	2	-9	4	-7	9	-3	3	-5	3	3
1	10	*acc*	Rank	6	10	2	9	**1**	7	3	8	3	3
20	10	*acc*	Wins	2	0	0	0	0	0	3	0	6	4
20	10	*acc*	Losses	0	2	4	1	4	1	0	3	0	0
20	10	*acc*	Diff	2	-2	-4	-1	-4	-1	3	-3	6	4
20	10	*acc*	Rank	4	7	9	5	9	5	3	8	**1**	2
all	all	*acc*	Wins	14	6	15	8	19	4	9	8	14	14
all	all	*acc*	Losses	4	20	8	15	11	17	10	16	7	3
all	all	*acc*	Diff	10	-14	7	-7	8	-13	-1	-8	7	11
all	all	*acc*	Rank	2	10	4	7	3	9	6	8	4	**1**
10	50	*stab*	Wins	1	0	0	0	0	0	0	0	0	0
10	50	*stab*	Losses	0	0	0	1	0	0	0	0	0	0
10	50	*stab*	Diff	1	0	0	-1	0	0	0	0	0	0
10	50	*stab*	Rank	**1**	2	2	10	2	2	2	2	2	2
1	10	*stab*	Wins	1	0	1	1	1	1	1	3	1	1
1	10	*stab*	Losses	0	9	1	0	1	0	0	0	0	0
1	10	*stab*	Diff	1	-9	0	1	0	1	1	3	1	1
1	10	*stab*	Rank	2	10	8	2	8	2	2	**1**	2	2
20	10	*stab*	Wins	1	0	2	0	2	2	1	2	1	2
20	10	*stab*	Losses	0	8	0	5	0	0	0	0	0	0
20	10	*stab*	Diff	1	-8	2	-5	2	2	1	2	1	2
20	10	*stab*	Rank	6	10	**1**	9	**1**	**1**	6	**1**	6	**1**
all	all	*stab*	Wins	3	0	3	1	3	3	2	5	2	3
all	all	*stab*	Losses	0	17	1	6	1	0	0	0	0	0
all	all	*stab*	Diff	3	-17	2	-5	2	3	2	5	2	3
all	all	*stab*	Rank	2	10	5	9	5	2	5	**1**	5	2
10	10	*NS*	Wins	4	2	2	0	9	0	2	8	6	5
10	10	*NS*	Losses	4	2	5	8	0	8	5	1	2	3
10	10	*NS*	Diff	0	0	-3	-8	9	-8	-3	7	4	2
10	10	*NS*	Rank	5	5	7	9	**1**	9	7	2	3	4
10	25	*NS*	Wins	9	0	7	5	8	2	3	2	0	6
10	25	*NS*	Losses	0	8	2	4	1	5	5	6	8	3
10	25	*NS*	Diff	9	-8	5	1	7	-3	-2	-4	-8	3
10	25	*NS*	Rank	**1**	9	3	5	2	7	6	8	9	4
10	50	*NS*	Wins	0	4	6	9	2	4	6	1	3	8
10	50	*NS*	Losses	9	4	2	0	7	4	2	8	6	1
10	50	*NS*	Diff	-9	0	4	9	-5	0	4	-7	-3	7
10	50	*NS*	Rank	10	5	3	**1**	8	5	3	9	7	2
1	10	*NS*	Wins	6	9	8	1	7	3	2	0	4	5
1	10	*NS*	Losses	3	0	1	8	2	6	7	9	5	4
1	10	*NS*	Diff	3	9	7	-7	5	-3	-5	-9	-1	1

Continued on next page

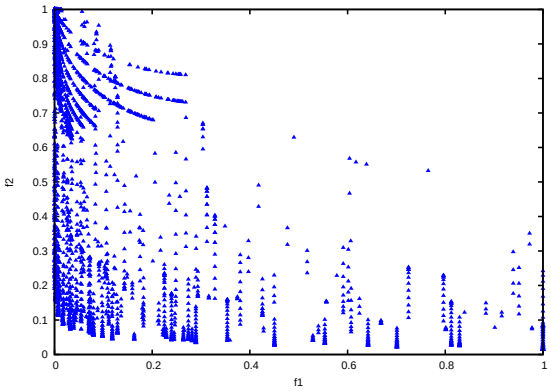
n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
1	10	*NS*	Rank	4	**1**	2	9	3	7	8	10	6	5
20	10	*NS*	Wins	6	9	0	6	1	4	4	2	8	3
20	10	*NS*	Losses	2	0	9	2	8	4	4	7	1	6
20	10	*NS*	Diff	4	9	-9	4	-7	0	0	-5	7	-3
20	10	*NS*	Rank	3	**1**	10	3	9	5	5	8	2	7
all	all	*NS*	Wins	25	24	23	21	27	13	17	13	21	27
all	all	*NS*	Losses	18	14	19	22	18	27	23	31	22	17
all	all	*NS*	Diff	7	10	4	-1	9	-14	-6	-18	-1	10
all	all	*NS*	Rank	4	**1**	5	6	3	9	8	10	6	**1**
10	10	all	Wins	10	2	4	1	15	1	3	14	8	10
10	10	all	Losses	4	11	8	14	0	12	9	1	6	3
10	10	all	Diff	6	-9	-4	-13	15	-11	-6	13	2	7
10	10	all	Rank	4	8	6	10	**1**	9	7	2	5	3
10	25	all	Wins	10	0	11	5	12	2	3	2	0	6
10	25	all	Losses	0	8	2	4	1	8	7	8	10	3
10	25	all	Diff	10	-8	9	1	11	-6	-4	-6	-10	3
10	25	all	Rank	2	9	3	5	**1**	7	6	7	10	4
10	50	all	Wins	2	10	10	15	2	4	7	1	5	9
10	50	all	Losses	11	4	2	1	14	7	5	12	6	3
10	50	all	Diff	-9	6	8	14	-12	-3	2	-11	-1	6
10	50	all	Rank	8	3	2	**1**	10	7	5	9	6	3
1	10	all	Wins	11	9	14	3	17	7	7	5	9	10
1	10	all	Losses	5	18	3	16	3	12	8	16	6	5
1	10	all	Diff	6	-9	11	-13	14	-5	-1	-11	3	5
1	10	all	Rank	3	8	2	10	**1**	7	6	9	5	4
20	10	all	Wins	9	9	2	6	3	6	8	4	15	9
20	10	all	Losses	2	10	13	8	12	5	4	10	1	6
20	10	all	Diff	7	-1	-11	-2	-9	1	4	-6	14	3
20	10	all	Rank	2	6	10	7	9	5	3	8	**1**	4
all	all	all	Wins	42	30	41	30	49	20	28	26	37	44
all	all	all	Losses	22	51	28	43	30	44	33	47	29	20
all	all	all	Diff	20	-21	13	-13	19	-24	-5	-21	8	24
all	all	all	Rank	2	8	4	7	3	10	6	8	5	**1**

Change Response Strategies applied to the Archive

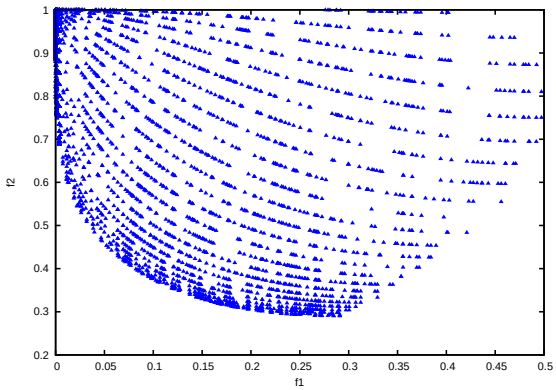
This section discusses results obtained by various responses to changes in the environment applied to the non-dominated solutions in the archive. The wins and losses of the various response approaches are presented in Tables 10.46 to 10.57, where the notation defined in Section 9.2 is used.



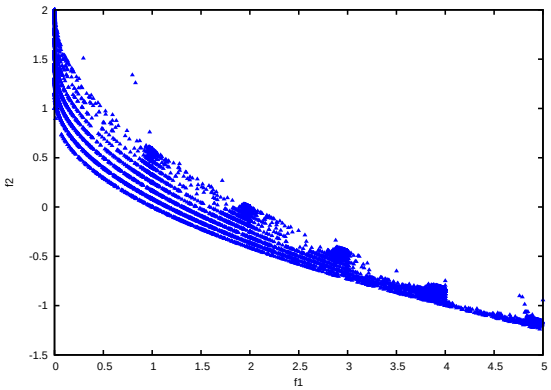
(a) POF^* for FDA1$_{Zhou}$



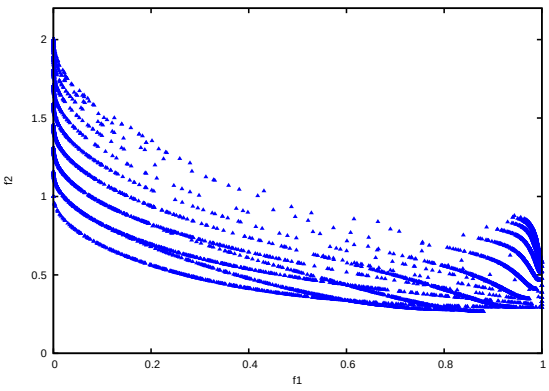
(b) POF^* for FDA2



(c) POF^* for FDA2$_{Camara}$

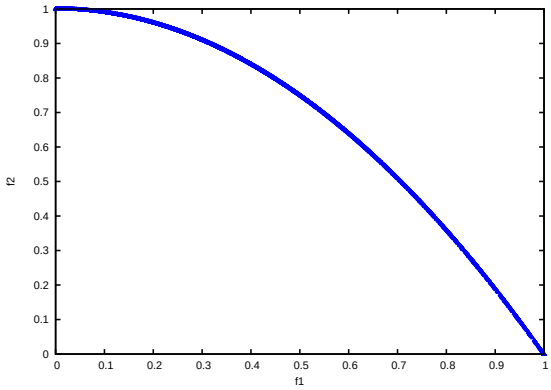


(d) POF^* for FDA3

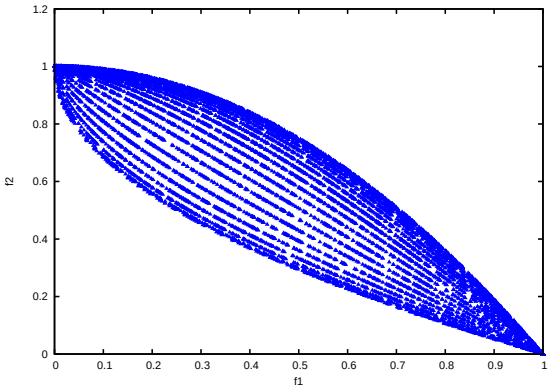


(e) POF^* for FDA3$_{Camara}$

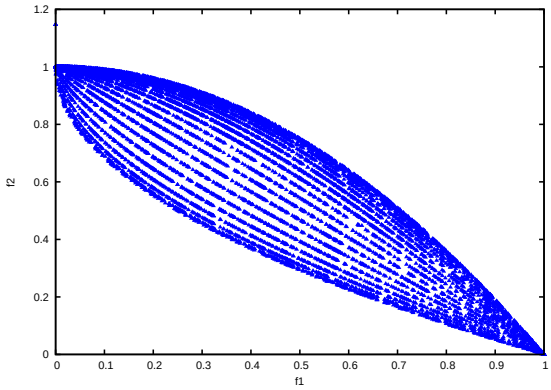
Figure 10.7: POF^* for FDA functions of DVEPSO using ri-c-30 for $n_t = 10$ and $\tau_t = 10$



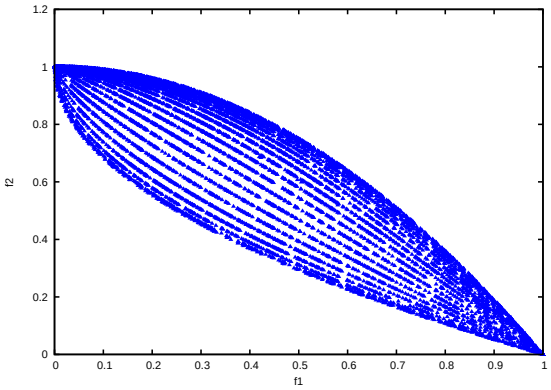
(a) POF^* for DIMP2



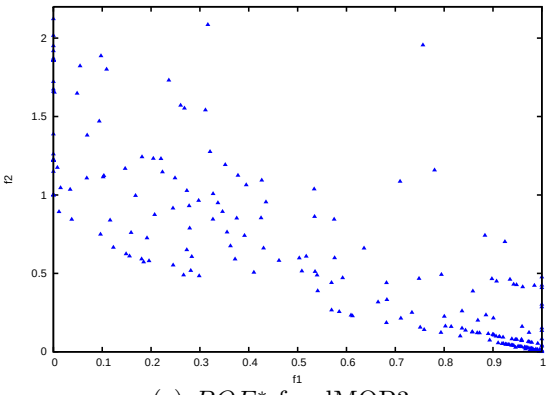
(b) POF^* for dMOP2



(c) POF^* for dMOP2$_{iso}$

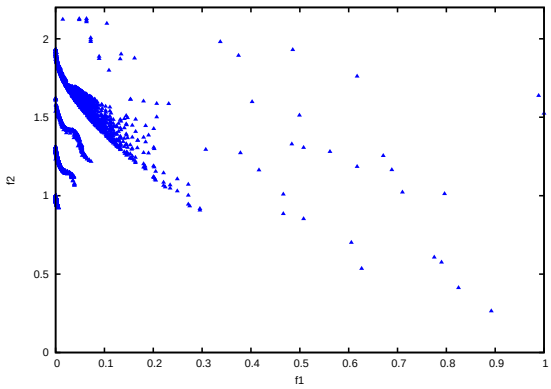


(d) POF^* for dMOP2$_{dec}$

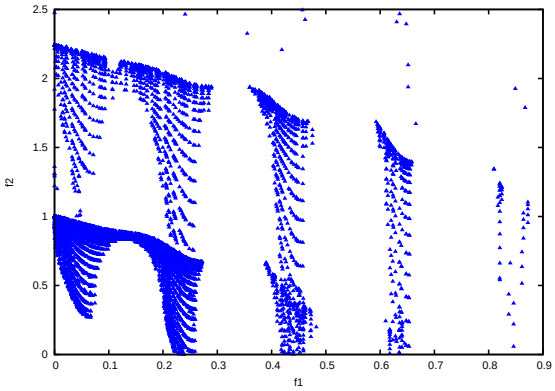


(e) POF^* for dMOP3

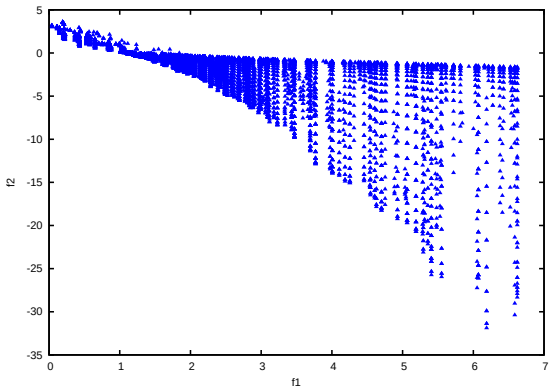
Figure 10.8: POF^* for DIMP2 and dMOP functions of DVEPSO using ri-c-30 for $n_t = 10$ and $\tau_t = 10$



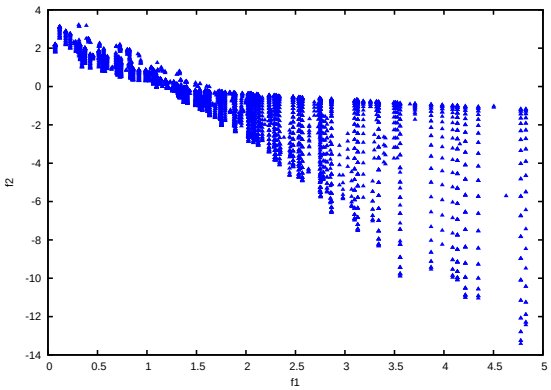
(a) POF^* for HE1



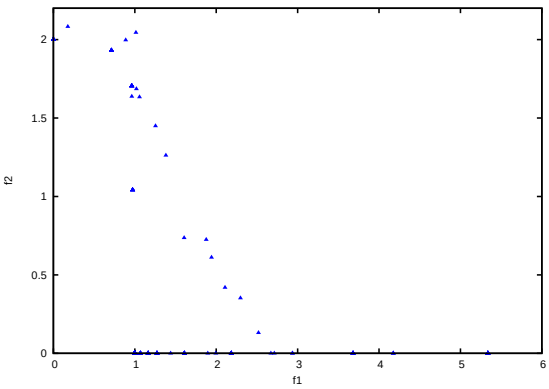
(b) POF^* for HE2



(c) POF^* for HE6



(d) POF^* for HE7



(e) POF^* for HE9

Figure 10.9: POF^* for HE functions of DVEPSO using *ri-c*-30 for $n_t = 10$ and $\tau_t = 10$

Results with regards to Performance Measures

This section discusses the results obtained for the performance measures by the responses applied to the archive. The wins and losses with regards to the various performance measures over all n_t-τ_t combinations are presented in Table 10.46.

The following are observed:

- The best performance for *acc* was obtained by *ac* and *ra*-0.5 performed the worst.
- For *stab*, *re* obtained the best performance and *ac* the worst.
- Measured against *NS*, the best rank was obtained by *re* and the worst rank by *ac*.

Table 10.46: Overall wins and losses for various performance measures obtained by various change response strategies applied to the archive

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	*acc*	Wins	74	69	48	58	60	65	77	71	158
all	all	*acc*	Losses	88	76	91	73	85	63	75	58	71
all	all	*acc*	Diff	-14	-7	-43	-15	-25	2	2	13	87
all	all	*acc*	Rank	6	5	9	7	8	3	3	2	**1**
all	all	*stab*	Wins	58	48	34	25	35	31	32	28	66
all	all	*stab*	Losses	28	30	30	26	28	31	37	34	113
all	all	*stab*	Diff	30	18	4	-1	7	0	-5	-6	-47
all	all	*stab*	Rank	**1**	2	4	6	3	5	7	8	9
all	all	*NS*	Wins	175	158	133	120	95	113	74	86	3
all	all	*NS*	Losses	29	43	49	57	91	72	108	105	403
all	all	*NS*	Diff	146	115	84	63	4	41	-34	-19	-400
all	all	*NS*	Rank	**1**	2	3	4	6	5	8	7	9

Results with regards to Various Frequencies and Severities of Change

Table **??** presents the wins and losses for the n_t-τ_t combinations.

For the different environment types the following observations are made:

- The worst performance for all n_t-τ_t combinations was obtained by *ac*.
- For $n_t = 10$ and $\tau_t = 50$, $n_t = 1$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$, the best rank was obtained by *re*.
- For $n_t = 10$ and $\tau_t = 10$, *rah*-1 performed the best.
- For $n_t = 10$ and $\tau_t = 25$, the best rank was obtained by *reh*.
- *cl* performed poorly, obtaining more losses than wins for all n_t-τ_t combinations.

Table 10.47: Overall wins and losses obtained by various change response strategies applied to the particles

n_t	τ_t	PM	Results	Particle response stategies									
				ri-c-30	**ri-a-30**	**ri-c-20**	**ri-a-20**	**ri-c-10**	**ri-a-10**	**re-c**	**re-a**	**reu-c**	**reu-a**
all	all	all	Wins	384	265	244	257	251	233	142	187	162	214
all	all	all	Losses	98	236	227	195	189	213	313	292	312	264
all	all	all	Diff	286	29	17	62	62	20	-171	-105	-150	-50
all	all	all	Rank	**1**	4	6	2	2	5	10	8	9	7

Results for Various Dynamic Multi-objective Optimisation Problem Types

This section discusses the results obtained by the various response approaches for DMOOPs of Type I, II and III respectively.

Type I DMOOPs

The wins and losses obtained for Type I DMOOPs are presented in Table 10.48.

Table 10.48: Overall wins and losses for various performance measures obtained by various change response strategies applied to the archive solving Type I DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	*acc*	Wins	2	2	3	3	6	1	1	1	1
all	all	*acc*	Losses	3	1	0	0	0	1	9	1	5
all	all	*acc*	Diff	-1	1	3	3	6	0	-8	0	-4
all	all	*acc*	Rank	7	4	2	2	**1**	5	9	5	8
all	all	*stab*	Wins	1	0	2	1	5	0	1	0	0
all	all	*stab*	Losses	1	1	0	0	0	1	1	1	5
all	all	*stab*	Diff	0	-1	2	1	5	-1	0	-1	-5
all	all	*stab*	Rank	4	6	2	3	**1**	6	4	6	9
all	all	*NS*	Wins	1	3	1	0	1	0	6	4	0
all	all	*NS*	Losses	2	0	2	1	2	4	1	2	2
all	all	*NS*	Diff	-1	3	-1	-1	-1	-4	5	2	-2
all	all	*NS*	Rank	4	2	4	4	4	9	**1**	3	8

With regards to the performance measures, the following are observed:

- The best performance for *acc* was obtained by *ra*-1 and the worst by *ra*-1.5.
- For *stab*, *ra*-1 also performed the best and *ac* performed the worst.
- Measured against *NS*, *ra*-1.5 obtained the best rank and *rah*-1 obtained the worst rank.

Table 10.49 presents the wins and losses with regards to the n_t-τ_t combinations.

Table 10.49: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the archive solving Type I DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
10	10	all	Wins	0	0	0	0	0	0	0	0	0
10	10	all	Losses	0	0	0	0	0	0	0	0	0
10	10	all	Diff	0	0	0	0	0	0	0	0	0
10	10	all	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	25	all	Wins	0	0	2	1	0	0	1	0	0
10	25	all	Losses	4	0	0	0	0	0	0	0	0
10	25	all	Diff	-4	0	2	1	0	0	1	0	0
10	25	all	Rank	9	4	**1**	2	4	4	2	4	4
10	50	all	Wins	0	0	0	0	0	0	0	4	0
10	50	all	Losses	1	0	1	0	0	0	1	0	1
10	50	all	Diff	-1	0	-1	0	0	0	-1	4	-1
10	50	all	Rank	6	2	6	2	2	2	6	**1**	6
1	10	all	Wins	4	5	4	3	2	1	7	1	1
1	10	all	Losses	1	0	1	1	2	4	8	2	9
1	10	all	Diff	3	5	3	2	0	-3	-1	-1	-8
1	10	all	Rank	2	**1**	2	4	5	8	6	6	9
20	10	all	Wins	0	0	0	0	10	0	0	0	0
20	10	all	Losses	0	2	0	0	0	2	2	2	2
20	10	all	Diff	0	-2	0	0	10	-2	-2	-2	-2
20	10	all	Rank	2	5	2	2	**1**	5	5	5	5

The following observations are made:

- For $n_t = 10$ and $\tau_t = 10$, there was no statistical significant diffference between the performance measure values of the various response approaches.
- The best performance for $n_t = 10$ and $\tau_t = 25$ was obtained by ra-0.5, with re performing the worst.
- In a slow changing environment ($\tau_t = 50$), rah-1.5 performed the best.
- The best rank for a severely changing environment was obtained by reh, with ac obtaining the worst rank.
- In a gradually changing environment ra-1 performed the best. Five approaches performed equally poor obtaining more losses than wins, namely reh, rah-1, ra-1.5, rah-1.5 and ac.

The wins and losses for Type I DMOO benchmark functions are presented in Table 10.50. For Type I DMOOPs the best rank was obtained by ra-1, with ac obtaining the worst rank.

Table 10.50: Overall wins and losses obtained by various change response strategies applied to the archive solving Type I DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	all	Wins	4	5	6	4	12	1	8	5	1
all	all	all	Losses	6	2	2	1	2	6	11	4	12
all	all	all	Diff	-2	3	4	3	10	-5	-3	1	-11
all	all	all	Rank	6	3	2	3	**1**	8	7	5	9

Type II DMOOPs

The wins and losses obtained by the various response approaches for Type II DMOOPs are presented in Table 10.51.

Table 10.51: Overall wins and losses for various performance measures obtained by various change response strategies applied to the archive solving Type II DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	*acc*	Wins	34	20	19	15	18	22	39	27	89
all	all	*acc*	Losses	33	32	44	35	42	33	25	28	11
all	all	*acc*	Diff	1	-12	-25	-20	-24	-11	14	-1	78
all	all	*acc*	Rank	3	6	9	7	8	5	2	4	**1**
all	all	*stab*	Wins	31	28	16	15	16	23	17	19	54
all	all	*stab*	Losses	25	17	25	18	23	18	31	22	40
all	all	*stab*	Diff	6	11	-9	-3	-7	5	-14	-3	14
all	all	*stab*	Rank	3	2	8	5	7	4	9	5	**1**
all	all	*NS*	Wins	53	53	44	45	39	33	30	30	0
all	all	*NS*	Losses	1	1	9	7	13	17	24	23	232
all	all	*NS*	Diff	52	52	35	38	26	16	6	7	-232
all	all	*NS*	Rank	**1**	**1**	4	3	5	6	8	7	9

The following observations are made:

- The best performance for all performance measures was obtained by *ac*.
- The worst rank for *acc* and *stab* were obtained by *ra*-0.5 and *ra*-1.5 respectively.
- For *NS* the worst performing approach was *ac*.
- Four approaches obtained more losses than wins for two performance measures, namely *ra*-0.5, *rah*-0.5, *ra*-1 and *rah*-1.5.

Table 10.49 presents the wins and losses for the n_t-τ_t combinations. With regards to the various environment types, the following are observed:

- The worst performing approach for all n_t-τ_t combinations was *ac*.

- For $n_t = 10$ and $\tau_t = 10$, *rah*-0.5 performed the best.
- For $n_t = 10$ and $\tau_t = 25$, both *re* and *reh* obtained the best rank.
- In slow changing environments ($\tau_t = 50$), *ra*-1.5 was the best performing approach.
- For severely and gradually changing environments, *re* performed the best.
- More losses than wins were awarded to *ac* for all n_t-τ_t combinations.

Table 10.52: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the archive solving Type II DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
10	10	all	Wins	28	26	19	26	26	22	17	17	49
10	10	all	Losses	32	13	24	9	22	13	33	18	66
10	10	all	Diff	-4	13	-5	17	4	9	-16	-1	-17
10	10	all	Rank	6	2	7	**1**	4	3	8	5	9
10	25	all	Wins	25	27	19	12	15	17	20	7	9
10	25	all	Losses	4	6	7	10	10	12	12	24	66
10	25	all	Diff	21	21	12	2	5	5	8	-17	-57
10	25	all	Rank	**1**	**1**	3	7	5	5	4	8	9
10	50	all	Wins	16	13	12	13	12	14	23	13	13
10	50	all	Losses	11	11	15	7	10	11	6	10	48
10	50	all	Diff	5	2	-3	6	2	3	17	3	-35
10	50	all	Rank	3	6	8	2	6	4	**1**	4	9
1	10	all	Wins	21	17	8	8	10	14	12	16	46
1	10	all	Losses	4	7	23	22	12	17	12	7	48
1	10	all	Diff	17	10	-15	-14	-2	-3	0	9	-2
1	10	all	Rank	**1**	2	9	8	5	7	4	3	5
20	10	all	Wins	28	18	21	16	10	11	14	23	26
20	10	all	Losses	8	13	9	12	24	15	17	14	55
20	10	all	Diff	20	5	12	4	-14	-4	-3	9	-29
20	10	all	Rank	**1**	4	2	5	8	7	6	3	9

Table 10.53 presents the wins and losses of the response approaches' over all performance measures and all n_t-τ_t combinations solving Type II DMOOPs.

For Type II DMOOPs, *re* performed the best and *ac* the worst. All other approaches were completely outperformed by *ac*, since *ac* obtained 140 more losses than wins. In constrast, all other approaches were awarded more losses than wins. However, *ac* performed the best with regards to *acc* for Type II environments.

Table 10.53: Overall wins and losses obtained by various change response strategies applied to the archive solving Type II DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	all	Wins	118	101	79	75	73	78	86	76	143
all	all	all	Losses	59	50	78	60	78	68	80	73	283
all	all	all	Diff	59	51	1	15	-5	10	6	3	-140
all	all	all	Rank	**1**	2	7	3	8	4	5	6	9

Type III DMOOPs

The wins and losses obtained for Type III DMOOPs are presented in Table 10.54.

Table 10.54: Overall wins and losses for various performance measures obtained by various change response strategies applied to the archive solving Type III DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	*acc*	Wins	38	47	26	40	36	42	37	43	68
all	all	*acc*	Losses	52	43	47	38	43	29	41	29	55
all	all	*acc*	Diff	-14	4	-21	2	-7	13	-4	14	13
all	all	*acc*	Rank	8	4	9	5	7	2	6	**1**	2
all	all	*stab*	Wins	26	20	16	9	14	8	14	9	12
all	all	*stab*	Losses	2	12	5	8	5	12	5	11	68
all	all	*stab*	Diff	24	8	11	1	9	-4	9	-2	-56
all	all	*stab*	Rank	**1**	5	2	6	3	8	3	7	9
all	all	*NS*	Wins	121	102	88	75	55	80	38	52	3
all	all	*NS*	Losses	26	42	38	49	76	51	83	80	169
all	all	*NS*	Diff	95	60	50	26	-21	29	-45	-28	-166
all	all	*NS*	Rank	**1**	2	3	5	6	4	8	7	9

The following observations are made:

- For *acc* the best performance was obtained by *rah*-1.5 and the worst performance by *ra*-0.5.
- The best rank for *stab* was awarded to *re*, while *ac* was awarded the worst rank.
- Measured against *NS*, *re* performed the best and *ac* the worst.

Table 10.49 presents the wins and losses with regards to the various n_t-τ_t combinations. With regards to the different environments, the following are observed:

- The approach that performed the worst for all n_t-τ_t combinations was *ac*.
- For $n_t = 10$ and $\tau_t = 10$, *ra*-0.5 performed the best.

- The best performing approach for $n_t = 10$ and $\tau_t = 25$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 20$ and $\tau_t = 10$, was *re*.
- For severely changing environments *rah*-0.5 performed the best.
- *ac* performed poorly, obtaining more losses than wins for all n_t-τ_t combinations.

Table 10.55: Overall wins and losses for various frequencies and severities of change obtained by various change response strategies applied to the archive solving Type III DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
10	10	all	Wins	39	32	36	25	15	34	22	15	17
10	10	all	Losses	18	18	7	19	31	13	18	31	80
10	10	all	Diff	21	14	29	6	-16	21	4	-16	-63
10	10	all	Rank	2	4	**1**	5	7	2	6	7	9
10	25	all	Wins	27	28	26	7	9	16	8	10	7
10	25	all	Losses	7	9	7	20	15	9	24	15	32
10	25	all	Diff	20	19	19	-13	-6	7	-16	-5	-25
10	25	all	Rank	**1**	2	2	7	6	4	8	5	9
10	50	all	Wins	30	20	5	15	13	7	10	16	12
10	50	all	Losses	11	10	11	10	16	21	8	20	21
10	50	all	Diff	19	10	-6	5	-3	-14	2	-4	-9
10	50	all	Rank	**1**	2	7	3	5	9	4	6	8
1	10	all	Wins	44	38	37	48	38	41	26	44	24
1	10	all	Losses	29	37	36	26	37	28	49	25	73
1	10	all	Diff	15	1	1	22	1	13	-23	19	-49
1	10	all	Rank	3	5	5	**1**	5	4	8	2	9
20	10	all	Wins	45	51	26	29	30	32	23	19	23
20	10	all	Losses	15	23	29	20	25	21	30	29	86
20	10	all	Diff	30	28	-3	9	5	11	-7	-10	-63
20	10	all	Rank	**1**	2	6	4	5	3	7	8	9

The overall wins and losses for Type III benchmark functions are presented in Table 10.56.

Table 10.56: Overall wins and losses obtained by various change response strategies applied to the archive solving Type III DMOOPs

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	all	Wins	185	169	130	124	105	130	89	104	83
all	all	all	Losses	80	97	90	95	124	92	129	120	292
all	all	all	Diff	105	72	40	29	-19	38	-40	-16	-209
all	all	all	Rank	**1**	2	3	5	7	4	8	6	9

The best performing approach for Type III DMOOPs was *re* and the worst performing approach was *ac*. For Type III DMOOPs the response *re* completely outperformed the other responses, obtaining 105 more wins than losses. The approach that was awarded the second highest rank obtained 72 more wins than losses.

Overall Performance

This section discusses the overall performance of the responses applied to the archive. The wins and losses over all DMOOPs, n_t-τ_t combinations and performance measures are presented in Table 10.57. The following observations are made:

- The best overall rank was obtained by *re*, obtaining 162 more wins than losses. Therefore, *re* completely outperformed the other archive response approaches.
- The worst performance was obtained by *ac*, that was awarded 360 more losses than wins.

Table 10.57: Overall wins and losses obtained by various change response strategies applied to the archive

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Archive response stategies**								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	all	Wins	307	275	215	203	190	209	183	185	227
all	all	all	Losses	145	149	170	156	204	166	220	197	587
all	all	all	Diff	162	126	45	47	-14	43	-37	-12	-360
all	all	all	Rank	**1**	2	4	3	7	5	8	6	9

However, *ac* obtained a lot of losses for NS. Table 10.58 presents the wins and losses without taking NS into account. The effect of NS can clearly be seen. *ac* obtains the best overall rank when taking only *acc* and *stab* into account.

Table 10.58: Overall wins and losses over *acc* and *stab* obtained by various change response strategies applied to the archive

$\mathbf{n_t}$	τ_t	**PM**	**Results**	**Archive response stategies**								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
all	all	all	Wins	132	117	82	83	95	96	109	99	224
all	all	all	Losses	116	106	121	99	113	94	112	92	184
all	all	all	Diff	16	11	-39	-16	-18	2	-3	7	40
all	all	all	Rank	2	3	9	7	8	5	6	4	**1**

The POF^*s found by *ac* for $n_t = 10$ and $\tau_t = 10$ are illustrated in Figures 10.10 to 10.12. A similar trend was observed than the POF^*s found by *ra-t*. However, for FDA3 *ac* obtained a better spread of solutions and for FDA2 *ac* found more solutions that were close to the true POF. It should be noted that for dMOP3, *ac* found a good spread of solutions on the true POF, with some outlier solutions further away from the true POF. Therefore, *ac* found the best POF^* for dMOP3 from all the winning approaches illustrated in Figures 10.2, 10.5, 10.8 and 10.12. Furthermore, *ac* obtained a good spread of solutions for HEF6 and HEF7.

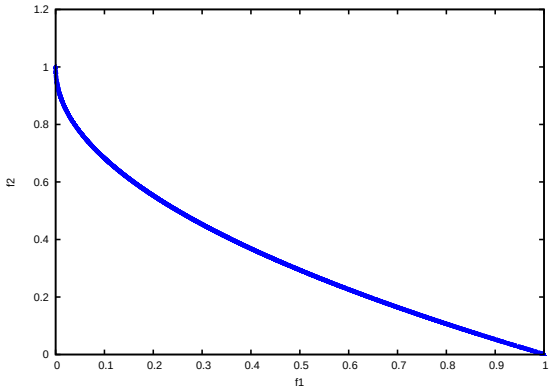
General Observations

This section discusses general observations that were made with regards to the performance of the responses applied to the archive.

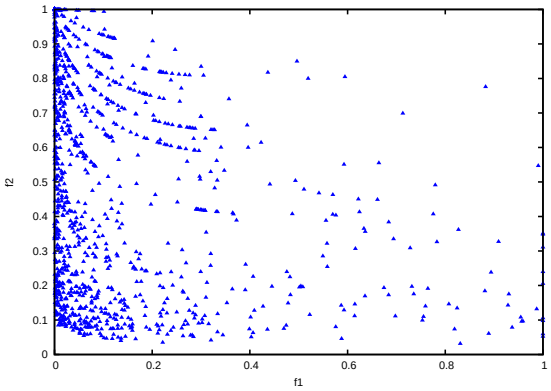
The best performing approach for Type I DMOOPs was *ra*-1. Surprisingly, *ac* performed the best for Type II and second best for Type III DMOOPs with regards to *acc*, but obtained the second lowest rank for Type I DMOOPs. Therefore, *ac* performs well when the POF changes over time. However, when the POF remains static, the other responses perform better, since they re-use previously obtained knowledge.

Similar to the response strategies applied to the particles, with DIMP2 and the FDA functions there was no statistical significant difference between the performance measure values of the various response strategies applied to the archive for many n_t-τ_t combinations.

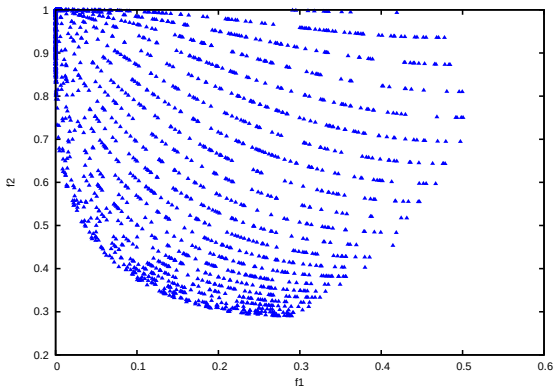
Re-evaluating the solutions in the archive re-uses previously obtained solutions. On the other hand, removing all solutions from the archive also remove optimal solutions found in previous time steps that were either still non-dominant after changes in the environment, or that would have become non-dominant again after applying hill climbing. The solutions found in the time steps that the environment was static (solutions found by *ac*), did not produce as good accuracy values as the combination of previously found solutions and newly found solutions (solutions by the re-evaluation approaches). Therefore, the re-evaluation approaches performed better than *ac* with HE9. It should be noted that HE9 is a difficult DMOOP to solve, since each decision variable has its own POS, the POSs have non-linear functions, and a transformation function is used for



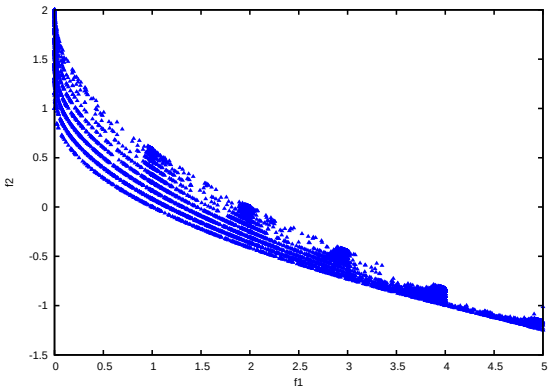
(a) POF^* for FDA1$_{Zhou}$



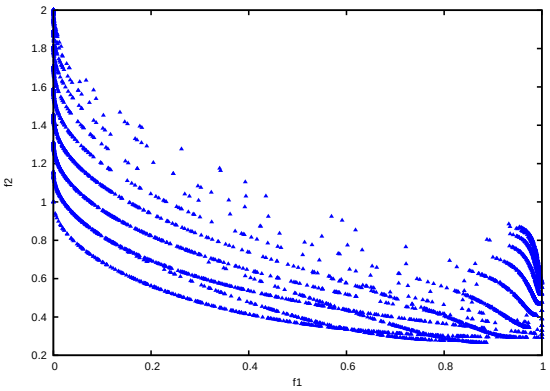
(b) POF^* for FDA2



(c) POF^* for FDA2$_{Camara}$

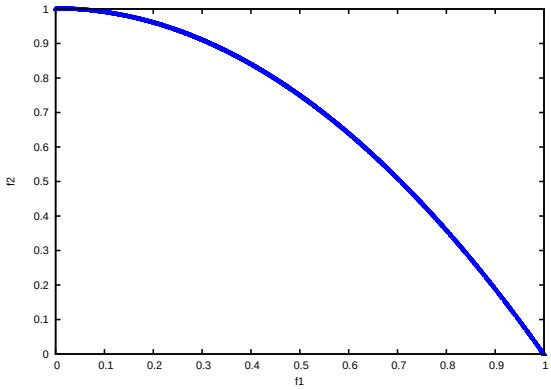


(d) POF^* for FDA3

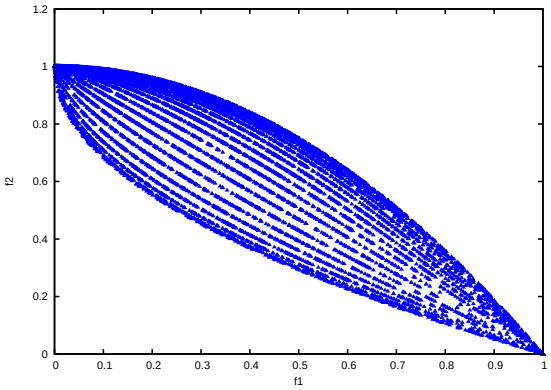


(e) POF^* for FDA3$_{Camara}$

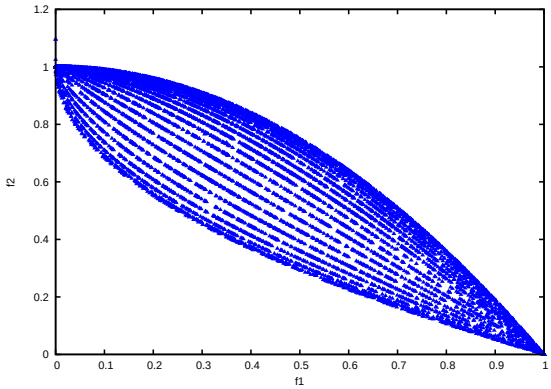
Figure 10.10: POF^* for FDA functions of DVEPSO using ac for $n_t = 10$ and $\tau_t = 10$



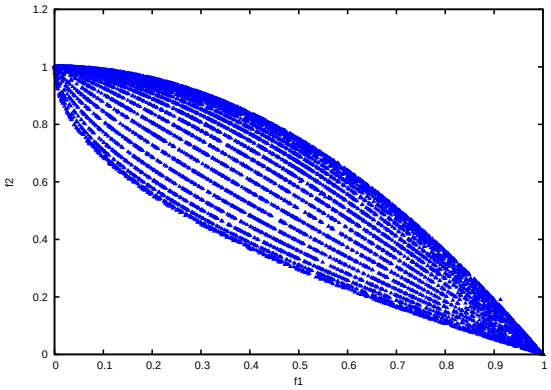
(a) POF^* for DIMP2



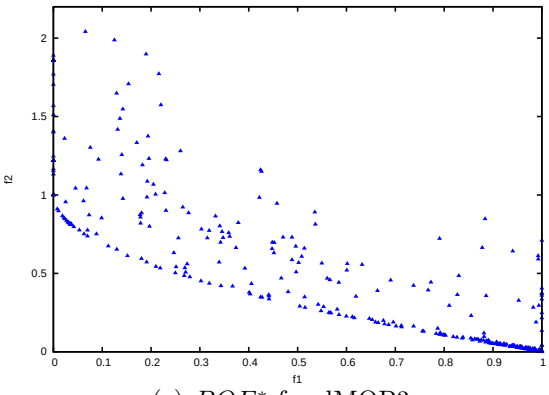
(b) POF^* for dMOP2



(c) POF^* for dMOP2$_{iso}$

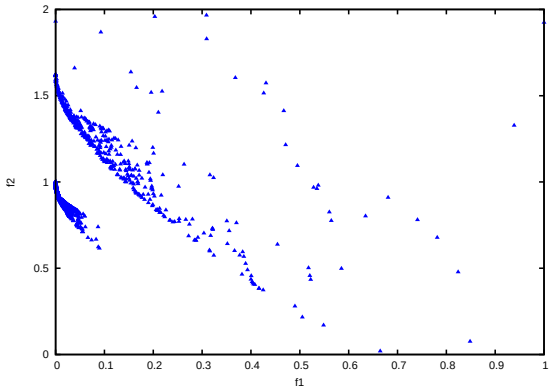


(d) POF^* for dMOP2$_{dec}$

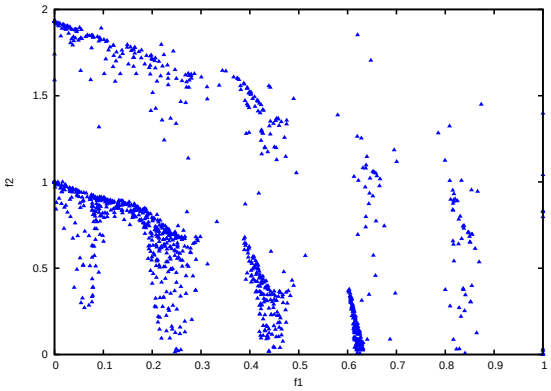


(e) POF^* for dMOP3

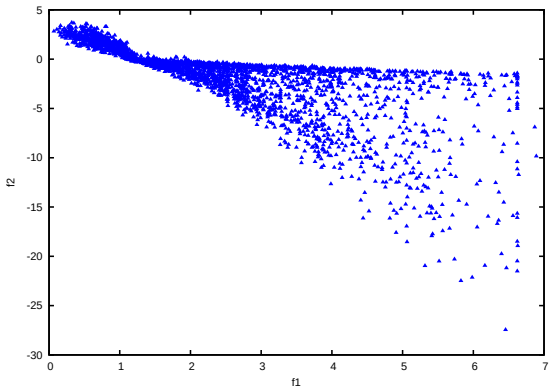
Figure 10.11: POF^* for DIMP2 and dMOP2 functions of DVEPSO using ac for $n_t = 10$ and $\tau_t = 10$



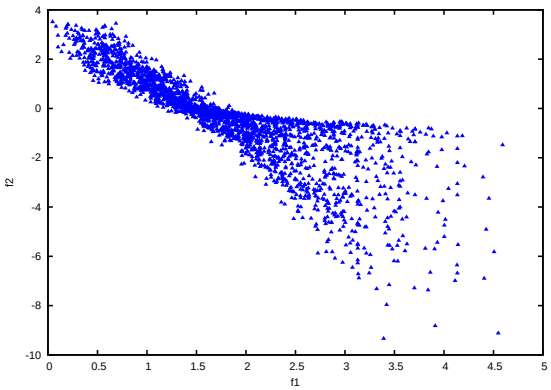
(a) POF^* for HE1



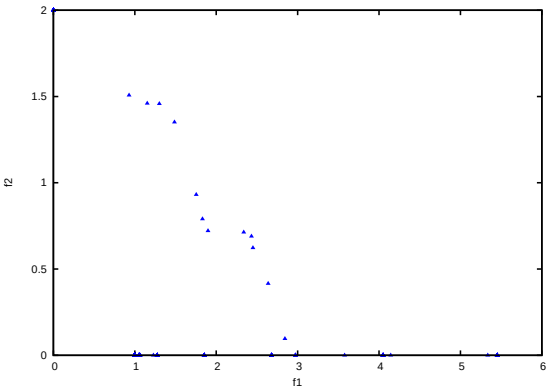
(b) POF^* for HE2



(c) POF^* for HE6



(d) POF^* for HE7



(e) POF^* for HE9

Figure 10.12: POF^* for HE functions of DVEPSO using ac for $n_t = 10$ and $\tau_t = 10$

the decision variables.

The wins and losses for HE9 are presented in Table 10.59. For HE9, *ac* obtained the lowest rank for *acc*, being awarded no wins and 10 losses. With regards to stability, there was not a huge difference in performance between the various approaches. Measured against NS, *ac* performed reasonably well. With regards to the overall wins and losses for HE9, *ac* obtained the second lowest rank. For the re-evaluation approaches, *reh* performed the best for *acc* and *re* the third best. Measured against *stab*, *re* performed the best and *reh* the worst. However, the best performing approaches obtained only one win, and *reh* was awarded no wins and only two losses. With regards to NS, both *re* and *reh* obtained rank six. With regards to the overall wins and losses for HE9, the re-evaluation approaches performed really well, obtaining the best and fourth best ranks.

Table 10.59: Wins and Losses of HE9 for various change response strategies applied to the archive

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
10	10	*acc*	Wins	0	2	0	0	0	0	0	0	0
10	10	*acc*	Losses	1	0	0	0	1	0	0	0	0
10	10	*acc*	Diff	-1	2	0	0	-1	0	0	0	0
10	10	*acc*	Rank	8	**1**	2	2	8	2	2	2	2
10	25	*acc*	Wins	0	0	4	0	0	0	0	0	0
10	25	*acc*	Losses	1	0	0	1	0	1	0	1	0
10	25	*acc*	Diff	-1	0	4	-1	0	-1	0	-1	0
10	25	*acc*	Rank	6	2	**1**	6	2	6	2	6	2
10	50	*acc*	Wins	6	0	0	0	0	0	2	0	0
10	50	*acc*	Losses	0	1	1	0	2	1	0	1	2
10	50	*acc*	Diff	6	-1	-1	0	-2	-1	2	-1	-2
10	50	*acc*	Rank	**1**	4	4	3	8	4	2	4	8
1	10	*acc*	Wins	3	7	4	2	2	0	2	1	0
1	10	*acc*	Losses	1	0	0	3	2	5	2	1	7
1	10	*acc*	Diff	2	7	4	-1	0	-5	0	0	-7
1	10	*acc*	Rank	3	**1**	2	7	4	8	4	4	9
20	10	*acc*	Wins	0	7	0	0	0	1	0	0	0
20	10	*acc*	Losses	1	0	1	1	1	0	2	1	1
20	10	*acc*	Diff	-1	7	-1	-1	-1	1	-2	-1	-1
20	10	*acc*	Rank	3	**1**	3	3	3	2	9	3	3
all	all	*acc*	Wins	9	16	8	2	2	1	4	1	0
all	all	*acc*	Losses	4	1	2	5	6	7	4	4	10
all	all	*acc*	Diff	5	15	6	-3	-4	-6	0	-3	-10
all	all	*acc*	Rank	3	**1**	2	5	7	8	4	5	9
10	25	*stab*	Wins	1	0	0	0	0	0	0	0	0
10	25	*stab*	Losses	0	0	1	0	0	0	0	0	0
10	25	*stab*	Diff	1	0	-1	0	0	0	0	0	0

Continued on next page

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
10	25	*stab*	Rank	**1**	2	9	2	2	2	2	2	2
20	10	*stab*	Wins	0	0	1	1	0	0	0	0	0
20	10	*stab*	Losses	0	2	0	0	0	0	0	0	0
20	10	*stab*	Diff	0	-2	1	1	0	0	0	0	0
20	10	*stab*	Rank	3	9	**1**	**1**	3	3	3	3	3
all	all	*stab*	Wins	1	0	1	1	0	0	0	0	0
all	all	*stab*	Losses	0	2	1	0	0	0	0	0	0
all	all	*stab*	Diff	1	-2	0	1	0	0	0	0	0
all	all	*stab*	Rank	**1**	9	3	**1**	3	3	3	3	3
10	10	*NS*	Wins	0	0	0	0	0	0	0	0	0
10	10	*NS*	Losses	0	0	0	0	0	0	0	0	0
10	10	*NS*	Diff	0	0	0	0	0	0	0	0	0
10	10	*NS*	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	25	*NS*	Wins	0	0	0	0	0	1	0	0	0
10	25	*NS*	Losses	0	0	0	0	0	0	1	0	0
10	25	*NS*	Diff	0	0	0	0	0	1	-1	0	0
10	25	*NS*	Rank	2	2	2	2	2	**1**	9	2	2
10	50	*NS*	Wins	0	0	0	0	0	0	0	0	0
10	50	*NS*	Losses	0	0	0	0	0	0	0	0	0
10	50	*NS*	Diff	0	0	0	0	0	0	0	0	0
10	50	*NS*	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	10	*NS*	Wins	2	2	4	5	0	4	1	4	2
1	10	*NS*	Losses	4	4	0	0	8	0	5	0	3
1	10	*NS*	Diff	-2	-2	4	5	-8	4	-4	4	-1
1	10	*NS*	Rank	6	6	2	**1**	9	2	8	2	5
20	10	*NS*	Wins	0	0	0	1	0	4	0	0	0
20	10	*NS*	Losses	1	1	0	0	1	1	0	0	1
20	10	*NS*	Diff	-1	-1	0	1	-1	3	0	0	-1
20	10	*NS*	Rank	6	6	3	2	6	**1**	3	3	6
all	all	*NS*	Wins	2	2	4	6	0	9	1	4	2
all	all	*NS*	Losses	5	5	0	0	9	1	6	0	4
all	all	*NS*	Diff	-3	-3	4	6	-9	8	-5	4	-2
all	all	*NS*	Rank	6	6	3	2	9	**1**	8	3	5
10	10	all	Wins	0	2	0	0	0	0	0	0	0
10	10	all	Losses	1	0	0	0	1	0	0	0	0
10	10	all	Diff	-1	2	0	0	-1	0	0	0	0
10	10	all	Rank	8	**1**	2	2	8	2	2	2	2
10	25	all	Wins	1	0	4	0	0	1	0	0	0
10	25	all	Losses	1	0	1	1	0	1	1	1	0
10	25	all	Diff	0	0	3	-1	0	0	-1	-1	0
10	25	all	Rank	2	2	**1**	7	2	2	7	7	2
10	50	all	Wins	6	0	0	0	0	0	2	0	0
10	50	all	Losses	0	1	1	0	2	1	0	1	2
10	50	all	Diff	6	-1	-1	0	-2	-1	2	-1	-2
10	50	all	Rank	**1**	4	4	3	8	4	2	4	8
1	10	all	Wins	5	9	8	7	2	4	3	5	2
1	10	all	Losses	5	4	0	3	10	5	7	1	10
1	10	all	Diff	0	5	8	4	-8	-1	-4	4	-8

Continued on next page

n_t	τ_t	PM	Results	Archive response stategies								
				re	**reh**	**ra-0.5**	**rah-0.5**	**ra-1**	**rah-1**	**ra-1.5**	**rah-1.5**	**ac**
1	10	all	Rank	5	2	**1**	3	8	6	7	3	8
20	10	all	Wins	0	7	1	2	0	5	0	0	0
20	10	all	Losses	2	3	1	1	2	1	2	1	2
20	10	all	Diff	-2	4	0	1	-2	4	-2	-1	-2
20	10	all	Rank	6	**1**	4	3	6	**1**	6	5	6
all	all	all	Wins	12	18	13	9	2	10	5	5	2
all	all	all	Losses	9	8	3	5	15	8	10	4	14
all	all	all	Diff	3	10	10	4	-13	2	-5	1	-12
all	all	all	Rank	4	**1**	**1**	3	9	5	7	6	8

10.3 Summary

This chapter investigated the effect of various parameters on the performance of DVEPSO. These parameters were approaches to manage boundary constraint violations, approaches to share knowledge between the various sub-swarms and responses to a change in the environment applied to either the particles of the sub-swarms or the non-dominated solutions in the archive.

The boundary constraint violation management approach that performed the best was the clamping (*cl*) approach. The clamping approach places any particle that violates a specific boundary of the search space on or close to the violated boundary.

The best and second best performing approaches to share knowledge between the sub-swarms of DVEPSO was the ring-tournament (*ri-t*) and random-tournament approach (*ra-t*) respectively. However, the random-tournament approach (*ra-t*) performed the best with regards to *acc*. With the random-tournament approach, the sub-swarm from which the global guide is selected for a specific swarm's particles' velocity update is randomly selected. Tournament selection is performed to select the global guide from the selected sub-swarm.

When a change in the environment occurs, a response should be applied to both the particles of the sub-swarms and the archive. The best performing response applied to the particles was *ri-c*-30. When a change was detected, 30% of the particles of the swarm(s) whose objective function changed were re-initialised. Re-initialised particles' positions were re-set to new random positions in the search space. After re-initialisation, all particles' pbest was re-set to their current positions and a new gbest was chosen

according to the new environment.

The response applied to the archive that performed the best with regards to *acc* and *stab* was *ac*. When a change in the environment was detected, all solutions were removed from the archive.

For each of these four parameters, the best performing approach are used in the experiments of the next chapter. The next chapter compares the performance of the best DVEPSO configuration against four state-of-the-art DMOO algorithms.

Chapter 11

Comparing the Dynamic Vector Evaluated Particle Swarm Optimisation Algorithm against State-of-the-art Dynamic Multi-objective Optimisation Algorithms

> *"Learn to adjust yourself to the conditions you have to endure, but make a point of trying to alter or correct conditions so that they are most favorable to you."* – William Frederick Book

In order to determine how efficiently a DMOO algorithm solves DMOOPs, its performance should be compared against the performance of other DMOO algorithms. This chapter discusses experiments conducted to compare the performance of DVEPSO against four state-of-the art DMOO algorithms. The experimental setup is discussed in Section 11.1. Section 11.2 discusses the results that were obtained by the various DMOO algorithms. Finally, the chapter is summarised in Section 11.3.

11.1 Experimental Setup

This section describes the experimental setup of the experiments discussed in this chapter. The process followed to optimise the parameters of the DMOO algorithms are discussed in Section 11.1.1. Section 11.1.2 discusses the experiments that were conducted to compare the performance of the DMOO algorithms. Furthermore, the statistical analysis that were performed on the obtained results are described.

11.1.1 Parameter Optimisation of Dynamic Multi-objective Optimisation Algorithms

The four DMOO algorithms that DVEPSO were compared against in the experiments are:

- The DNSGA-II-A algorithm, an NSGA-II algorithm adapted for DMOO and proposed by Deb *et al.* [46]. If a change in the environment is detected, a percentage of individuals are randomly selected and replaced with newly created individuals.
- The DNSGA-II-B algorithm, an NSGA-II algorithm that selects a percentage of individuals randomly and replaces them with individuals that are mutated from existing individuals when a change is detected. DNSGA-II-B was proposed by Deb *et al.* [46].
- The dCOEA algorithm, a dynamic competitive-cooperative coevolutionary algorithm proposed by Goh and Tan [67].
- The DMOPSO algorithm, a MOPSO algorithm adapted for DMOO by Lechuga [102].

The DNSGA-II algorithms were selected since NSGA-II performed so well with MOO that it became a benchmark in the field. DMOPSO is one of only two PSO algorithms proposed for DMOO and was one of the first PSO algorithms proposed to solve MOOPs. The dCOEA algorithm was selected as a multi-population approach where the sub-populations co-operate with one another, as is the case with DVEPSO. These four algorithms are discussed in more detail in Chapter 8.

The source code of the dCOEA algorithm was obtained from the authors of [67]. The source code of the static NSGA-II algorithm was obtained from [109] and was adapted for DMOO according to [46]. The source code of MOPSO was obtained from the authors

of [102], adapted for DMOO according to [102] and extended using sentry particles to detect a change in the environment.

For each of the three EAs listed above, the following approach was followed to optimise its parameters: the parameters and values for the parameters were identified from the literature. For each parameter value, the algorithm executed 30 independent runs and each run continued for 1000 iterations. Fifteen benchmark functions were used as discussed in Section 9.4.1. For all benchmark functions, the severity of change (n_t) was set to 1, 10 and 20 and the frequency of change (τ_t) was set to 10, 25 and 50. Three performance measures were used as discussed in Section 9.4.1. Wins and losses were calculated for each parameter value as discussed in Section 9.4.1. Based on the wins and losses, the best value for the parameter was selected. For dCOEA there was no statistical significant difference in the performance of the algorithm for different values of the SC_{ratio} and R_{size} parameters. Therefore, for these two parameters the default values of [67] were selected. For the PSO-based algorithms, the same c_1, c_2 and w values were used. The best configuration for DVEPSO was selected based on the results of the experiments discussed in Chapters 9 and 10.

According to Malan and Engelbrecht, using the same number of particles or individuals when comparing algorithms are not adequate to ensure a fair comparison [115]. Therefore, for these experiments the number of particles or individuals assigned to each algorithm depended upon the amount of new information that each individual contributed after an algorithm iteration. For the DNSGA-II algorithms, DVEPSO, and dynamic MOPSO (DMOPSO) each individual or particle provides a value for each decision variable, i.e. a complete solution. However, dCOEA's individuals only provide a value for one decision variable. Therefore, DNSGA-II, DVEPSO and DMOPSO were each assigned 100 individuals and dCOEA were assigned $100n_x$ individuals. The selected configuration for each of the DMOO algorithms is presented in Tables 11.1 and 11.2.

11.1.2 Experiments Comparing the Performance of Dynamic Multi-objective Optimisation Algorithms

Similar to the experiments discussed in Section 11.1.1, all experiments comparing the performance of the DMOAs consisted of 30 independent runs and each run continued

Table 11.1: Parameter values of the DMOEAs

DMOEA	n_i	p_c	p_m	**Other**
DNSGA-II-A	100	0.9	$\frac{4}{x_k}$	polynomial mutation, SBX crossover, $\zeta = 30$
DNSGA-II-B	100	0.8	$\frac{5}{x_k}$	polynomial mutation, SBX crossover, $\zeta = 30$
dCOEA	$100x_k$	0.5	$\frac{9}{L}$	$SC_{ratio} = 0.7$, $R_{size} = 5$

Table 11.2: Parameter values of the PSO-based DMOAs

DMOA	n_i	c_1	c_2	w	**Other**
DMOPSO	100	1.49	1.49	0.72	self-adapting σ_{share}
DVEPSO	100	1.49	1.49	0.72	guide update: p_s-g_r, boundary violation management: *cl*, knowledge sharing: *ra-t*, response applied to archive: *ac*, response applied to particles: *ri*-30-*c*

for 1000 iterations. For all benchmark functions, the severity of change (n_t) was set to 1, 10 and 20 and the frequency of change (τ_t) was set to 10, 25 and 50.

Eighteen benchmark functions were used to compare the performance of the five DMOO algorithms. The fifteen benchmark functions discussed in Section 9.4.1 were selected. In addition, three three-objective DMOOPs were used, namely FDA5, FDA5$_{iso}$ and FDA5$_{dec}$.

Three performance measures were used to quantify the performance of algorithms as discussed in Section 9.4.1, namely *acc*, *stab* and NS.

Statistical analysis of the obtained data was performed as discussed in Section 9.4.1. The null hypotheses for these experiments was that there is no statistical significant difference between the performance of the five DMOO algorithms. The alternative hypothesis is that there is a difference in mean performance.

11.2 Results

This section presents the results obtained by the various DMOAs. The results are discussed considering the various n_t-τ_t combinations, with regards to three performance measures and with regards to DMOOP Types I to III. General observations are also highlighted. Tables 11.3 to 11.28 present the wins and losses. Only the tables highlighting interesting trends and that are therefore discussed, are presented in this section. The other wins and losses tables are presented in Appendix D. Only statistical significant values are included in the tables. The p-values obtained for the various Mann-Whitney U tests, as well as the average performance measure values, are presented in Appendix D.

Results with regards to Performance Measures

Table 11.3 presents the wins and losses for each performance measure calculated over all DMOOPs and all n_t-τ_t combinations.

Table 11.3: Overall Wins and Losses for Various Performance Measures

PM	Results	DMOO Algorithm				
		DNSGA-II-A	**DNSGA-II-B**	**dCOEA**	**DMOPSO**	**DVEPSO**
acc	Wins	94	109	129	118	119
acc	Losses	120	104	164	86	95
acc	Diff	-26	5	-35	32	24
acc	Rank	4	3	5	**1**	2
stab	Wins	67	94	39	117	96
stab	Losses	89	66	200	39	50
stab	Diff	-22	28	-161	78	46
stab	Rank	4	3	5	**1**	2
NS	Wins	185	187	116	53	111
NS	Losses	83	78	202	195	129
NS	Diff	102	109	-86	-142	-18
NS	Rank	2	**1**	4	5	3

The following observations are made:

- DMOPSO obtained the best performance for *acc* and *stab* and ranked the worst for NS.
- DNSGA-II-B performed the best for NS.
- DNSGA-II-A and dCOEA were awarded more losses than wins for both *acc* and *stab*. In addition, dCOEA also obtained more losses than wins for NS.

- Both DNSGA-II algorithms obtained more wins than losses for NS. All other algorithms obtained more losses than wins for NS.
- The worst rank for *acc* and *stab* was obtained by dCOEA. Furthermore, DMOPSO obtained the worst rank for NS.
- DVEPSO obtained the second best rank for *acc* and *stab*, and the third best rank for NS.

Results with regards to Various Frequencies and Severities of Change

The wins and losses calculated over all performance measures and DMOOPs for the various n_t-τ_t combinations are presented in Table 11.4.

Table 11.4: Overall Wins and Losses for Various Frequencies and Severities of Change

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	65	77	91	62	89
10	10	all	Losses	97	84	113	97	59
10	10	all	Diff	-32	-7	-22	-35	30
10	10	all	Rank	4	2	3	5	**1**
10	25	all	Wins	72	80	40	46	55
10	25	all	Losses	37	31	114	60	51
10	25	all	Diff	35	49	-74	-14	4
10	25	all	Rank	2	**1**	5	4	3
10	50	all	Wins	69	76	47	35	49
10	50	all	Losses	36	32	93	63	52
10	50	all	Diff	33	44	-46	-28	-3
10	50	all	Rank	2	**1**	5	4	3
1	10	all	Wins	66	72	49	87	73
1	10	all	Losses	72	62	115	42	56
1	10	all	Diff	-6	10	-66	45	17
1	10	all	Rank	4	3	5	**1**	2
20	10	all	Wins	74	85	57	58	60
20	10	all	Losses	50	39	131	58	56
20	10	all	Diff	24	46	-74	0	4
20	10	all	Rank	2	**1**	5	4	3

With regards to the various environments, the following are observed:

- Mixed results were obtained with regards to the various environments. DVEPSO performed the best for $n_t = 10$ and $\tau_t = 10$, DMOPSO performed the best for $n_t = 1$ and $\tau_t = 10$ and DNSGA-II-B performed the best for $n_t = 10$ and $\tau_t = 25$,

$n_t = 10$ and $\tau_t = 50$, and $n_t = 20$ and $\tau_t = 10$.

- DVEPSO was ranked in the top three for all environments.
- The worst performance for all environments, except $n_t = 10$ and $\tau_t = 10$, was obtained by dCOEA. For $n_t = 10$ and $\tau_t = 10$, DMOPSO performed the worst.
- DNSGA-II-B and DVEPSO performed well, obtaining more wins than losses for all environments, except one. More losses than wins were obtained by dCOEA for all environments. DMOPSO obtained more losses than wins for three environments, namely $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$. DNSGA-II-A was awarded more losses than wins for two environments, namely $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$.

From Table 11.3 it is clear that the wins and losses obtained for NS may scew the results. Therefore, the wins and losses calculated over all DMOOPs and over *acc* and *stab* (excluding NS) for the different environment types are presented in Table 11.5.

Table 11.5: Overall Wins and Losses for Various Frequencies and Severities of Change measured over *acc* and *stab*

$\mathbf{n_t}$	$\tau_\mathbf{t}$	**PM**	**Results**	**DMOO Algorithm**				
				DNSGA-II-A	**DNSGA-II-B**	**dCOEA**	**DMOPSO**	**DVEPSO**
10	10	all	Wins	33	42	48	57	67
10	10	all	Losses	66	57	85	40	30
10	10	all	Diff	-33	-15	-37	17	37
10	10	all	Rank	4	3	5	2	**1**
10	25	all	Wins	34	41	16	38	32
10	25	all	Losses	24	19	76	18	24
10	25	all	Diff	10	22	-60	20	8
10	25	all	Rank	3	**1**	5	2	4
10	50	all	Wins	32	39	25	29	27
10	50	all	Losses	22	17	59	26	28
10	50	all	Diff	10	22	-34	3	-1
10	50	all	Rank	2	**1**	5	3	4
1	10	all	Wins	28	37	34	63	51
1	10	all	Losses	54	44	65	19	31
1	10	all	Diff	-26	-7	-31	44	20
1	10	all	Rank	4	3	5	**1**	2
20	10	all	Wins	34	44	45	48	38
20	10	all	Losses	43	33	79	22	32
20	10	all	Diff	-9	11	-34	26	6
20	10	all	Rank	4	2	5	**1**	3

The following observations are made:

- DVEPSO performed well in fast environments, obtaining the first, second and third rank in $n_t = 10$ and $\tau_t = 10$, $n_t = 1$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$, respectively. In slower changing environments it obtained the second lowest rank.
- DMOPSO performed well, being ranked in the top three in all environments and being ranked the best for $n_t = 1$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$.
- In slower changing environments ($n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$) DNSGA-II-B performed the best. In addition, DNSGA-II-B was ranked in the top three in all environments, similar to DMOPSO.
- DNSGA-II-A performed not so good, obtaining the second lowest rank in all environments with $\tau_t = 10$. In slower changing environments ($n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$) it obtained the second and third best rank.

Results for Various Dynamic Multi-objective Optimisation Problem Types

For the different DMOOP Types, the POS or POF or both changes over time. This section discusses the performance of the DMOAs with regards to the DMOOP Types I, II and III.

Type I DMOOPs

This section discusses the wins and losses of the guide update approaches for Type I DMOOPs, namely DIMP2 and dMOP3. The wins and losses for Type I DMOOPs obtained by the DMOAs over all n_t-τ_t combinations are presented in Table 11.6.
The following observations are made:

- Both DVEPSO and dCOEA obtained the best rank for *acc*. DNSGA-II-A performed the worst with regards to *acc* and was awarded 10 more losses than wins.
- With regards to *stab*, DNSGA-II-B and DMOPSO performed the best and dCOEA obtained the third best rank. The worst performance for *stab* was obtained by DVEPSO, obtaining thirteen more losses than wins.
- For NS, DNSGA-II-B performed the best and DVEPSO the worst. In addition, DNSGA-II-B was the only MOAs that obtained more wins than losses for NS.

Table 11.6: Overall Wins and Losses solving Type I DMOOPs for Various Performance Measures

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	*acc*	Wins	7	11	23	9	20
all	all	*acc*	Losses	17	13	15	13	12
all	all	*acc*	Diff	-10	-2	8	-4	8
all	all	*acc*	Rank	5	3	**1**	4	**1**
all	all	*stab*	Wins	6	15	6	13	1
all	all	*stab*	Losses	18	14	14	12	14
all	all	*stab*	Diff	-12	1	-8	1	-13
all	all	*stab*	Rank	4	**1**	3	**1**	5
all	all	*NS*	Wins	14	16	8	6	0
all	all	*NS*	Losses	15	13	17	19	15
all	all	*NS*	Diff	-1	3	-9	-13	-15
all	all	*NS*	Rank	2	**1**	3	4	5

Table 11.7 presents the wins and losses for Type I DMOOPs obtained by the DMOAs in various types of environments.

The following is observed for the various n_t-τ_t combinations:

- For $n_t = 10$ and $\tau_t = 10$, dCOEA performed the best and DNSGA-II-A performed the worst. All algorithms, except dCOEA, obtained more losses than wins.
- DNSGA-II-B obtained the best rank for slow changing environments, namely $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$. For $n_t = 10$ and $\tau_t = 25$, dCOEA performed the worst and for $n_t = 10$ and $\tau_t = 50$, DMOPSO performed the worst.
- In fast and severely changing environments ($n_t = 1$ and $\tau_t = 10$), DMOPSO obtained the best rank and was the only algorithm being awarded more wins than losses. For $n_t = 1$ and $\tau_t = 10$, the worst performance was obtained by DVEPSO.
- In slow and gradually changing environments ($n_t = 20$ and $\tau_t = 10$), DNSGA-II-B performed the best and DNSGA-II-A performed the worst.

Table 11.7: Overall Wins and Losses solving Type I DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	4	7	12	6	4
10	10	all	Losses	14	10	5	9	9
10	10	all	Diff	-10	-3	7	-3	-5
10	10	all	Rank	5	2	**1**	2	4
10	25	all	Wins	5	8	6	5	4
10	25	all	Losses	8	7	11	8	7
10	25	all	Diff	-3	1	-5	-3	-3
10	25	all	Rank	2	**1**	5	2	2
10	50	all	Wins	9	12	7	2	4
10	50	all	Losses	8	6	11	13	9
10	50	all	Diff	1	6	-4	-11	-5
10	50	all	Rank	2	**1**	3	5	4
1	10	all	Wins	5	6	6	9	4
1	10	all	Losses	9	10	9	6	9
1	10	all	Diff	-4	-4	-3	3	-5
1	10	all	Rank	3	3	2	**1**	5
20	10	all	Wins	4	9	6	6	5
20	10	all	Losses	11	7	10	8	7
20	10	all	Diff	-7	2	-4	-2	-2
20	10	all	Rank	5	**1**	4	2	2

The wins and losses for all Type I DMOOPs over all performance measures and all n_t-τ_t combinations are presented in Table 11.8. DNSGA-II-B performed the best, with dCOEA obtaining the second best rank. DNSGA-II-B was the only algorithm that obtained more wins than losses, while all the other MOAs obtained more losses than wins.

Table 11.8: Overall Wins and Losses solving Type I DMOOPs

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	27	42	37	28	21
all	all	all	Losses	50	40	46	44	41
all	all	all	Diff	-23	2	-9	-16	-20
all	all	all	Rank	5	**1**	2	3	4

Furthermore, it should be noted that only DVEPSO and dCOEA were able to solve DIMP2, and DVEPSO was the only algorithm that converged successfully to the true

POF for DIMP2. The DNSGA-II algorithms found only one solution per run and DMOPSO did not find any solutions.

The wins and losses obtained by DVEPSO and dCOEA for DIMP2 are presented in Table 11.9. The following are observed:

- DVEPSO completely outperformed dCOEA with regards to *acc*.
- For $n_t = 10$ and $\tau_t = 10$ there was no statistically significant difference in the *stab* values of the two algorithms. However, for all other n_t-τ_t combinations DVEPSO outperformed dCOEA.
- There was no statistically significant difference in the performance of DVEPSO and dCOEA with regards to NS.
- Measuring the performance of the algorithms for each n_t-τ_t combination over all performance measures, DVEPSO outperformed dCOEA for all n_t-τ_t combinations.
- DVEPSO completely outperformed dCOEA, obtaining 9 wins, with dCOEA obtaining zero wins.

Table 11.9: Wins and Losses of DIMP2 obtained by the DMOO algorithms

$\mathbf{n_t}$	$\tau_\mathbf{t}$	**PM**	**Results**	**DMOO Algorithm**	
				dCOEA	**DVEPSO**
10	10	*acc*	Wins	0	1
10	10	*acc*	Losses	1	0
10	10	*acc*	Diff	-1	1
10	10	*acc*	Rank	2	**1**
10	25	*acc*	Wins	0	1
10	25	*acc*	Losses	1	0
10	25	*acc*	Diff	-1	1
10	25	*acc*	Rank	2	**1**
10	50	*acc*	Wins	0	1
10	50	*acc*	Losses	1	0
10	50	*acc*	Diff	-1	1
10	50	*acc*	Rank	2	**1**
1	10	*acc*	Wins	0	1
1	10	*acc*	Losses	1	0
1	10	*acc*	Diff	-1	1
1	10	*acc*	Rank	2	**1**
20	10	*acc*	Wins	0	1
20	10	*acc*	Losses	1	0
20	10	*acc*	Diff	-1	1
20	10	*acc*	Rank	2	**1**
all	all	*acc*	Wins	0	5
all	all	*acc*	Losses	5	0
all	all	*acc*	Diff	-5	5
all	all	*acc*	Rank	2	**1**

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm	
				dCOEA	DVEPSO
10	10	*stab*	Wins	0	0
10	10	*stab*	Losses	0	0
10	10	*stab*	Diff	0	0
10	10	*stab*	Rank	**1**	**1**
10	25	*stab*	Wins	0	1
10	25	*stab*	Losses	1	0
10	25	*stab*	Diff	-1	1
10	25	*stab*	Rank	2	**1**
10	50	*stab*	Wins	0	1
10	50	*stab*	Losses	1	0
10	50	*stab*	Diff	-1	1
10	50	*stab*	Rank	2	**1**
1	10	*stab*	Wins	0	1
1	10	*stab*	Losses	1	0
1	10	*stab*	Diff	-1	1
1	10	*stab*	Rank	2	**1**
20	10	*stab*	Wins	0	1
20	10	*stab*	Losses	1	0
20	10	*stab*	Diff	-1	1
20	10	*stab*	Rank	2	**1**
all	all	*stab*	Wins	0	4
all	all	*stab*	Losses	4	0
all	all	*stab*	Diff	-4	4
all	all	*stab*	Rank	2	**1**
10	10	NS	Wins	0	0
10	10	NS	Losses	0	0
10	10	NS	Diff	0	0
10	10	NS	Rank	**1**	**1**
10	25	NS	Wins	0	0
10	25	NS	Losses	0	0
10	25	NS	Diff	0	0
10	25	NS	Rank	**1**	**1**
10	50	NS	Wins	0	0
10	50	NS	Losses	0	0
10	50	NS	Diff	0	0
10	50	NS	Rank	**1**	**1**
1	10	NS	Wins	0	0
1	10	NS	Losses	0	0
1	10	NS	Diff	0	0
1	10	NS	Rank	**1**	**1**
20	10	NS	Wins	0	0
20	10	NS	Losses	0	0
20	10	NS	Diff	0	0
20	10	NS	Rank	**1**	**1**
all	all	NS	Wins	0	0
all	all	NS	Losses	0	0
all	all	NS	Diff	0	0
all	all	NS	Rank	**1**	**1**
10	10	all	Wins	0	1
10	10	all	Losses	1	0

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm	
				dCOEA	DVEPSO
10	10	all	Diff	-1	1
10	10	all	Rank	2	**1**
10	25	all	Wins	0	2
10	25	all	Losses	2	0
10	25	all	Diff	-2	2
10	25	all	Rank	2	**1**
10	50	all	Wins	0	2
10	50	all	Losses	2	0
10	50	all	Diff	-2	2
10	50	all	Rank	2	**1**
1	10	all	Wins	0	2
1	10	all	Losses	2	0
1	10	all	Diff	-2	2
1	10	all	Rank	2	**1**
20	10	all	Wins	0	2
20	10	all	Losses	2	0
20	10	all	Diff	-2	2
20	10	all	Rank	2	**1**
all	all	all	Wins	0	9
all	all	all	Losses	9	0
all	all	all	Diff	-9	9
all	all	all	Rank	2	**1**

Type II DMOOPs

The wins and losses for Type II DMOOPs obtained by the DMOAs over all n_t-τ_t combinations are presented in Table 11.10. The Type II DMOOPs are FDA1$_{Zhou}$, FDA2, FDA3, FDA3$_{Camara}$, dMOP2, dMOP2$_{iso}$, dMOP2$_{dec}$, FDA5, FDA5$_{iso}$ and FDA5$_{dec}$.

Table 11.10: Overall Wins and Losses solving Type II DMOOPs for Various Performance Measures

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	*acc*	Wins	55	55	36	56	63
all	all	*acc*	Losses	43	41	106	41	34
all	all	*acc*	Diff	12	14	-70	15	29
all	all	*acc*	Rank	4	3	5	2	**1**
all	all	*stab*	Wins	36	45	18	53	59
all	all	*stab*	Losses	43	30	104	20	14
all	all	*stab*	Diff	-7	15	-86	33	45
all	all	*stab*	Rank	4	3	5	2	**1**
all	all	*NS*	Wins	96	92	60	31	90
all	all	*NS*	Losses	48	50	116	105	50
all	all	*NS*	Diff	48	42	-56	-74	40
all	all	*NS*	Rank	**1**	2	4	5	3

The following observations are made:

- DVEPSO performed the best with regards to *acc* and *stab*. The best performance with regards to NS was obtained by DNSGA-II-A.
- The worst performance for both *acc* and *stab* was obtained by dCOEA. For NS, DMOPSO performed the worst.
- dCOEA performed poorly with regards to *acc*, being awarded more losses than wins.
- For *stab*, both DNSGA-II-A and dCOEA obtained more losses than wins.
- Only DMOPSO and dCOEA obtained more losses than wins for NS. All other algorithms performed well, obtaining more wins than losses for NS.

Table 11.11 presents the wins and losses for Type II DMOOPs obtained by the DMOAs in various types of environments.

Table 11.11: Overall Wins and Losses solving Type II DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	39	39	29	31	52
10	10	all	Losses	32	29	69	41	19
10	10	all	Diff	7	10	-40	-10	33
10	10	all	Rank	3	2	5	4	**1**
10	25	all	Wins	39	40	13	22	38
10	25	all	Losses	16	16	66	35	19
10	25	all	Diff	23	24	-53	-13	19
10	25	all	Rank	2	**1**	5	4	3
10	50	all	Wins	37	39	15	11	28
10	50	all	Losses	12	12	52	33	21
10	50	all	Diff	25	27	-37	-22	7
10	50	all	Rank	2	**1**	5	4	3
1	10	all	Wins	33	34	23	47	55
1	10	all	Losses	47	40	67	23	15
1	10	all	Diff	-14	-6	-44	24	40
1	10	all	Rank	4	3	5	2	**1**
20	10	all	Wins	39	40	34	29	39
20	10	all	Losses	27	24	72	34	24
20	10	all	Diff	12	16	-38	-5	15
20	10	all	Rank	3	**1**	5	4	2

The following are observed with regards to the various n_t-τ_t combinations:

- In fast changing environments ($n_t = 1$ and $\tau_t = 10$, and $n_t = 10$ and $\tau_t = 10$) DVEPSO obtained the best rank. Furthermore, DVEPSO was ranked in the top three for all environments.
- DNSGA-II-B obtained the best rank for $n_t = 10$ and $\tau_t = 25$, $n_t = 10$ and $\tau_t = 50$ and $n_t = 20$ and $\tau_t = 10$. Similar to DVEPSO, DNSGA-II-B was ranked in the top three for all environments.
- dCOEA performed the worst for all environments.
- DVEPSO performed really well, being the only algorithm that obtained more wins than losses for all environments. DNSGA-II-A and DNSGA-II-B also performed well, being awarded more losses than wins for only $n_t = 1$ and $\tau_t = 10$. More losses than wins were obtained by DMOPSO for all environments, except $n_t = 1$ and $\tau_t = 10$. dCOEA performed poorly, being awarded more losses than wins for all environments.

The results indicate that when solving Type II DMOOPs, DMOPSO obtains so much losses for NS, that it performs poorly even though its performance for *acc* and *stab* is good (refer to Table 11.10). Table 11.12 presents the wins and losses for Type II DMOOPs for various n_t-τ_t combinations, measured over *acc* and *stab* and therefore not taking NS into account.

The following are observed:

- DVEPSO performed the best for $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$. For $n_t = 20$ and $\tau_t = 10$, DVEPSO obtained the second best rank and for the rest of the environments DVEPSO obtained the third best rank.
- DNSGA-II-B was awarded the best rank for $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$. For all other environments, DNSGA-II-B obtained the third best rank.
- DMOPSO performed the best for $n_t = 20$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$, DMOPSO obtained the second best rank. However, for $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$, DMOPSO was awarded the second lowest rank.
- dCOEA performed the worst for all environments.

Table 11.12: Overall Wins and Losses solving Type II DMOOPs for Various Frequencies and Severities of Change measured over *acc* and *stab*

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	22	21	6	28	33
10	10	all	Losses	19	17	53	13	8
10	10	all	Diff	3	4	-47	15	25
10	10	all	Rank	4	3	5	2	**1**
10	25	all	Wins	19	21	1	18	19
10	25	all	Losses	8	7	44	10	9
10	25	all	Diff	11	14	-43	8	10
10	25	all	Rank	2	**1**	5	4	3
10	50	all	Wins	17	20	5	9	12
10	50	all	Losses	6	4	32	12	9
10	50	all	Diff	11	16	-27	-3	3
10	50	all	Rank	2	**1**	5	4	3
1	10	all	Wins	14	18	15	30	35
1	10	all	Losses	31	24	38	12	7
1	10	all	Diff	-17	-6	-23	18	28
1	10	all	Rank	4	3	5	2	**1**
20	10	all	Wins	19	20	27	24	23
20	10	all	Losses	22	19	43	14	15
20	10	all	Diff	-3	1	-16	10	8
20	10	all	Rank	4	3	5	**1**	2

The wins and losses for all Type II DMOOPs over all performance measures and all n_t-τ_t combinations are presented in Table 11.13. DVEPSO outperformed the other algorithms, obtaining the best rank and being awarded 114 more wins than losses. The worst rank was awarded to dCOEA. Both DMOPSO and dCOEA performed poorly, obtaining more losses than wins. In contrast, both DNSGA-II algorithms were awarded more wins than losses, with DNSGA-II-B and DNSGA-II-A obtaining the second and third best rank respectively.

Table 11.13: Overall Wins and Losses solving Type II DMOOPs

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	187	192	114	140	212
all	all	all	Losses	134	121	326	166	98
all	all	all	Diff	53	71	-212	-26	114
all	all	all	Rank	3	2	5	4	**1**

The wins and losses measured over *acc* and *stab* (and not taking NS into account) are presented in Table 11.14. DVEPSO performed the best and DMOPSO obtained the second best rank. dCOEA performed the worst, obtaining 156 more losses than wins.

Table 11.14: Overall Wins and Losses solving Type II DMOOPs measured over *acc* and *stab*

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	91	100	54	109	122
all	all	all	Losses	86	71	210	61	48
all	all	all	Diff	5	29	-156	48	74
all	all	all	Rank	4	3	5	2	**1**

Type III DMOOPs

Table 11.15 presents the wins and losses with regards to the various performance measures obtained by the DMOA for Type III DMOOPs. The Type III DMOOPs are HE1, HE2, HE6 to HE9 and $FDA2_{Camara}$.

Table 11.15: Overall Wins and Losses solving Type III DMOOPs for Various Performance Measures

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	*acc*	Wins	32	43	70	53	36
all	all	*acc*	Losses	60	50	43	32	49
all	all	*acc*	Diff	-28	-7	27	21	-13
all	all	*acc*	Rank	5	3	**1**	2	4
all	all	*stab*	Wins	25	34	15	51	36
all	all	*stab*	Losses	28	22	82	7	22
all	all	*stab*	Diff	-3	12	-67	44	14
all	all	*stab*	Rank	4	3	5	**1**	2
all	all	NS	Wins	75	79	48	16	21
all	all	NS	Losses	20	15	69	71	64
all	all	NS	Diff	55	64	-21	-55	-43
all	all	NS	Rank	2	**1**	3	5	4

The following observations are made:

- dCOEA performed the best with regards to *acc*, with DNSGA-II-A performing the worst. Only dCOEA and DMOPSO obtained more wins than losses for *acc*.
- For *stab*, DMOPSO obtained the best performance and dCOEA performed the worst. Two DMOAs, dCOEA and DNSGA-II-A, were awarded more losses than

wins for *stab*.

- The best rank for NS was obtained by DNSGA-II-B and the worst rank was awarded to DMOPSO. More wins than losses were obtained by dCOEA and DMOPSO.
- DVEPSO obtained the second lowest rank for *acc* and NS, since it struggled to converge to discontinous POFs. For *stab*, DVEPSO obtained the second best rank.

The wins and losses with regards to the various environment types for Type III DMOOPs are presented in Table 11.16.

Table 11.16: Overall Wins and Losses solving Type III DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	22	27	38	25	17
10	10	all	Losses	27	21	27	23	31
10	10	all	Diff	-5	6	11	2	-14
10	10	all	Rank	4	2	**1**	3	5
10	25	all	Wins	28	33	24	19	17
10	25	all	Losses	19	14	40	23	25
10	25	all	Diff	9	19	-16	-4	-8
10	25	all	Rank	2	**1**	5	3	4
10	50	all	Wins	23	26	28	22	21
10	50	all	Losses	22	20	33	23	22
10	50	all	Diff	1	6	-5	-1	-1
10	50	all	Rank	2	**1**	5	3	3
1	10	all	Wins	28	33	23	31	18
1	10	all	Losses	22	18	42	19	32
1	10	all	Diff	6	15	-19	12	-14
1	10	all	Rank	3	**1**	5	2	4
20	10	all	Wins	31	37	20	23	20
20	10	all	Losses	18	14	52	22	25
20	10	all	Diff	13	23	-32	1	-5
20	10	all	Rank	2	**1**	5	3	4

For the various n_t-τ_t combinations, the following are observed:

- DNSGA-II-B obtained the best rank for all environments, except for $n_t = 10$ and $\tau_t = 10$, where it obtained the second best rank.
- dCOEA performed the worst in all environments, except $n_t = 10$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 10$, it performed the best.

- DMOPSO performed reasonably well, obtaining either the second or third best rank for all environments.
- DVEPSO obtained the third best rank for $n_t = 10$ and $\tau_t = 50$. However, it obtained the second lowest rank for $n_t = 10$ and $\tau_t = 25$, $n_t = 1$ and $\tau_t = 10$ and $n_t = 20$ and $\tau_t = 10$. Furthermore, DVEPSO obtained the worst rank for $n_t = 10$ and $\tau_t = 10$.
- DNSGA-II-B was the only algorithm that obtained more wins than losses for all n_t-τ_t combinations.

Table 11.17: Overall Wins and Losses solving Type III DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	all	Wins	22	27	38	25	17
10	10	all	Losses	27	21	27	23	31
10	10	all	Diff	-5	6	11	2	-14
10	10	all	Rank	4	2	**1**	3	5
10	25	all	Wins	28	33	24	19	17
10	25	all	Losses	19	14	40	23	25
10	25	all	Diff	9	19	-16	-4	-8
10	25	all	Rank	2	**1**	5	3	4
10	50	all	Wins	23	26	28	22	21
10	50	all	Losses	22	20	33	23	22
10	50	all	Diff	1	6	-5	-1	-1
10	50	all	Rank	2	**1**	5	3	3
1	10	all	Wins	28	33	23	31	18
1	10	all	Losses	22	18	42	19	32
1	10	all	Diff	6	15	-19	12	-14
1	10	all	Rank	3	**1**	5	2	4
20	10	all	Wins	31	37	20	23	20
20	10	all	Losses	18	14	52	22	25
20	10	all	Diff	13	23	-32	1	-5
20	10	all	Rank	2	**1**	5	3	4

Table 11.18 presents the wins and losses for Type III DMOOPs measured over all performance measures and all n_t-τ_t combinations. The best overall performance for Type III DMOOPs was obtained by DNSGA-II-B, with dCOEA performing the worst. DNSGA-II-A, DNSGA-II-B and DMOPSO were awarded more wins than losses. The DNSGA-II algorithms obtained the top two ranks and therefore outperformed the PSO-

based DMOAs.

Table 11.18: Overall Wins and Losses solving Type III DMOOPs

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	132	156	133	120	93
all	all	all	Losses	108	87	194	110	135
all	all	all	Diff	24	69	-61	10	-42
all	all	all	Rank	2	**1**	5	3	4

Overall Performance

This section discusses the overall performance of the DMOAs. The overall wins and losses over all DMOOPs, n_t-τ_t combinations and performance measures are presented in Table 11.19.

Table 11.19: Overall Wins and Losses by the various DMOO algorithms

Results	DMOO Algorithm				
	DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
Wins	346	390	284	288	326
Losses	292	248	566	320	274
Diff	54	142	-282	-32	52
Rank	2	**1**	5	4	3

DNSGA-II-B performed the best obtaining 142 more wins than losses. The second best performance was obtained by DNSGA-II-A, being awarded 54 more wins than losses. DVEPSO was awarded the third best rank, obtaining 52 more wins than losses. The second worst rank was obtained by dCOEA and DMOPSO performed the worst. All DMOEAs obtained more wins than losses. On the other hand, all PSO-based DMOAs obtained more losses than wins. Therefore, the DMOEAs completely outperformed the PSO-based DMOAs.

However, many wins and losses obtained by the DMOEAs are for NS. Table 11.20 represents the overall wins and losses over all DMOOPs and n_t-τ_t combinations, without taking NS into account. The effect of the wins and losses with regards to NS can clearly be seen. DMOPSO now obtains the highest rank with 110 more wins than losses. The second best rank is awarded to DVEPSO, with DVEPSO being awarded 70 more wins than losses. DNSGA-II-B is ranked third, obtaining 33 more wins than losses.

DNSGA-II-A and dCOEA obtains the second lowest and lowest rank respectively, with both algorithms being awarded more losses than wins. Therefore, with regards to *acc* and *stab*, the PSO-based DMOAs completely outperforms the DMOEAs.

Table 11.20: Overall Wins and Losses for *acc* and *stab* by the various DMOO algorithms

Results	**DMOO Algorithm**				
	DNSGA-II-A	**DNSGA-II-B**	**dCOEA**	**DMOPSO**	**DVEPSO**
Wins	161	203	168	235	215
Losses	209	170	364	125	145
Diff	-48	33	-196	110	70
Rank	4	3	5	**1**	2

General Observations

This section discusses general observations that were made about the performance of the DMOAs. Trends that vary from the overall performance of the DMOA (presented in Table 11.19) are highlighted below.

The difference in performance for dMOP2, dMOP2$_{iso}$ and dMOP2$_{dec}$ are presented in Tables 11.21 to 11.23. For dMOP2, DVEPSO obtained the best overall performance and DMOPSO obtained the second best rank. The lowest two ranks were awarded to DNSGA-II-A and dCOEA, with dCOEA obtaining more losses than wins. In contrast, for dMOP2$_{iso}$ the DNSGA-II algorithms obtained the top two ranks, with DVEPSO obtaining the third best rank. The DNSGA-II algorithms and DVEPSO obtained more wins than losses. Furthermore, dCOEA performed the worst. For dMOP2$_{dec}$, DVEPSO performed the best, with DNSGA-II-B and DNSGA-II-A obtaining the second and third best rank respectively. dCOEA performed the worst, and was the only algorithm that was awarded more losses than wins.

Table 11.21: Wins and Losses of dMOP2 obtained by the DMOO algorithms

$\mathbf{n_t}$	$\tau_\mathbf{t}$	**PM**	**Results**	**DMOO Algorithm**				
				DNSGA-II-A	**DNSGA-II-B**	**dCOEA**	**DMOPSO**	**DVEPSO**
10	10	*acc*	Wins	0	0	0	3	3
10	10	*acc*	Losses	2	2	2	0	0
10	10	*acc*	Diff	-2	-2	-2	3	3
10	10	*acc*	Rank	3	3	3	**1**	**1**
10	25	*acc*	Wins	1	1	0	1	1
10	25	*acc*	Losses	0	0	4	0	0

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	25	*acc*	Diff	1	1	-4	1	1
10	25	*acc*	Rank	**1**	**1**	5	**1**	**1**
10	50	*acc*	Wins	3	4	0	1	1
10	50	*acc*	Losses	1	0	4	2	2
10	50	*acc*	Diff	2	4	-4	-1	-1
10	50	*acc*	Rank	2	**1**	5	3	3
1	10	*acc*	Wins	0	1	1	3	3
1	10	*acc*	Losses	3	2	3	0	0
1	10	*acc*	Diff	-3	-1	-2	3	3
1	10	*acc*	Rank	5	3	4	**1**	**1**
20	10	*acc*	Wins	1	1	0	1	1
20	10	*acc*	Losses	0	0	4	0	0
20	10	*acc*	Diff	1	1	-4	1	1
20	10	*acc*	Rank	**1**	**1**	5	**1**	**1**
all	all	*acc*	Wins	5	7	1	9	9
all	all	*acc*	Losses	6	4	17	2	2
all	all	*acc*	Diff	-1	3	-16	7	7
all	all	*acc*	Rank	4	3	5	**1**	**1**
10	10	*stab*	Wins	1	1	0	1	1
10	10	*stab*	Losses	0	0	4	0	0
10	10	*stab*	Diff	1	1	-4	1	1
10	10	*stab*	Rank	**1**	**1**	5	**1**	**1**
10	25	*stab*	Wins	1	1	0	1	1
10	25	*stab*	Losses	0	0	4	0	0
10	25	*stab*	Diff	1	1	-4	1	1
10	25	*stab*	Rank	**1**	**1**	5	**1**	**1**
10	50	*stab*	Wins	1	1	0	1	1
10	50	*stab*	Losses	0	0	4	0	0
10	50	*stab*	Diff	1	1	-4	1	1
10	50	*stab*	Rank	**1**	**1**	5	**1**	**1**
1	10	*stab*	Wins	0	0	0	3	3
1	10	*stab*	Losses	2	2	2	0	0
1	10	*stab*	Diff	-2	-2	-2	3	3
1	10	*stab*	Rank	3	3	3	**1**	**1**
20	10	*stab*	Wins	1	1	0	3	3
20	10	*stab*	Losses	2	2	4	0	0
20	10	*stab*	Diff	-1	-1	-4	3	3
20	10	*stab*	Rank	3	3	5	**1**	**1**
all	all	*stab*	Wins	4	4	0	9	9
all	all	*stab*	Losses	4	4	18	0	0
all	all	*stab*	Diff	0	0	-18	9	9
all	all	*stab*	Rank	3	3	5	**1**	**1**
10	10	*NS*	Wins	1	1	3	0	3
10	10	*NS*	Losses	2	2	1	3	0
10	10	*NS*	Diff	-1	-1	2	-3	3
10	10	*NS*	Rank	3	3	2	5	**1**
10	25	*NS*	Wins	2	2	0	0	2
10	25	*NS*	Losses	1	1	2	2	0
10	25	*NS*	Diff	1	1	-2	-2	2
10	25	*NS*	Rank	2	2	4	4	**1**

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	50	NS	Wins	2	2	0	0	2
10	50	NS	Losses	1	1	2	2	0
10	50	NS	Diff	1	1	-2	-2	2
10	50	NS	Rank	2	2	4	4	**1**
1	10	NS	Wins	2	1	0	3	3
1	10	NS	Losses	2	3	4	0	0
1	10	NS	Diff	0	-2	-4	3	3
1	10	NS	Rank	3	4	5	**1**	**1**
20	10	NS	Wins	2	2	0	1	3
20	10	NS	Losses	1	1	4	2	0
20	10	NS	Diff	1	1	-4	-1	3
20	10	NS	Rank	2	2	5	4	**1**
all	all	NS	Wins	9	8	3	4	13
all	all	NS	Losses	7	8	13	9	0
all	all	NS	Diff	2	0	-10	-5	13
all	all	NS	Rank	2	3	5	4	**1**
10	10	all	Wins	2	2	3	4	7
10	10	all	Losses	4	4	7	3	0
10	10	all	Diff	-2	-2	-4	1	7
10	10	all	Rank	3	3	5	2	**1**
10	25	all	Wins	4	4	0	2	4
10	25	all	Losses	1	1	10	2	0
10	25	all	Diff	3	3	-10	0	4
10	25	all	Rank	2	2	5	4	**1**
10	50	all	Wins	6	7	0	2	4
10	50	all	Losses	2	1	10	4	2
10	50	all	Diff	4	6	-10	-2	2
10	50	all	Rank	2	**1**	5	4	3
1	10	all	Wins	2	2	1	9	9
1	10	all	Losses	7	7	9	0	0
1	10	all	Diff	-5	-5	-8	9	9
1	10	all	Rank	3	3	5	**1**	**1**
all	all	all	Wins	18	19	4	22	31
all	all	all	Losses	17	16	48	11	2
all	all	all	Diff	1	3	-44	11	29
all	all	all	Rank	4	3	5	2	**1**

Table 11.22: Wins and Losses of dMOP2$_{iso}$ obtained by the DMOO algorithms

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	acc	Wins	3	3	2	0	0
10	10	acc	Losses	0	0	2	3	3
10	10	acc	Diff	3	3	0	-3	-3
10	10	acc	Rank	**1**	**1**	3	4	4
10	25	acc	Wins	3	3	0	0	0
10	25	acc	Losses	0	0	2	2	2
10	25	acc	Diff	3	3	-2	-2	-2
10	25	acc	Rank	**1**	**1**	3	3	3

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	50	*acc*	Wins	3	3	1	0	1
10	50	*acc*	Losses	0	0	3	3	2
10	50	*acc*	Diff	3	3	-2	-3	-1
10	50	*acc*	Rank	**1**	**1**	4	5	3
1	10	*acc*	Wins	3	3	0	1	1
1	10	*acc*	Losses	0	0	4	2	2
1	10	*acc*	Diff	3	3	-4	-1	-1
1	10	*acc*	Rank	**1**	**1**	5	3	3
20	10	*acc*	Wins	3	3	0	1	1
20	10	*acc*	Losses	0	0	4	2	2
20	10	*acc*	Diff	3	3	-4	-1	-1
20	10	*acc*	Rank	**1**	**1**	5	3	3
all	all	*acc*	Wins	15	15	3	2	3
all	all	*acc*	Losses	0	0	15	12	11
all	all	*acc*	Diff	15	15	-12	-10	-8
all	all	*acc*	Rank	**1**	**1**	5	4	3
10	10	*stab*	Wins	2	2	0	0	2
10	10	*stab*	Losses	1	1	2	2	0
10	10	*stab*	Diff	1	1	-2	-2	2
10	10	*stab*	Rank	2	2	4	4	**1**
10	25	*stab*	Wins	3	3	0	1	1
10	25	*stab*	Losses	0	0	4	2	2
10	25	*stab*	Diff	3	3	-4	-1	-1
10	25	*stab*	Rank	**1**	**1**	5	3	3
10	50	*stab*	Wins	1	1	0	0	0
10	50	*stab*	Losses	0	0	2	0	0
10	50	*stab*	Diff	1	1	-2	0	0
10	50	*stab*	Rank	**1**	**1**	5	3	3
1	10	*stab*	Wins	2	2	0	1	3
1	10	*stab*	Losses	1	1	4	2	0
1	10	*stab*	Diff	1	1	-4	-1	3
1	10	*stab*	Rank	2	2	5	4	**1**
20	10	*stab*	Wins	2	3	0	1	3
20	10	*stab*	Losses	2	1	4	2	0
20	10	*stab*	Diff	0	2	-4	-1	3
20	10	*stab*	Rank	3	2	5	4	**1**
all	all	*stab*	Wins	10	11	0	3	9
all	all	*stab*	Losses	4	3	16	8	2
all	all	*stab*	Diff	6	8	-16	-5	7
all	all	*stab*	Rank	3	**1**	5	4	2
10	10	*NS*	Wins	1	1	1	0	2
10	10	*NS*	Losses	1	1	0	3	0
10	10	*NS*	Diff	0	0	1	-3	2
10	10	*NS*	Rank	3	3	2	5	**1**
10	25	*NS*	Wins	1	1	1	0	3
10	25	*NS*	Losses	1	1	1	3	0
10	25	*NS*	Diff	0	0	0	-3	3
10	25	*NS*	Rank	2	2	2	5	**1**
10	50	*NS*	Wins	1	1	1	0	3
10	50	*NS*	Losses	1	1	1	3	0

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	50	NS	Diff	0	0	0	-3	3
10	50	NS	Rank	2	2	2	5	**1**
1	10	NS	Wins	2	2	0	1	3
1	10	NS	Losses	1	1	4	2	0
1	10	NS	Diff	1	1	-4	-1	3
1	10	NS	Rank	2	2	5	4	**1**
20	10	NS	Wins	2	2	0	1	3
20	10	NS	Losses	1	1	4	2	0
20	10	NS	Diff	1	1	-4	-1	3
20	10	NS	Rank	2	2	5	4	**1**
all	all	NS	Wins	7	7	3	2	14
all	all	NS	Losses	5	5	10	13	0
all	all	NS	Diff	2	2	-7	-11	14
all	all	NS	Rank	2	2	4	5	**1**
10	10	all	Wins	6	6	3	0	4
10	10	all	Losses	2	2	4	8	3
10	10	all	Diff	4	4	-1	-8	1
10	10	all	Rank	**1**	**1**	4	5	3
10	25	all	Wins	7	7	1	1	4
10	25	all	Losses	1	1	7	7	4
10	25	all	Diff	6	6	-6	-6	0
10	25	all	Rank	**1**	**1**	4	4	3
10	50	all	Wins	5	5	2	0	4
10	50	all	Losses	1	1	6	6	2
10	50	all	Diff	4	4	-4	-6	2
10	50	all	Rank	**1**	**1**	4	5	3
1	10	all	Wins	7	7	0	3	7
1	10	all	Losses	2	2	12	6	2
1	10	all	Diff	5	5	-12	-3	5
1	10	all	Rank	**1**	**1**	5	4	**1**
20	10	all	Wins	7	8	0	3	7
20	10	all	Losses	3	2	12	6	2
20	10	all	Diff	4	6	-12	-3	5
20	10	all	Rank	3	**1**	5	4	2
all	all	all	Wins	32	33	6	7	26
all	all	all	Losses	9	8	41	33	13
all	all	all	Diff	23	25	-35	-26	13
all	all	all	Rank	2	**1**	5	4	3

Table 11.23: Wins and Losses of dMOP2$_{dec}$ obtained by the DMOO algorithms

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	acc	Wins	3	3	0	1	1
10	10	acc	Losses	0	0	4	2	2
10	10	acc	Diff	3	3	-4	-1	-1
10	10	acc	Rank	**1**	**1**	5	3	3
10	25	acc	Wins	3	3	0	1	1
10	25	acc	Losses	0	0	4	2	2

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	25	*acc*	Diff	3	3	-4	-1	-1
10	25	*acc*	Rank	**1**	**1**	5	3	3
10	50	*acc*	Wins	3	3	0	1	1
10	50	*acc*	Losses	0	0	4	2	2
10	50	*acc*	Diff	3	3	-4	-1	-1
10	50	*acc*	Rank	**1**	**1**	5	3	3
1	10	*acc*	Wins	3	3	0	1	1
1	10	*acc*	Losses	0	0	4	2	2
1	10	*acc*	Diff	3	3	-4	-1	-1
1	10	*acc*	Rank	**1**	**1**	5	3	3
20	10	*acc*	Wins	1	1	0	3	3
20	10	*acc*	Losses	2	2	4	0	0
20	10	*acc*	Diff	-1	-1	-4	3	3
20	10	*acc*	Rank	3	3	5	**1**	**1**
all	all	*acc*	Wins	13	13	0	7	7
all	all	*acc*	Losses	2	2	20	8	8
all	all	*acc*	Diff	11	11	-20	-1	-1
all	all	*acc*	Rank	**1**	**1**	5	3	3
10	10	*stab*	Wins	2	3	0	1	3
10	10	*stab*	Losses	2	1	4	2	0
10	10	*stab*	Diff	0	2	-4	-1	3
10	10	*stab*	Rank	3	2	5	4	**1**
10	25	*stab*	Wins	2	3	0	1	3
10	25	*stab*	Losses	2	1	4	2	0
10	25	*stab*	Diff	0	2	-4	-1	3
10	25	*stab*	Rank	3	2	5	4	**1**
10	50	*stab*	Wins	2	3	0	1	3
10	50	*stab*	Losses	2	1	4	2	0
10	50	*stab*	Diff	0	2	-4	-1	3
10	50	*stab*	Rank	3	2	5	4	**1**
1	10	*stab*	Wins	2	3	0	1	3
1	10	*stab*	Losses	2	1	4	2	0
1	10	*stab*	Diff	0	2	-4	-1	3
1	10	*stab*	Rank	3	2	5	4	**1**
20	10	*stab*	Wins	1	1	0	3	3
20	10	*stab*	Losses	2	2	4	0	0
20	10	*stab*	Diff	-1	-1	-4	3	3
20	10	*stab*	Rank	3	3	5	**1**	**1**
all	all	*stab*	Wins	9	13	0	7	15
all	all	*stab*	Losses	10	6	20	8	0
all	all	*stab*	Diff	-1	7	-20	-1	15
all	all	*stab*	Rank	3	2	5	3	**1**
10	10	*NS*	Wins	1	1	3	0	3
10	10	*NS*	Losses	2	2	1	3	0
10	10	*NS*	Diff	-1	-1	2	-3	3
10	10	*NS*	Rank	3	3	2	5	**1**
10	25	*NS*	Wins	1	1	1	0	3
10	25	*NS*	Losses	1	1	1	3	0
10	25	*NS*	Diff	0	0	0	-3	3
10	25	*NS*	Rank	2	2	2	5	**1**

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	50	NS	Wins	1	1	1	0	3
10	50	NS	Losses	1	1	1	3	0
10	50	NS	Diff	0	0	0	-3	3
10	50	NS	Rank	2	2	2	5	**1**
1	10	NS	Wins	1	1	0	3	3
1	10	NS	Losses	2	2	4	0	0
1	10	NS	Diff	-1	-1	-4	3	3
1	10	NS	Rank	3	3	5	**1**	**1**
20	10	NS	Wins	2	2	0	1	3
20	10	NS	Losses	1	1	4	2	0
20	10	NS	Diff	1	1	-4	-1	3
20	10	NS	Rank	2	2	5	4	**1**
all	all	NS	Wins	6	6	5	4	15
all	all	NS	Losses	7	7	11	11	0
all	all	NS	Diff	-1	-1	-6	-7	15
all	all	NS	Rank	2	2	4	5	**1**
10	10	all	Wins	6	7	3	2	7
10	10	all	Losses	4	3	9	7	2
10	10	all	Diff	2	4	-6	-5	5
10	10	all	Rank	3	2	5	4	**1**
10	25	all	Wins	6	7	1	2	7
10	25	all	Losses	3	2	9	7	2
10	25	all	Diff	3	5	-8	-5	5
10	25	all	Rank	3	**1**	5	4	**1**
10	50	all	Wins	6	7	1	2	7
10	50	all	Losses	3	2	9	7	2
10	50	all	Diff	3	5	-8	-5	5
10	50	all	Rank	3	**1**	5	4	**1**
1	10	all	Wins	6	7	0	5	7
1	10	all	Losses	4	3	12	4	2
1	10	all	Diff	2	4	-12	1	5
1	10	all	Rank	3	2	5	4	**1**
20	10	all	Wins	4	4	0	7	9
20	10	all	Losses	5	5	12	2	0
20	10	all	Diff	-1	-1	-12	5	9
20	10	all	Rank	3	3	5	2	**1**
all	all	all	Wins	28	32	5	18	37
all	all	all	Losses	19	15	51	27	8
all	all	all	Diff	9	17	-46	-9	29
all	all	all	Rank	3	2	5	4	**1**

Tables 11.24 to 11.26 present the wins and losses for FDA5, FDA5$_{iso}$ and FDA5$_{dec}$. For FDA5, dCOEA obtained the best overall performance and DNSGA-II-B obtained the second best rank. The worst performance was obtained by DVEPSO. Both PSO-based DMOAs were awarded more losses than wins. However, all other DMOAs obtained more wins than losses. The best overall performance for FDA5$_{iso}$ was obtained by DNSGA-II-A, with DNSGA-II-B obtaining the second best rank. dCOEA performed the worst,

obtaining the worst rank for all performance measures. For FDA5$_{dec}$, both DNSGA-II algorithms performed the best and both PSO-based DMOAs performed the worst. dCOEA performed the best with regards to *acc* and *stab*, but the worst with regards to NS.

Table 11.24: Wins and Losses of FDA5

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
1	10	*acc*	Wins	2	0	0	0	0
1	10	*acc*	Losses	0	0	0	1	1
1	10	*acc*	Diff	2	0	0	-1	-1
1	10	*acc*	Rank	**1**	2	2	4	4
20	10	*acc*	Wins	0	0	4	0	0
20	10	*acc*	Losses	1	1	0	1	1
20	10	*acc*	Diff	-1	-1	4	-1	-1
20	10	*acc*	Rank	2	2	**1**	2	2
all	all	*acc*	Wins	2	0	4	0	0
all	all	*acc*	Losses	1	1	0	2	2
all	all	*acc*	Diff	1	-1	4	-2	-2
all	all	*acc*	Rank	2	3	**1**	4	4
1	10	*stab*	Wins	0	1	0	1	1
1	10	*stab*	Losses	3	0	0	0	0
1	10	*stab*	Diff	-3	1	0	1	1
1	10	*stab*	Rank	5	**1**	4	**1**	**1**
all	all	*stab*	Wins	0	1	4	1	1
all	all	*stab*	Losses	4	1	0	1	1
all	all	*stab*	Diff	-4	0	4	0	0
all	all	*stab*	Rank	5	2	**1**	2	2
10	10	NS	Wins	2	2	2	0	0
10	10	NS	Losses	0	0	0	3	3
10	10	NS	Diff	2	2	2	-3	-3
10	10	NS	Rank	**1**	**1**	**1**	4	4
10	25	NS	Wins	3	2	3	0	0
10	25	NS	Losses	0	1	1	3	3
10	25	NS	Diff	3	1	2	-3	-3
10	25	NS	Rank	**1**	3	2	4	4
10	50	NS	Wins	3	2	3	0	0
10	50	NS	Losses	0	1	1	3	3
10	50	NS	Diff	3	1	2	-3	-3
10	50	NS	Rank	**1**	3	2	4	4
1	10	NS	Wins	2	1	4	2	0
1	10	NS	Losses	2	3	0	1	3
1	10	NS	Diff	0	-2	4	1	-3
1	10	NS	Rank	3	4	**1**	2	5
20	10	NS	Wins	3	3	0	1	1
20	10	NS	Losses	0	0	4	2	2
20	10	NS	Diff	3	3	-4	-1	-1
20	10	NS	Rank	**1**	**1**	5	3	3
all	all	NS	Wins	13	10	12	3	1

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	NS	Losses	2	5	6	12	14
all	all	NS	Diff	11	5	6	-9	-13
all	all	NS	Rank	**1**	3	2	4	5
10	10	all	Wins	2	2	2	0	0
10	10	all	Losses	0	0	0	3	3
10	10	all	Diff	2	2	2	-3	-3
10	10	all	Rank	**1**	**1**	**1**	4	4
10	25	all	Wins	3	2	3	0	0
10	25	all	Losses	0	1	1	3	3
10	25	all	Diff	3	1	2	-3	-3
10	25	all	Rank	**1**	3	2	4	4
10	50	all	Wins	3	2	3	0	0
10	50	all	Losses	0	1	1	3	3
10	50	all	Diff	3	1	2	-3	-3
10	50	all	Rank	**1**	3	2	4	4
1	10	all	Wins	4	2	4	3	1
1	10	all	Losses	5	3	0	2	4
1	10	all	Diff	-1	-1	4	1	-3
1	10	all	Rank	3	3	**1**	2	5
20	10	all	Wins	3	3	8	1	1
20	10	all	Losses	2	2	4	4	4
20	10	all	Diff	1	1	4	-3	-3
20	10	all	Rank	2	2	**1**	4	4
all	all	all	Wins	15	11	20	4	2
all	all	all	Losses	7	7	6	15	17
all	all	all	Diff	8	4	14	-11	-15
all	all	all	Rank	2	3	**1**	4	5

Table 11.25: Wins and Losses of FDA5$_{iso}$

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	*acc*	Wins	1	1	0	1	1
10	10	*acc*	Losses	0	0	4	0	0
10	10	*acc*	Diff	1	1	-4	1	1
10	10	*acc*	Rank	**1**	**1**	5	**1**	**1**
10	25	*acc*	Wins	1	1	0	1	1
10	25	*acc*	Losses	0	0	4	0	0
10	25	*acc*	Diff	1	1	-4	1	1
10	25	*acc*	Rank	**1**	**1**	5	**1**	**1**
20	10	*acc*	Wins	0	0	4	0	0
20	10	*acc*	Losses	1	1	0	1	1
20	10	*acc*	Diff	-1	-1	4	-1	-1
20	10	*acc*	Rank	2	2	**1**	2	2
all	all	*acc*	Wins	2	2	4	2	2
all	all	*acc*	Losses	1	1	8	1	1
all	all	*acc*	Diff	1	1	-4	1	1
all	all	*acc*	Rank	**1**	**1**	5	**1**	**1**
10	10	*stab*	Wins	1	0	0	1	1

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	*stab*	Losses	0	0	3	0	0
10	10	*stab*	Diff	1	0	-3	1	1
10	10	*stab*	Rank	**1**	4	5	**1**	**1**
1	10	*stab*	Wins	0	1	0	1	1
1	10	*stab*	Losses	0	0	3	0	0
1	10	*stab*	Diff	0	1	-3	1	1
1	10	*stab*	Rank	4	**1**	5	**1**	**1**
20	10	*stab*	Wins	0	0	4	0	0
20	10	*stab*	Losses	1	1	0	1	1
20	10	*stab*	Diff	-1	-1	4	-1	-1
20	10	*stab*	Rank	2	2	**1**	2	2
all	all	*stab*	Wins	1	1	4	2	2
all	all	*stab*	Losses	1	1	6	1	1
all	all	*stab*	Diff	0	0	-2	1	1
all	all	*stab*	Rank	3	3	5	**1**	**1**
10	10	NS	Wins	3	3	0	1	1
10	10	NS	Losses	0	0	4	2	2
10	10	NS	Diff	3	3	-4	-1	-1
10	10	NS	Rank	**1**	**1**	5	3	3
10	25	NS	Wins	3	3	0	1	1
10	25	NS	Losses	0	0	4	2	2
10	25	NS	Diff	3	3	-4	-1	-1
10	25	NS	Rank	**1**	**1**	5	3	3
10	50	NS	Wins	3	3	0	1	1
10	50	NS	Losses	0	0	4	2	2
10	50	NS	Diff	3	3	-4	-1	-1
10	50	NS	Rank	**1**	**1**	5	3	3
1	10	NS	Wins	4	1	0	1	1
1	10	NS	Losses	0	1	4	1	1
1	10	NS	Diff	4	0	-4	0	0
1	10	NS	Rank	**1**	2	5	2	2
20	10	NS	Wins	3	3	0	1	1
20	10	NS	Losses	0	0	4	2	2
20	10	NS	Diff	3	3	-4	-1	-1
20	10	NS	Rank	**1**	**1**	5	3	3
all	all	NS	Wins	16	13	0	5	5
all	all	NS	Losses	0	1	20	9	9
all	all	NS	Diff	16	12	-20	-4	-4
all	all	NS	Rank	**1**	2	5	3	3
10	10	all	Wins	5	4	0	3	3
10	10	all	Losses	0	0	11	2	2
10	10	all	Diff	5	4	-11	1	1
10	10	all	Rank	**1**	2	5	3	3
10	25	all	Wins	4	4	0	2	2
10	25	all	Losses	0	0	8	2	2
10	25	all	Diff	4	4	-8	0	0
10	25	all	Rank	**1**	**1**	5	3	3
10	50	all	Wins	3	3	0	1	1
10	50	all	Losses	0	0	4	2	2
10	50	all	Diff	3	3	-4	-1	-1

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	50	all	Rank	**1**	**1**	5	3	3
1	10	all	Wins	4	2	0	2	2
1	10	all	Losses	0	1	7	1	1
1	10	all	Diff	4	1	-7	1	1
1	10	all	Rank	**1**	2	5	2	2
20	10	all	Wins	3	3	8	1	1
20	10	all	Losses	2	2	4	4	4
20	10	all	Diff	1	1	4	-3	-3
20	10	all	Rank	2	2	**1**	4	4
all	all	all	Wins	19	16	8	9	9
all	all	all	Losses	2	3	34	11	11
all	all	all	Diff	17	13	-26	-2	-2
all	all	all	Rank	**1**	2	5	3	3

Table 11.26: Wins and Losses of FDA5$_{dec}$

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	*acc*	Wins	0	0	4	0	0
10	10	*acc*	Losses	1	1	0	1	1
10	10	*acc*	Diff	-1	-1	4	-1	-1
10	10	*acc*	Rank	2	2	**1**	2	2
1	10	*acc*	Wins	0	0	4	0	0
1	10	*acc*	Losses	1	1	0	1	1
1	10	*acc*	Diff	-1	-1	4	-1	-1
1	10	*acc*	Rank	2	2	**1**	2	2
20	10	*acc*	Wins	0	0	4	0	0
20	10	*acc*	Losses	1	1	0	1	1
20	10	*acc*	Diff	-1	-1	4	-1	-1
20	10	*acc*	Rank	2	2	**1**	2	2
all	all	*acc*	Wins	0	0	12	0	0
all	all	*acc*	Losses	3	3	0	3	3
all	all	*acc*	Diff	-3	-3	12	-3	-3
all	all	*acc*	Rank	2	2	**1**	2	2
1	10	*stab*	Wins	0	0	4	0	0
1	10	*stab*	Losses	1	1	0	1	1
1	10	*stab*	Diff	-1	-1	4	-1	-1
1	10	*stab*	Rank	2	2	**1**	2	2
20	10	*stab*	Wins	0	0	4	0	0
20	10	*stab*	Losses	1	1	0	1	1
20	10	*stab*	Diff	-1	-1	4	-1	-1
20	10	*stab*	Rank	2	2	**1**	2	2
all	all	*stab*	Wins	0	0	8	0	0
all	all	*stab*	Losses	2	2	0	2	2
all	all	*stab*	Diff	-2	-2	8	-2	-2
all	all	*stab*	Rank	2	2	**1**	2	2
10	10	*NS*	Wins	3	3	0	1	1
10	10	*NS*	Losses	0	0	4	2	2
10	10	*NS*	Diff	3.0	3.0	-4.0	-1.0	-1.0

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	NS	Rank	**1**	**1**	5	3	3
10	25	NS	Wins	3	3	0	1	1
10	25	NS	Losses	0	0	4	2	2
10	25	NS	Diff	3.0	3.0	-4.0	-1.0	-1.0
10	25	NS	Rank	**1**	**1**	5	3	3
10	50	NS	Wins	3	3	0	1	1
10	50	NS	Losses	0	0	4	2	2
10	50	NS	Diff	3.0	3.0	-4.0	-1.0	-1.0
10	50	NS	Rank	**1**	**1**	5	3	3
1	10	NS	Wins	3	3	0	1	1
1	10	NS	Losses	0	0	4	2	2
1	10	NS	Diff	3.0	3.0	-4.0	-1.0	-1.0
1	10	NS	Rank	**1**	**1**	5	3	3
20	10	NS	Wins	3	3	0	1	1
20	10	NS	Losses	0	0	4	2	2
20	10	NS	Diff	3.0	3.0	-4.0	-1.0	-1.0
20	10	NS	Rank	**1**	**1**	5	3	3
all	all	NS	Wins	15	15	0	5	5
all	all	NS	Losses	0	0	20	10	10
all	all	NS	Diff	15	15	-20	-5	-5
all	all	NS	Rank	**1**	**1**	5	3	3
10	10	all	Wins	3	3	4	1	1
10	10	all	Losses	1	1	4	3	3
10	10	all	Diff	2	2	0	-2	-2
10	10	all	Rank	**1**	**1**	3	4	4
10	25	all	Wins	3	3	0	1	1
10	25	all	Losses	0	0	4	2	2
10	25	all	Diff	3	3	-4	-1	-1
10	25	all	Rank	**1**	**1**	5	3	3
10	50	all	Wins	3	3	0	1	1
10	50	all	Losses	0	0	4	2	2
10	50	all	Diff	3	3	-4	-1	-1
10	50	all	Rank	**1**	**1**	5	3	3
1	10	all	Wins	3	3	8	1	1
1	10	all	Losses	2	2	4	4	4
1	10	all	Diff	1	1	4	-3	-3
1	10	all	Rank	2	2	**1**	4	4
20	10	all	Wins	3	3	8	1	1
20	10	all	Losses	2	2	4	4	4
20	10	all	Diff	1	1	4	-3	-3
20	10	all	Rank	2	2	**1**	4	4
all	all	all	Wins	15	15	20	5	5
all	all	all	Losses	5	5	20	15	15
all	all	all	Diff	10	10	0	-10	-10
all	all	all	Rank	**1**	**1**	3	4	4

The results for the DMOOPs with a discontinous POF, HE1 and HE2, are presented in Tables 11.27 to 11.28. For HE1, DNSGA-II-B obtained the best performance and DVEPSO performed the worst. DMOPSO obtained the second best rank. Both

DVEPSO and dCOEA performed poorly, obtaining more losses than wins. For HE2, the best performance was obtained by DNSGA-II-A and DNSGA-II-B. DVEPSO and dCOEA performed poorly, obtaining more losses than wins, with DVEPSO obtaining the worst rank. From the results it can clearly be seen that DVEPSO struggles to converge towards discontinous POFs.

Table 11.27: Wins and Losses of HE1 obtained by the DMOO algorithms

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	*acc*	Wins	1	2	3	3	0
10	10	*acc*	Losses	3	2	1	0	3
10	10	*acc*	Diff	-2	0	2	3	-3
10	10	*acc*	Rank	4	3	2	**1**	5
10	25	*acc*	Wins	2	3	1	3	0
10	25	*acc*	Losses	2	1	3	0	3
10	25	*acc*	Diff	0	2	-2	3	-3
10	25	*acc*	Rank	3	2	4	**1**	5
10	50	*acc*	Wins	1	2	1	3	0
10	50	*acc*	Losses	1	1	2	0	3
10	50	*acc*	Diff	0	1	-1	3	-3
10	50	*acc*	Rank	3	2	4	**1**	5
1	10	*acc*	Wins	2	3	1	3	0
1	10	*acc*	Losses	2	1	3	0	3
1	10	*acc*	Diff	0	2	-2	3	-3
1	10	*acc*	Rank	3	2	4	**1**	5
20	10	*acc*	Wins	2	3	1	3	0
20	10	*acc*	Losses	2	1	3	0	3
20	10	*acc*	Diff	0	2	-2	3	-3
20	10	*acc*	Rank	3	2	4	**1**	5
all	all	*acc*	Wins	8	13	7	15	0
all	all	*acc*	Losses	10	6	12	0	15
all	all	*acc*	Diff	-2	7	-5	15	-15
all	all	*acc*	Rank	3	2	4	**1**	5
10	10	*stab*	Wins	0	2	3	3	1
10	10	*stab*	Losses	4	2	1	0	2
10	10	*stab*	Diff	-4	0	2	3	-1
10	10	*stab*	Rank	5	3	2	**1**	4
10	25	*stab*	Wins	2	3	0	3	1
10	25	*stab*	Losses	2	1	4	0	2
10	25	*stab*	Diff	0	2	-4	3	-1
10	25	*stab*	Rank	3	2	5	**1**	4
10	50	*stab*	Wins	0	1	0	3	2
10	50	*stab*	Losses	2	1	2	0	1
10	50	*stab*	Diff	-2	0	-2	3	1
10	50	*stab*	Rank	4	3	4	**1**	2
1	10	*stab*	Wins	2	3	0	3	1
1	10	*stab*	Losses	2	1	4	0	2
1	10	*stab*	Diff	0	2	-4	3	-1

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
1	10	*stab*	Rank	3	2	5	**1**	4
20	10	*stab*	Wins	2	3	0	3	1
20	10	*stab*	Losses	2	1	4	0	2
20	10	*stab*	Diff	0	2	-4	3	-1
20	10	*stab*	Rank	3	2	5	**1**	4
all	all	*stab*	Wins	6	12	3	15	6
all	all	*stab*	Losses	12	6	15	0	9
all	all	*stab*	Diff	-6	6	-12	15	-3
all	all	*stab*	Rank	4	2	5	**1**	3
10	10	*NS*	Wins	3	3	1	1	0
10	10	*NS*	Losses	0	0	3	2	3
10	10	*NS*	Diff	3	3	-2	-1	-3
10	10	*NS*	Rank	**1**	**1**	4	3	5
10	25	*NS*	Wins	3	3	1	1	0
10	25	*NS*	Losses	0	0	3	2	3
10	25	*NS*	Diff	3	3	-2	-1	-3
10	25	*NS*	Rank	**1**	**1**	4	3	5
10	50	*NS*	Wins	3	3	1	1	0
10	50	*NS*	Losses	0	0	3	2	3
10	50	*NS*	Diff	3	3	-2	-1	-3
10	50	*NS*	Rank	**1**	**1**	4	3	5
1	10	*NS*	Wins	3	3	1	1	0
1	10	*NS*	Losses	0	0	3	2	3
1	10	*NS*	Diff	3	3	-2	-1	-3
1	10	*NS*	Rank	**1**	**1**	4	3	5
20	10	*NS*	Wins	3	3	1	1	0
20	10	*NS*	Losses	0	0	3	2	3
20	10	*NS*	Diff	3	3	-2	-1	-3
20	10	*NS*	Rank	**1**	**1**	4	3	5
all	all	*NS*	Wins	15	15	5	5	0
all	all	*NS*	Losses	0	0	15	10	15
all	all	*NS*	Diff	15	15	-10	-5	-15
all	all	*NS*	Rank	**1**	**1**	4	3	5
10	10	all	Wins	4	7	7	7	1
10	10	all	Losses	7	4	5	2	8
10	10	all	Diff	-3	3	2	5	-7
10	10	all	Rank	4	2	3	**1**	5
10	25	all	Wins	7	9	2	7	1
10	25	all	Losses	4	2	10	2	8
10	25	all	Diff	3	7	-8	5	-7
10	25	all	Rank	3	**1**	5	2	4
10	50	all	Wins	4	6	2	7	2
10	50	all	Losses	3	2	7	2	7
10	50	all	Diff	1	4	-5	5	-5
10	50	all	Rank	3	2	4	**1**	4
1	10	all	Wins	7	9	2	7	1
1	10	all	Losses	4	2	10	2	8
1	10	all	Diff	3	7	-8	5	-7
1	10	all	Rank	3	**1**	5	2	4
20	10	all	Wins	7	9	2	7	1

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
20	10	all	Losses	4	2	10	2	8
20	10	all	Diff	3	7	-8	5	-7
20	10	all	Rank	3	**1**	5	2	4
all	all	all	Wins	29	40	15	35	6
all	all	all	Losses	22	12	42	10	39
all	all	all	Diff	7	28	-27	25	-33
all	all	all	Rank	3	**1**	4	2	5

Table 11.28: Wins and Losses of HE2 obtained by the DMOO algorithms

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
10	10	*acc*	Wins	2	2	1	3	0
10	10	*acc*	Losses	1	1	3	0	3
10	10	*acc*	Diff	1	1	-2	3	-3
10	10	*acc*	Rank	2	2	4	**1**	5
10	25	*acc*	Wins	1	1	1	1	0
10	25	*acc*	Losses	0	0	3	0	1
10	25	*acc*	Diff	1	1	-2	1	-1
10	25	*acc*	Rank	**1**	**1**	5	**1**	4
10	50	*acc*	Wins	2	2	1	3	0
10	50	*acc*	Losses	1	1	3	0	3
10	50	*acc*	Diff	1	1	-2	3	-3
10	50	*acc*	Rank	2	2	4	**1**	5
1	10	*acc*	Wins	2	2	1	3	0
1	10	*acc*	Losses	1	1	3	0	3
1	10	*acc*	Diff	1	1	-2	3	-3
1	10	*acc*	Rank	2	2	4	**1**	5
20	10	*acc*	Wins	1	1	1	1	0
20	10	*acc*	Losses	0	0	3	0	1
20	10	*acc*	Diff	1	1	-2	1	-1
20	10	*acc*	Rank	**1**	**1**	5	**1**	4
all	all	*acc*	Wins	8	8	5	11	0
all	all	*acc*	Losses	3	3	15	0	11
all	all	*acc*	Diff	5	5	-10	11	-11
all	all	*acc*	Rank	2	2	4	**1**	5
10	10	*stab*	Wins	1	1	1	3	0
10	10	*stab*	Losses	1	1	1	0	3
10	10	*stab*	Diff	0	0	0	3	-3
10	10	*stab*	Rank	2	2	2	**1**	5
10	25	*stab*	Wins	1	1	1	1	0
10	25	*stab*	Losses	0	0	3	0	1
10	25	*stab*	Diff	1	1	-2	1	-1
10	25	*stab*	Rank	**1**	**1**	5	**1**	4
10	50	*stab*	Wins	1	1	1	1	0
10	50	*stab*	Losses	0	0	3	0	1
10	50	*stab*	Diff	1	1	-2	1	-1
10	50	*stab*	Rank	**1**	**1**	5	**1**	4
1	10	*stab*	Wins	2	2	1	3	0

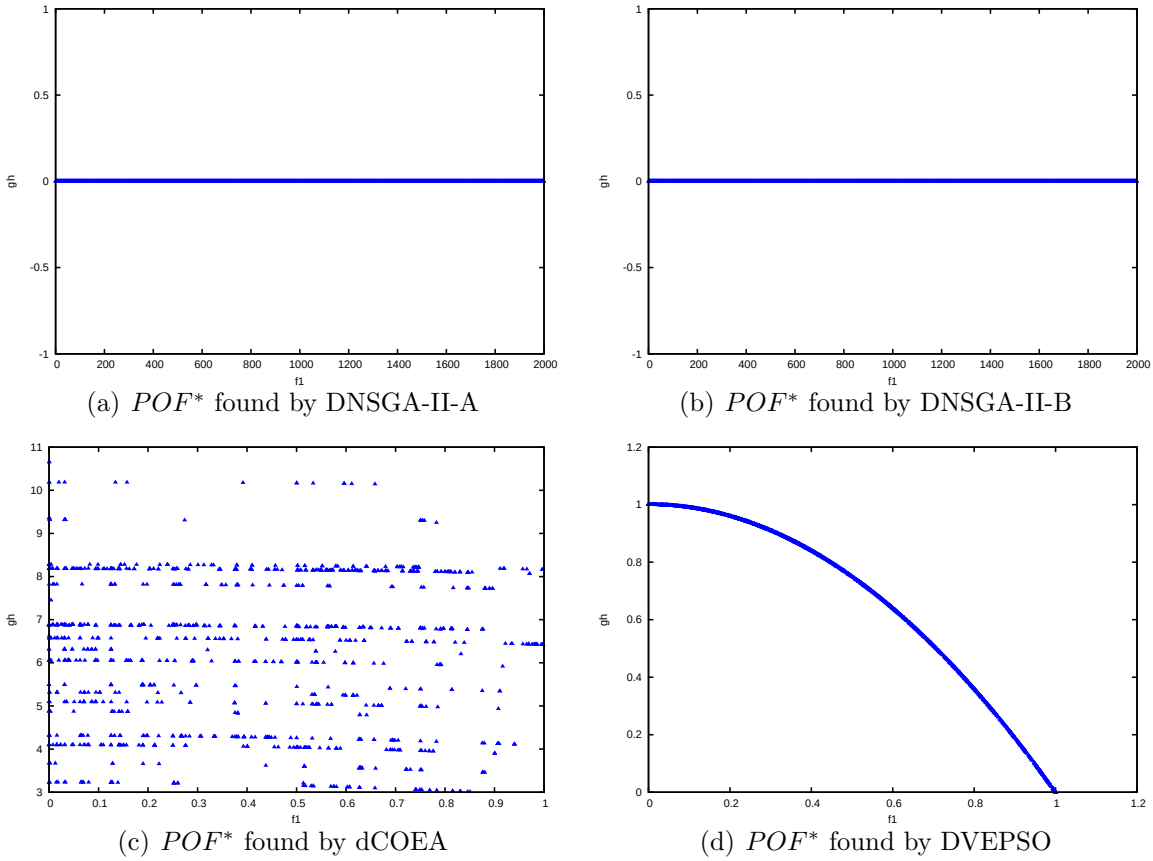
Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
1	10	*stab*	Losses	1	1	3	0	3
1	10	*stab*	Diff	1	1	-2	3	-3
1	10	*stab*	Rank	2	2	4	**1**	5
20	10	*stab*	Wins	2	2	0	3	1
20	10	*stab*	Losses	1	1	4	0	2
20	10	*stab*	Diff	1	1	-4	3	-1
20	10	*stab*	Rank	2	2	5	**1**	4
all	all	*stab*	Wins	7	7	4	11	1
all	all	*stab*	Losses	3	3	14	0	10
all	all	*stab*	Diff	4	4	-10	11	-9
all	all	*stab*	Rank	2	2	5	**1**	4
10	10	*NS*	Wins	3	3	1	1	0
10	10	*NS*	Losses	0	0	3	2	3
10	10	*NS*	Diff	3	3	-2	-1	-3
10	10	*NS*	Rank	**1**	**1**	4	3	5
10	25	*NS*	Wins	3	3	1	1	0
10	25	*NS*	Losses	0	0	3	2	3
10	25	*NS*	Diff	3	3	-2	-1	-3
10	25	*NS*	Rank	**1**	**1**	4	3	5
10	50	*NS*	Wins	3	3	1	1	0
10	50	*NS*	Losses	0	0	3	2	3
10	50	*NS*	Diff	3	3	-2	-1	-3
10	50	*NS*	Rank	**1**	**1**	4	3	5
1	10	*NS*	Wins	3	3	1	1	0
1	10	*NS*	Losses	0	0	3	2	3
1	10	*NS*	Diff	3	3	-2	-1	-3
1	10	*NS*	Rank	**1**	**1**	4	3	5
20	10	*NS*	Wins	3	3	1	1	0
20	10	*NS*	Losses	0	0	3	2	3
20	10	*NS*	Diff	3	3	-2	-1	-3
20	10	*NS*	Rank	**1**	**1**	4	3	5
all	all	*NS*	Wins	15	15	5	5	0
all	all	*NS*	Losses	0	0	15	10	15
all	all	*NS*	Diff	15	15	-10	-5	-15
all	all	*NS*	Rank	**1**	**1**	4	3	5
10	10	all	Wins	6	6	3	7	0
10	10	all	Losses	2	2	7	2	9
10	10	all	Diff	4	4	-4	5	-9
10	10	all	Rank	2	2	4	**1**	5
10	25	all	Wins	5	5	3	3	0
10	25	all	Losses	0	0	9	2	5
10	25	all	Diff	5	5	-6	1	-5
10	25	all	Rank	**1**	**1**	5	3	4
10	50	all	Wins	6	6	3	5	0
10	50	all	Losses	1	1	9	2	7
10	50	all	Diff	5	5	-6	3	-7
10	50	all	Rank	**1**	**1**	4	3	5
1	10	all	Wins	7	7	3	7	0
1	10	all	Losses	2	2	9	2	9
1	10	all	Diff	5	5	-6	5	-9

Continued on next page

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
1	10	all	Rank	**1**	**1**	4	**1**	5
20	10	all	Wins	6	6	2	5	1
20	10	all	Losses	1	1	10	2	6
20	10	all	Diff	5	5	-8	3	-5
20	10	all	Rank	**1**	**1**	5	3	4
all	all	all	Wins	30	30	14	27	1
all	all	all	Losses	6	6	44	10	36
all	all	all	Diff	24	24	-30	17	-35
all	all	all	Rank	**1**	**1**	4	3	5

The POF^* found by the DMOAs for DIMP2, dMOP3, FDA5, FDA5$_{iso}$, FDA5$_{dec}$, and HE2 are illustrated in Figures 11.1 to 11.5.



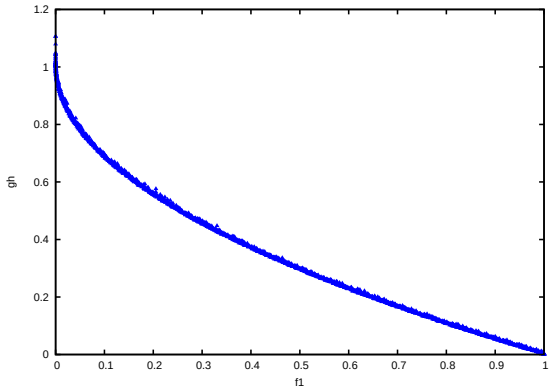
(a) POF^* found by DNSGA-II-A

(b) POF^* found by DNSGA-II-B

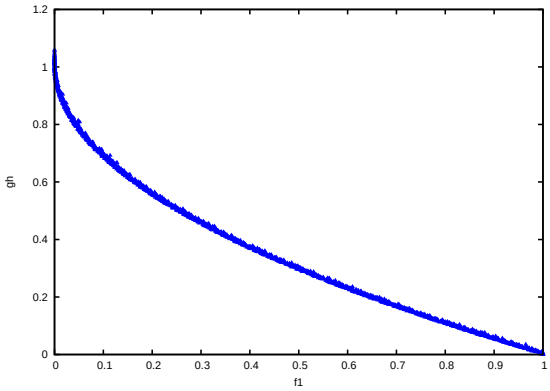
(c) POF^* found by dCOEA

(d) POF^* found by DVEPSO

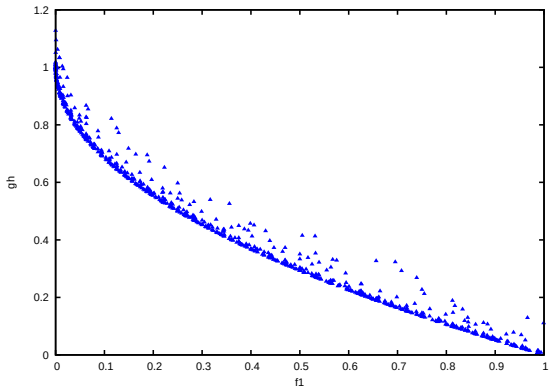
Figure 11.1: POF^* for DIMP2 for $n_t = 10$ and $\tau_t = 50$ found by the various DMOAs. DMOPSO did not find any solutions.



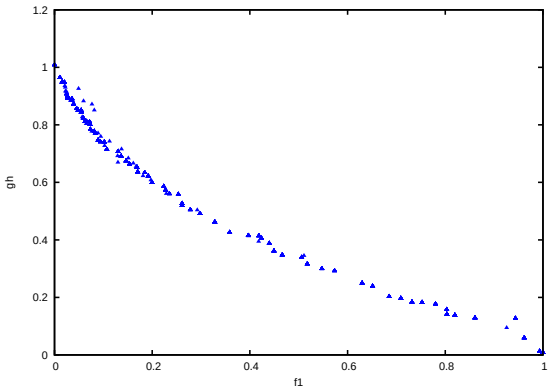
(a) POF^* found by DNSGA-II-A



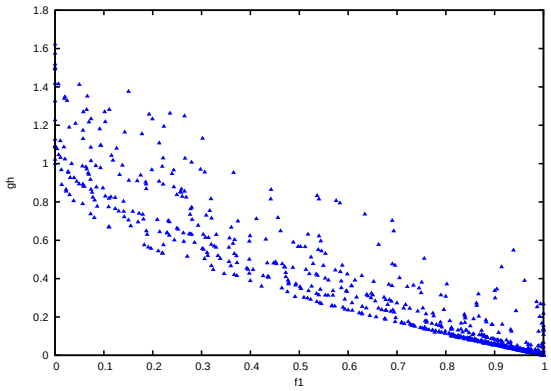
(b) POF^* found by DNSGA-II-B



(c) POF^* found by dCOEA

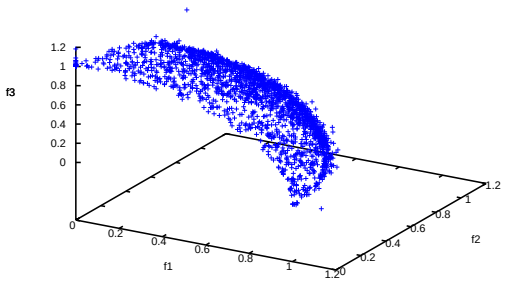


(d) POF^* found by DMOPSO

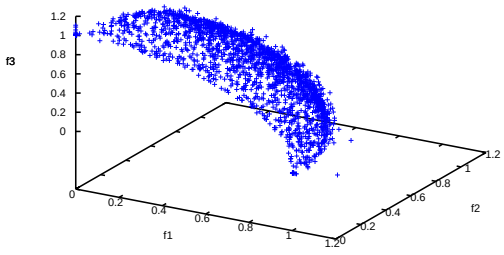


(e) POF^* found by DVEPSO

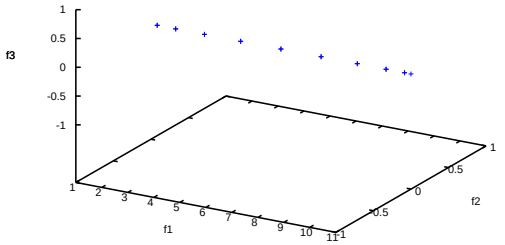
Figure 11.2: POF^* for dMOP3 for $n_t = 10$ and $\tau_t = 50$ found by the various DMOAs



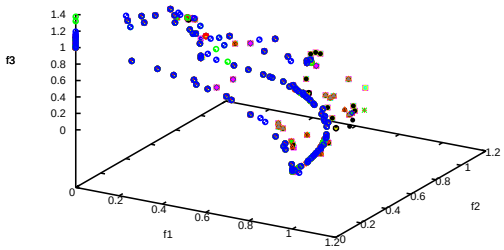
(a) POF^* found by DNSGA-II-A



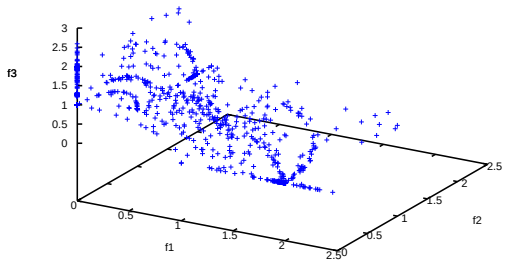
(b) POF^* found by DNSGA-II-B



(c) POF^* found by dCOEA

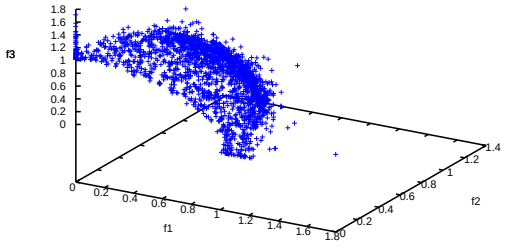


(d) POF^* found by DMOPSO

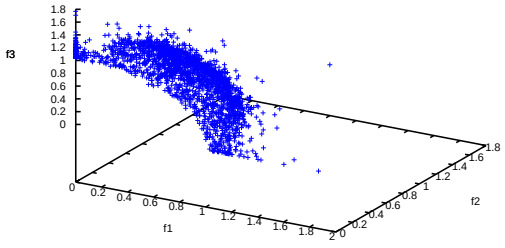


(e) POF^* found by DVEPSO

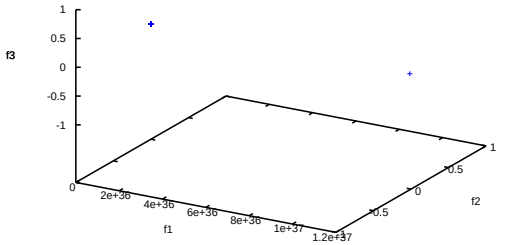
Figure 11.3: POF^* for FDA5$_{iso}$ for $n_t = 10$ and $\tau_t = 50$ found by the various DMOAs



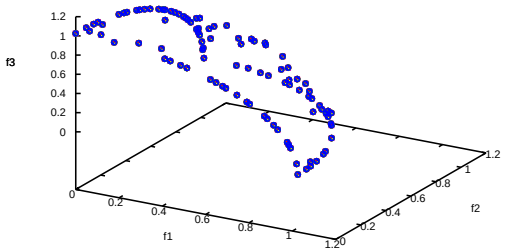
(a) POF^* found by DNSGA-II-A



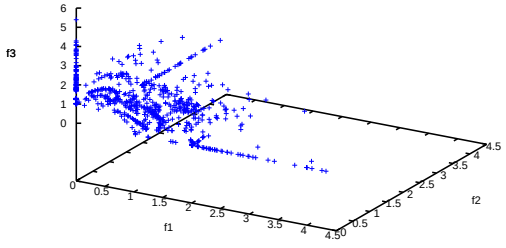
(b) POF^* found by DNSGA-II-B



(c) POF^* found by dCOEA

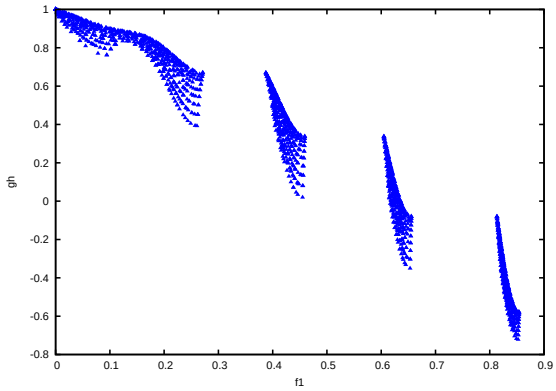


(d) POF^* found by DMOPSO

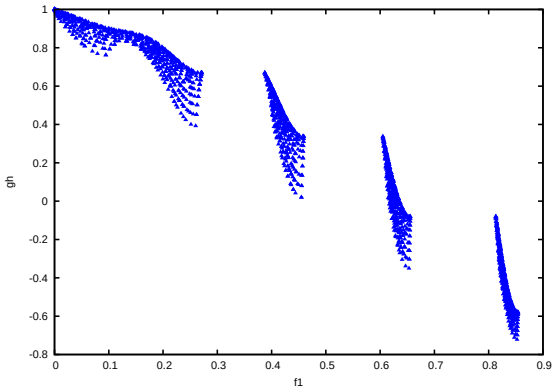


(e) POF^* found by DVEPSO

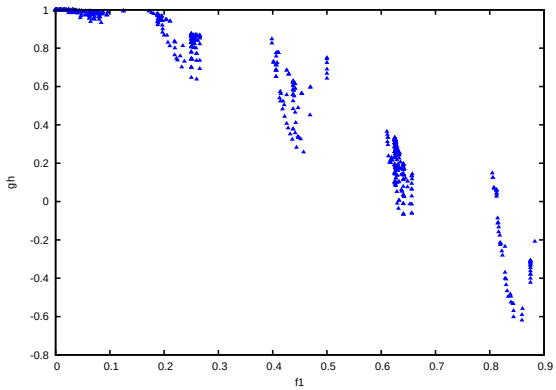
Figure 11.4: POF^* for FDA5$_{dec}$ for $n_t = 10$ and $\tau_t = 50$ found by the various DMOAs



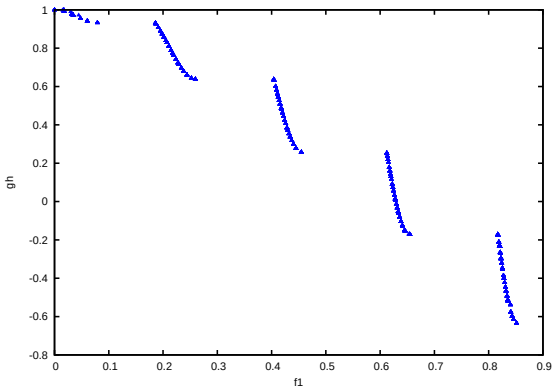
(a) POF^* found by DNSGA-II-A



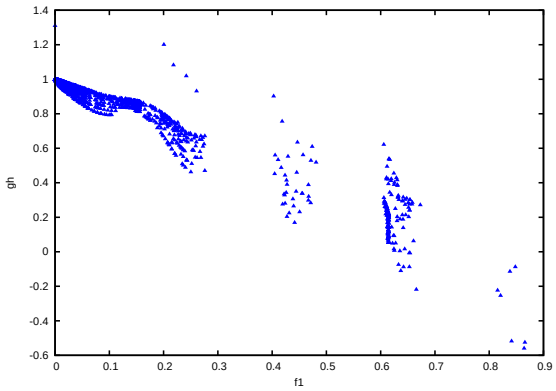
(b) POF^* found by DNSGA-II-B



(c) POF^* found by dCOEA



(d) POF^* found by DMOPSO



(e) POF^* found by DVEPSO

Figure 11.5: POF^* for HE2 for $n_t = 10$ and $\tau_t = 50$ found by the various DMOAs

For DIMP2, DVEPSO outperformed the other DMOAs and found a POF^* close to the true POF (POF). All other DMOAs struggled to converge towards POF. The DNSGA-II algorithms converged really well towards POF of dMOP3. DMOPSO also converged well, but found a worse spread of solutions than the DNSGA-II algorithms. dCOEA performed well, but also found a few solutions that were further away from POF. DVEPSO found a number of solutions further away from POF, but did manage to find solutions close to POF. For FDA5$_{iso}$, the DNSGA-II algorithms found a diverse set of solutions. DVEPSO found less solutions than the DNSGA-II algorithms, but a better spread of solutions than MOPSO. However, dCOEA struggled to converge towards POF of FDA5$_{iso}$. A similar trend than FDA5$_{iso}$ was observed for FDA5$_{dec}$. For HE2, all algorithms converged towards POF. However, DVEPSO found a few solutions further away from POF.

11.3 Summary

This chapter investigated the performance of five DMOAs solving 2-objective and 3-objective DMOOPs of Types I to III, with various frequencies and severities of change. Three performance measures were used to quantify the algorithms' performance, namely accuracy that measures the difference in HV values of the approximated and true POF (acc), stability that measures the effect of the environmental change on the accuracy of the algorithm ($stab$) and the number of non-dominated solutions found by the algorithm (NS).

The five DMOAs investigated were: an NSGA-II algorithm adapted for DMOO where if a change in the environment is detected, a percentage of individuals are randomly selected and replaced with newly created individuals (DNSGA-II-A); an NSGA-II algorithm that selects a percentage of individuals randomly and replaces them with individuals that are mutated from existing individuals when a change is detected (DNSGA-II-B); a dynamic competitive-cooperative coevolutionary algorithm (dCOEA); a MOPSO algorithm adapted for DMOO (DMOPSO); and the multi-swarm PSO-based algorithm proposed in this thesis (DVEPSO).

With regards to the various DMOOP types, DVEPSO obtained the second lowest

rank for Type I DMOOPs, obtaining the best rank for *acc*, but the lowest rank for *stab* and NS. DVEPSO struggled to converge to the POF of dMOP3 where the variable that controls the spread of solutions randomly changes over time. However, DVEPSO was the only algorithm that successfully converged towards the POF of DIMP2, where the decision variables change at different rates over time. DVEPSO obtained the overall best rank for Type II DMOOPs, and obtained the best rank for both *acc* and *stab*, and the third best rank for NS. For Type III DMOOPs, DVEPSO obtained the fourth, second and fourth rank for *acc*, *stab* and NS respectively. DVEPSO struggled to converge towards discontinuous POFs and therefore did not perform as well with regards to the Type III DMOOPs.

Measuring the algorithms' performance over all performance measures, DNSGA-II-B obtained the best overall performance, with DNSGA-II-A obtaining the second best rank and DVEPSO the third best rank. dCOEA obtained the overall worst rank. However, all DMOEAs obtained a high number of wins for NS. Measuring the algorithms' performance over *acc* and *stab* and therefore not taking NS into account lead to DMOPSO obtaining the best performance, DVEPSO obtaining the second best rank and DNSGA-II-B being awarded the third best rank. Once again, dCOEA performed the worst. The DMOEAs obtained the most solutions in general. However, with regards to *acc* and *stab*, the PSO-based MOAs completely outperformed the MOEAs.

In addition, at least one DMOEA outperformed the PSO-based DMOAs on DMOOPs with either an isolated or deceptive POF. Furthermore, most of the DMOAs outperformed DVEPSO on DMOOPs with a discontinuous POF.

The next chapter concludes the research presented in this thesis and proposes possible future work.

Part III

Dynamic Vector Evaluated Particle Swarm Optimisation

Chapter 9

Introduction to Dynamic Vector Evaluated Particle Swarm Optimisation Algorithm

"Goals allow you to control the direction of change in your favor."
–Brian Tracy

This chapter discusses the VEPSO algorithm that has been adapted to solve DMOOPs. The adapted VEPSO algorithm, dynamic VEPSO (DVEPSO), is discussed in Section 9.1. Section 9.2 discusses the tasks of the DVEPSO algorithm that are performed at the top-algorithm level, while Section 9.3 discusses the tasks of the sub-swarms that are performed at the lower-algorithm level. Experiments that were conducted to investigate the influence of various guide update approaches on the performance of DVEPSO are discussed in Section 9.4. Information is provided with regards to the benchmark functions, performance measures and the default configuration of the DVEPSO algorithm used for the experiments, as well as the statistical analysis that was conducted on the obtained data. Furthermore, the obtained results are analysed and discussed. A summary of this chapter is provided in Section 9.5.

9.1 Dynamic Vector Evaluated Particle Swarm Optimisation Algorithm

This section discusses the changes made to the SMOO VEPSO algorithm discussed in Section 7.2 in order to solve DMOOPs. The adapted algorithm, DVEPSO, is presented in Algorithm 9.

Algorithm 9 DVEPSO for DMOO

```
1.  for number of iterations do
2.      check whether a change has occurred
3.      if change has occurred
4.          respond to change
5.          remove dominated solutions from archive
6.      perform PSO iteration
7.      if new solutions are non-dominated
8.          if space in archive
9.              add new solutions to archive
10.     else
11.         remove solutions from archive
12.         add new solutions to archive
13.     select sentry particles
```

Similar to VEPSO, the DVEPSO algorithm consists of two layers, namely a top layer that manages the sub-swarms and a lower layer that contains the sub-swarms. This is illustrated in Figure 9.1.

In order to track a changing POF an algorithm must be able to detect that a change in the environment has occurred and then respond to the change appropriately. Therefore, when solving DMOOPs, the sub-swarms in the lower layer check whether the environment has changed, in addition to optimising the assigned objective function. When VEPSO is used to solve static MOOPs, sharing of knowledge between the sub-swarms and the management of the archive (as discussed in Section 7.2) are managed at the top level. However, the top layer of DVEPSO also manages the way in which the sub-swarms

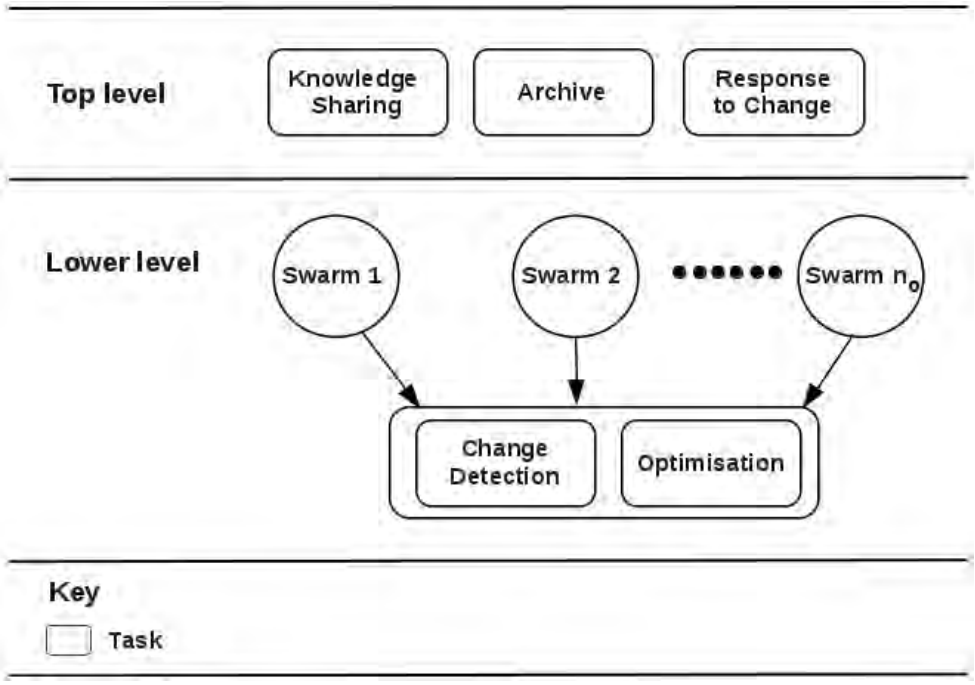


Figure 9.1: The two layers of the DVEPSO algorithm

respond to a change once the change has been detected.

9.2 Top-level Tasks

This section discusses a task that is performed at the top level of the DVEPSO algorithm, namely responding to a change in the environment. This task is performed in addition to the top-level tasks performed by VEPSO (refer to Section 7.2).

If a change has been detected by one or more of the sub-swarms, DVEPSO has to respond to the change to ensure tracking of the changing POF. When a change has been detected, one of the following responses are used:

- re-evaluate all particles in the sub-swarm, or
- re-initialise a percentage of the particles in the sub-swarm.

Re-evaluating the particles ensures that all previously obtained information is preserved. However, the particles already converged towards the POF, and therefore the diversity

of the swarm has to be increased to increase exploration of a new environment. If re-evaluation is used, additional ways should be used to increase the swarm's diversity. However, re-initialisation introduces diversity by re-initialising a certain percentage of the swarm's particles. Re-initialisation preserves previously obtained information from the particles that are not re-initialised. However, it may occur that particles with optimal positions in the new environment are re-initialised and thereby the information is lost.

Greeff and Engelbrecht [72] proposed that the above listed responses can be applied to either all sub-swarms, or to only the sub-swarm(s) whose objective function has changed. Applying the response to all sub-swarms increases the diversity of all sub-swarms and thereby increases the exploration of the sub-swarms. If a sub-swarm's objective function did not change and re-initialisation is used, a percentage of previously obtained information is removed. However, the increasing diversity may lead to exploration of the search space that was not explored before.

After one of the above responses was applied, the following re-evaluations or updates are performed:

- The pbest of each particle is reset to the particle's current position. This ensures that the particle is not biased towards the previous optima. If the new optima is far away from the previous optima and the particle is biased towards the previous optima, it may become stuck at the previous optima or a local optima without finding the new optima.
- Once the particles' pbests are reset, a new gbest is determined. This ensures that the gbest does not attract the other particles towards a previous optimum that is not optimal anymore.

Furthermore, if a change in the environment occurs, the following approaches are proposed to manage the archive [78]:

- remove all solutions from the archive (referred to as ac), or
- re-evaluate the solutions in the archive against the current DMOOP. Then, all solutions that were previously non-dominated but became dominated after the change in the environment occurred, are:
  - removed from the archive (referred to as a_{re}). This approach does not use previously obtained knowledge in the new environment. When an environ-

ment change is severe, previously found solutions that are still non-dominated in the new environment, may cause new non-dominated solutions that are in close proximity of the previous solutions to be removed from the archive, even if they are more optimal than the previously found solution. This may occur when selecting which solutions to remove from a full archive are based on removing solutions from crowded regions of the approximated POF.

  - hill-climbing is applied to a dominated solution in an attempt to change these solutions back to non-dominated solutions. If hill-climbing is unsuccessful, the dominated solution is removed from the archive. However, if hill-climbing is successful, the dominated solution is removed from the archive and the new solution obtained through hill-climbing is added to the archive. This approach is referred to as a_{reh}. This approach re-uses previously obtained knowledge in the new environment and will only be useful if the environmental change is not severe.

- when a change in the environment occurs, a number of particles whose positions represent non-dominated solutions are randomly selected. The average change that the selected particles experience in each objective (or dimension), c_{avg_k}, is calculated. Then, if a selected particle's objective value differs by a threshold β_k (e.g. $\beta_k = c_{avg_k}/2.0$), the solutions in the archive that are within a specied radius c_r (e.g. distance to closest selected particle/2.0) from the selected particle, are deleted. This approach is referred to as a_r. If a_r is used to manage the archive, then before a_r is executed, either a_{re} or a_{reh} is performed. If a_{re} was fist performed, this approach is referred to as a_{ra}. Otherwise, if a_{reh} was first performed, this approach is referred to as a_{rah}. Applying a_r to the archive removes solutions from a certain region of the archive (that falls within the radius c_r of a selected particle) if the environment changes drastically for the decision variable values that produced the solutions of the specific region. This ensures that newly found solutions are added to the archive when the environment changes drastically, increasing the diversity of the archive.

9.3 Low-level Tasks

This section discusses the tasks of change detection and guide updates that are performed at the lower-level of DVEPSO by the sub-swarms. These tasks are performed in addition to the other low-level tasks performed by VEPSO (refer to Section 7.2).

9.3.1 Change Detection

In order to solve DMOOPs, DVEPSO must be able to detect a change that occurred in the environment. Change detection is done using sentry particles [22], where a specified number of particles are randomly selected and re-evaluated after the algorithm performed the specific iteration, but before the next iteration starts. If the sentry particle's fitness value differs after re-evaluation with more than a specified value, the swarm is notified that a change in the environment has occurred. If a change in the environment of a sub-swarm has occurred, the sub-swarm alerts the top-level of DVEPSO. The top-level then informs the sub-swarms which response to execute.

9.3.2 Guide Update Approaches

Similar to VEPSO, the search process of DVEPSO is driven through the local and global guides. VEPSO uses no Pareto-dominance information for the guide updates. However, for DVEPSO, guide update approaches that use Pareto-dominance information and therefore do dominance checking are also investigated. The following guide update approaches are proposed for DVEPSO [71]:

- The standard VEPSO guide update, where the particle's fitness is measured with regards to only the objective function that the specific swarm optimises. Only if an improvement in the fitness of the current guide can be obtained, is the guide updated. No Pareto-dominance information is used. With reference to a local guide, this approach is referred to as p_s and with reference to a global guide, g_s.
- The dominant approach, where each particle's fitness is measured with respect to all objectives of the DMOOP. If the particle's position dominates the current local guide, the particle's current position is selected as the new local guide. This

strategy is referred to as p_d. If this approach is used to update a global guide, it is referred to as g_d.

- The non-dominated approach, where a guide is updated if the new position is non-dominated with respect to the guide. When used as a local guide update, it is referred to as p_n, and g_n if used as a global guide update.
- The random approach, where a guide is updated if the new position is non-dominated with respect to the guide, by randomly selecting either the particle position or the corresponding guide. When used as a local guide update, it is referred to as p_r and g_r if used as a global guide update.

The effectiveness of these approaches when used by DVEPSO is unknown. Therefore, experiments were conducted to investigate the influence of these guide update approaches on the performance of DVEPSO. The next section discusses the experiments and the results that were obtained from the experiments.

9.4 Effectiveness of Guide Update Approaches

Various guide update approaches exist as discussed in Section 9.3. This section describes experiments that were conducted to investigate the influence of the various guide update approaches on the performance of DVEPSO. It should be noted that this section focuses on guide update approaches, and not on guide selection approaches. Guide selection approaches focus on the selection of solutions from the archive to guide the optimisation process to ensure a diverse set of solutions. The guides that are selected from the archive are then used as the local (personal best) and global guides (global best) of the PSO algorithm. The guide update approaches discussed in this section focus on methods that are used to update the swarm's local (personal best) and global (global best) guides using the solutions found by the particles.

Section 9.4.1 discusses the experimental setup and the benchmark functions and performance measures that were used to evaluate the performance of the various guide update approaches. The DVEPSO configuration used for the experiments, as well as the statistical analysis process that was performed on the obtained data, are also discussed. The results obtained from the experiments are discussed in Section 9.4.2.

9.4.1 Experimental Setup

All combinations of the local and global guide updates discussed in Section 9.3 were used in the experiments.

All experiments consisted of 30 independent runs and each run continued for 1000 iterations. For all benchmark functions, the severity of change (n_t) was set to 1, 10 and 20 and the frequency of change (τ_t) was set to either 10, 25 or 50. This selection of n_t and τ_t values enables the evaluation of DVEPSO in both a fast and slowly changing environment, and an evironment that changes either gradually or severely over time.

The PSO parameters were set to values that lead to convergent behaviour [63], namely $w = 0.72$ and $c_1 = c_2 = 1.49$. Convergent behaviour ensures that the particles converge towards the current POF. After a change in the environment, diversity is introduced into the swarm to ensure more exploration to find the new POF.

All code was implemented in the Computational Intelligence library (CIlib) [122]. All simulations were run on the Sun Hybrid System's Harpertown and Nehalem Systems of the Center for High Performance Computing [24]. The SUN Nehalem system has an Intel Nehalem processor of 2.93 GHz, 2304 CPU cores, 3465 Gb of Memory and produces 24 TFlops at peak performance [24]. The SUN Harpertown system has an Intel Xeon processor of 3.0 GHz, 384 CPU cores, 768 Gb of Memory and produces 3 TFlops at peak performance [24].

Benchmark Functions

Based on the analysis of DMOOPs in Chapter 3, fifteen benchmark functions were selected of various DMOOP Types to study the influence of guide update approaches on the performance of DVEPSO, namely a modified version of DIMP2 with a concave POF (referred to as DIMP2 in the rest of the thesis), FDA1$_{Zhou}$, FDA2, FDA2$_{Camara}$, FDA3 [58], FDA3$_{Camara}$, dMOP2, dMOP3, dMOP2$_{iso}$, dMOP2$_{dec}$, HE1, HE2, HE6, HE7 and HE9.

DIMP2 is a Type I problem where each decision variable has its own rate of change, except the variable x_1 that controls the spread of solutions. FDA1$_{Zhou}$ has non-linear dependencies between the decision variables and is a Type II problem. FDA2 and dMOP2 are Type II DMOOPs with a POF that changes from convex to concave. FDA2$_{Camara}$

also has a POF that changes from convex to concave over time, but is a Type III DMOOP. FDA3 and FDA3$_{Camara}$ are Type II DMOOPs with a convex POF where the density of solutions in the POF changes over time. dMOP3 is a Type I DMOOP with a convex POF where the spread of the POF solutions changes over time. HE1 and HE2 are both a Type III DMOOP with a discontinuous POF that consists of various disconnected continuous sub-regions. HE6, HE7 and HE9 are Type III DMOOPs where each decision variable has a different POS and the POSs are non-linear functions. dMOP2$_{iso}$ and dMOP2$_{dec}$ are similar to dMOP2, but with an isolated and deceptive POF respectively.

Even though the DMOOPs FDA2 and FDA3 are problematic (refer to Section 3.2.1), they were selected for the experiments to determine whether DVEPSO can still track the changing POF in spite of the issues with these DMOO functions.

Performance Measures

Chapter 4 discussed the analysis of DMOO performance measures. Based on this analysis, three performance measures were selected for this study, to determine the performance of DVEPSO for the different guide update approaches.

The first performance measure is the number of non-dominated solutions (NS) in the found POF. Even though this measure does not provide any information with regards to the quality of the solutions, it provides additional information when comparing the performance of various algorithms.

The second performance measure is the acc_{alt} measure (see Equation (4.25)), referred to in this chapter as *acc*. A low *acc* value indicates a good performance. The calculation of *acc* requires sampled solutions of the true POF, POF'. For these experiments, POF' solutions were created for each DMOOP by dividing the range of each variable into one thousand equally sized intervals. For each combination of decision variable values the objective function values were calculated using the equation of the true POF, POF, for the specific DMOOP. This process was followed for each n_t-τ_t combination. The HV was calculated according to [7], using the source code available at [61].

The effect of the changes in the environment on *acc* of the algorithm is quantified by the third measure, namely *stab* (refer to Equation (4.21)), where a low *stab* value indicates good performance.

Algorithm Configuration

The following default configuration of DVEPSO was used for the experiments:

- Each swarm has 20 particles and a random swarm topology is used.
- The non-dominated solutions found so far is stored in an archive with size set to 100. If the archive is full, a solution from a crowded region in the found POF is removed. The croweded region is determined by calculating the distance between each solution in the archive and its nearest solution in the archive, and selecting the solution(s) with the smallest distance value.
- Sentry particles is used for change detection (refer to lines 2 and 13 in Algorithm 9). If a change has been detected, 30% of the particles of the swarm(s) whose objective function changed is re-initialised (refer to line 4 in Algorithm 9). The non-dominated solutions in the archive are re-evaluated and the solutions that have become dominated are removed from the archive (refer to line 5 in Algorithm 9). Each particle's pbest is set to its current position and a new gbest is determined.

Statistical Analysis

This section discusses the statistical analysis procedure performed on the obtained data. For each function and for each n_t-τ_t combination, a Kruskal-Wallis test was performed over the obtained data to determine whether there is a statistical significant difference in performance. For each performance measure the obtained data is the mean of the performance measure values for each iteration just before a change occurred in the environment over 30 runs. If this test indicated that there was a difference, pairwise Mann-Whitney U tests were performed between the pairs of obtained data for all the guide update approaches.

For each pair of guide update approaches, if the pairwise Mann-Whitney U test indicated a statistically significant difference, a win was recorded for the winning algorithm and a loss for the losing algorithm.

All statistical tests were performed for a confidence level of 95%. The null hypothesis was that there is no statistical significant difference between the performance of the various guide update approaches. The alternative hypothesis was that there is a difference

in mean performance.

9.4.2 Results

This section presents the results obtained by the various guide update approaches. The results are discussed considering the various n_t-τ_t combinations, with regards to three performance measures and with regards to DMOOP Types I to III. General observations are also highlighted. Tables 9.1 to 9.13 present the wins and losses. Only the tables highlighting interesting trends are discussed and therefore presented in this section. The other wins and losses tables are presented in Appendix D. Only statistical significant values are included in the tables. The p-values obtained for the various Mann-Whitney U tests, as well as the average performance measure values, are presented in Appendix D.

Results with regards to Performance Measures

Table 9.1 presents the wins and losses for each performance measure calculated over all DMOOPs and all n_t-τ_t combinations.

Table 9.1: Overall Wins and Losses for Various Performance Measures

PM	Results	pbest-gbest combination															
		s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
acc	Wins	222	219	229	287	150	146	164	149	165	161	156	162	171	138	144	188
acc	Losses	347	345	313	225	169	165	109	128	149	120	123	123	152	151	137	95
acc	Diff	-125	-126	-84	62	-19	-19	55	21	16	41	33	39	19	-13	7	93
acc	Rank	15	16	14	2	12	12	3	7	9	4	6	5	8	11	10	**1**
stab	Wins	297	246	300	267	61	50	39	62	72	32	59	28	64	35	60	34
stab	Losses	111	110	69	96	88	132	126	113	84	113	102	126	84	136	89	127
stab	Diff	186	136	231	171	-27	-82	-87	-51	-12	-81	-43	-98	-20	-101	-29	-93
stab	Rank	2	4	**1**	3	7	12	13	10	5	11	9	15	6	16	8	14
NS	Wins	267	396	348	449	67	226	243	132	81	236	141	241	62	233	135	251
NS	Losses	303	205	212	171	384	148	131	190	380	133	165	158	377	141	182	141
NS	Diff	-36	191	136	278	-317	78	112	-58	-299	103	-24	83	-315	92	-47	110
NS	Rank	11	2	3	**1**	16	9	4	13	14	6	10	8	15	7	12	5

With regards to *acc*, the following observations are made:

- The best and second best performance were obtained by p_d-g_r and p_s-g_r respectively.
- All p_s combinations, except p_s-g_r, performed poorly and p_s-g_n obtained the worst

rank. With regards to the g_s combinations, p_s-g_s and p_n-g_s performed poorly. However, p_d-g_s and p_r-g_s performed reasonably well.

- For the p_n combinations, p_n-g_d and p_n-g_r performed well, but p_n-g_s and p_n-g_n performed poorly. All g_n combinations performed poorly, except p_r-g_n that performed well.
- All the p_d combinations performed average, except p_d-g_r that obtained the best performance. For the g_d combinations, p_n-g_d and p_r-g_d performed well. However, p_s-g_d and p_d-g_d performed badly.
- All p_r combinations performed reasonably well and all g_r combinations performed really well.
- With the exception of p_r-g_r, using the same update approach for both pbest and gbest lead to a poor performance.

The following observations are made with regards to *stab*:

- The best performance was obtained by p_s-g_d and the worst by p_d-g_n.
- In contrast to their performance with regards to *acc*, all p_s combinations performed really well with regards to *stab*. Furthermore, all g_s combinations performed well.
- Except p_n-g_s that performed well, all p_n combinations performed average or poorly. The g_n combinations obtained a mixed performance with regards to *stab*. A good performance was obtained by p_s-g_n, an average performance by p_r-g_n and a poor performance by p_n-g_n and p_d-g_n.
- For the p_d combinations, p_d-g_s and p_d-g_d performed well. However, p_d-g_n and p_d-g_r performed really bad. All g_d combinations performed well, except p_n-g_d that performed poorly.
- In contrast with the p_r combinations' performance with regards to *acc*, p_r-g_s performed well with regards to *stab*, p_r-g_n and p_r-g_d performed average and p_r-g_r performed poorly. All g_r combinations performed rather poorly, except p_s-g_r that performed well.
- Using the same update approach for both pbest and gbest produced a really good performance for p_s-g_s, an average performance for p_d-g_d and a poor performance for p_n-g_n and p_r-g_r.

With regards to NS, the following observations are made:

- The best performance was obtained by p_s-g_r and p_n-g_s performed the worst.
- All p_s combinations produced good results, except p_s-g_s. However, all g_s combinations performed poorly.
- Two p_n combinations performed poorly, namely p_n-g_s and p_n-g_r. However, p_n-g_r performed well and p_n-g_n performed average. All g_n combinations performed average, except p_s-g_n that performed well.
- For the p_d combinations, p_d-g_n and p_d-g_r performed well, while p_d-g_s and p_d-g_d performed poorly. Mixed results were also obtained by g_d. Good performance was achieved with p_s-g_d and p_n-g_d. However, average and poor performance were obtained by p_r-g_d and p_d-g_d respectively.
- All p_r combinations performed average, except p_r-g_s that performed poorly. All g_r combinations performed well or average, except p_n-g_r that performed badly.
- Using the same update approach for both pbest and gbest lead to either average (p_n-g_n and p_r-g_r) or poor performance (p_s-g_s and p_d-g_d).

The guide update approach of the original VEPSO algorithm, p_s-g_s, obtained the second lowest rank with regards to *acc*, the second best rank with regards to *stab* and the eleventh rank with regards to NS. Therefore, the guide update approaches that use Pareto-dominance information outperformed this approach with regards to all performance measures.

Another approach to measure the performance of a DMOO algorithm, is to analyse the performance of the algorithm in various types of environments, such as a fast or slow changing environment and a gradually or severely changing environment. Therefore, the next section discusses the overall performance of the guide update approaches, measured over all performance measures and all n_t-τ_t combinations.

Results with regards to Various Frequencies and Severities of Change

The wins and losses calculated over all performance measures and DMOOPs for the various n_t-τ_t combinations are presented in Table 9.2.
For a fast changing environment ($n_t = 10$ and $\tau_t = 10$) the following observations are made:

- The best performance was obtained by p_s-g_r and p_n-g_s performed the worst.
- All p_s combinations performed really well. Two g_s combinations performed well, namely p_s-g_s and p_r-g_s. The other two g_s combinations performed poorly.
- A poor performance was obtained by all p_n combinations, except p_n-g_d that performed well. In contrast, a good performance was obtained by all g_n combinations, except p_n-g_n.
- All p_r combinations performed average, except p_r-g_s that performed poorly. However, all g_r combinations performed well, except p_n-g_r.
- An good or average performance was obtained by all p_d combinations, except p_d-g_s that obtained a poor performance. In addition, a good or average performance was obtained by all g_d combinations.

Table 9.2: Overall Wins and Losses for Various Frequencies and Severities of Change

$\mathbf{n_t}$	τ_t	**ResultsResults**	**pbest-gbest combination**															
			s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
10	25	Diff	81	123	145	98	-102	-14	-17	-20	-86	-22	-5	-17	-98	-34	-25	-7
10	25	Rank	4	2	**1**	3	16	7	8	10	14	11	5	8	15	13	12	6
10	50	Wins	126	172	151	219	55	124	136	72	68	133	72	130	58	116	75	148
10	50	Losses	174	141	135	127	159	92	71	86	148	75	74	87	156	97	81	65
10	50	Diff	-48	31	16	92	-104	32	65	-14	-80	58	-2	43	-98	19	-6	83
10	50	Rank	13	7	9	**1**	16	6	3	12	14	4	10	5	15	8	11	2
1	10	Wins	174	169	180	200	82	85	101	76	100	87	85	80	89	96	75	95
1	10	Losses	197	177	176	106	117	105	71	109	116	71	83	82	100	93	96	75
1	10	Diff	-23	-8	4	94	-35	-20	30	-33	-16	16	2	-2	-11	3	-21	20
1	10	Rank	14	9	5	**1**	16	12	2	15	11	4	7	8	10	6	13	3
20	10	Wins	125	144	144	177	54	81	67	62	57	73	61	70	50	68	61	65
20	10	Losses	142	100	106	76	101	65	74	67	98	59	81	71	105	76	69	69
20	10	Diff	-17	44	38	101	-47	16	-7	-5	-41	14	-20	-1	-55	-8	-8	-4
20	10	Rank	12	2	3	**1**	15	4	9	8	14	5	13	6	16	10	10	7

The following observations are made for a slower changing environment, i.e. with $\tau_t = 25$ and $\tau_t = 50$:

- The best performance for $\tau_t = 25$ and $\tau_t = 50$ were obtained by p_s-g_d and p_s-g_r respectively. For both $\tau_t = 25$ and $\tau_t = 50$, the worst performance was obtained by p_n-g_s.
- All p_s combinations performed poorly, except p_s-g_s that performed well for $\tau_t = 25$. For both $\tau_t = 25$ and $\tau_t = 50$, all g_s combinations performed poorly, except p_s-g_s

that performed well for $\tau_t = 25$.

- For $\tau_t = 25$ all p_n combinations performed average, except p_n-g_s that performed poorly. However, with $\tau_t = 50$, p_n-g_n and p_n-g_d performed well, while the other two p_n combinations performed poorly. For $\tau_t = 50$ all g_n combinations performed well. However, for $\tau_t = 25$, p_s-g_n and p_n-g_n performed well, while p_d-g_n and p_r-g_n performed poorly.
- In both environments, all p_r combinations performed well or average, except p_r-g_s that performed poorly. All g_r combinations performed well for both $\tau_t = 25$ and $\tau_t = 50$, except p_n-g_r that performed average for $\tau_t = 25$ and poorly for $\tau_t = 50$.
- All p_d combinations performed poorly for $\tau_t = 25$, except p_d-g_r that performed well. However, for $\tau_t = 50$, p_d-g_r and p_d-g_n obtained a good performance, while the other two p_d combinations performed poorly. For the g_d combinations, all combinations performed well for $\tau_t = 25$, except p_d-g_d. For $\tau_t = 50$, all g_d combinations obtained an average performance.

For a severely changing environment ($n_t = 1$), the following observations are made:

- The best rank was obtained by p_s-g_r and the worst rank by p_n-g_s.
- All p_s combinations performed well, except p_s-g_s that performed poorly. In contrast, all g_s combinations performed rather poorly.
- Only one p_n combination, namely p_n-g_d performed well, while the other p_n combinations performed poorly. On the other hand, all g_n combinations performed well, except p_n-g_n that performed poorly.
- An average performance was obtained by p_r-g_s, and the other p_r combinations performed well. With the exception of p_d-g_r that performed really bad, all g_r combinations obtained a good rank.
- A good rank was obtained by p_d-g_n and p_d-g_r, an average rank by p_d-g_s and a poor rank by p_d-g_d. All g_d combinations performed well, except p_d-g_d that performed badly.

The following observations are made for a gradually changing environment ($n_t = 20$):

- The best performance was obtained by p_s-g_r, while p_d-g_s performed the worst.
- A really good performance was obtained by all p_s combinations, except p_s-g_s that

performed poorly. In contrast, a very bad rank was obtained by all g_s combinations.

- All p_n combinations performed well, except p_n that obtained a very poor performance. A good performance was also obtained by all g_n combinations.
- Two p_r combinations performed well, namely p_r-g_n and p_r-g_r. However, the other two p_r combinations performed badly. All g_r combinations performed well.
- All p_d combinations obtained an average performance, except p_d-g_s that obtained a very bad performance. A good or average performance was obtained by all g_d combinations, except p_r-g_d that performed poorly.

The original VEPSO algorithm's guide update approach, p_s-g_s, obtained the third and fourth highest rank for $n_t = 10$ and $\tau_t = 10$, and $n_t = 10$ and $\tau_t = 25$ respectively. However, p_s-g_s obtained rank thirteen, fourteen and twelve for $n_t = 10$ and $\tau_t = 50$, $n_t = 1$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$ respectively. Therefore, p_s-g_s struggles in slower changing environments, as well as environments that change either gradually or more severely.

Results for Various Dynamic Multi-objective Optimisation Problem Types

The DMOOPs against which DVEPSO was tested against, are of various DMOOP Types. With the different DMOOP Types, the POS or POF or both change over time. This section discusses the performance of the various guide update approaches with regards to the DMOOP Types I, II and III.

Type I DMOOPs

The wins and losses of the guide update approaches for Type I DMOOPs with regards to the performance measures over all n_t-τ_t combinations are presented in Table 9.3. The Type I DMOOPs are DIMP2 and dMOP3.

The following observations are made with regards to *acc*:

- The best performance with regards to *acc* was obtained by p_r-g_s and the worst performance by p_s-g_s.
- All p_s combinations performed really poor. Two g_s combinations performed well, namely p_r-g_s and p_d-g_s. However, the other two g_s combinations obtained a poor rank.

- Two p_n combinations, p_n-g_d and p_n-g_r, performed well, while the other two p_n combinations performed poorly. A similar trend was observed with the g_d combinations, where p_d-g_n and p_r-g_n performed well and the other two g_n combinations obtained a poor performance.
- For the p_r combinations, p_r-g_s and p_r-g_n performed really well and p_r-g_d and p_r-g_r performed average.
- All p_d combinations performed really well. In contrast, all g_d combinations obtained an average performance, except p_s-g_d that performed poorly.

Table 9.3: Overall Wins and Losses solving Type I DMOOPs for Various Performance Measures

PM	**Results**	**pbest-gbest combination**															
		s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
acc	Wins	0	0	1	0	26	27	33	30	34	31	31	28	29	28	32	32
acc	Losses	82	76	70	72	12	16	8	2	0	1	9	4	2	2	6	0
acc	Diff	-82	-76	-69	-72	14	11	25	28	34	30	22	24	27	26	26	32
acc	Rank	16	15	13	14	11	12	8	4	**1**	3	10	9	5	6	6	2
stab	Wins	1	0	0	0	10	8	13	10	16	8	9	9	8	11	10	14
stab	Losses	32	36	19	15	1	5	2	1	0	0	9	2	0	1	4	0
stab	Diff	-31	-36	-19	-15	9	3	11	9	16	8	0	7	8	10	6	14
stab	Rank	15	16	14	13	5	11	3	5	**1**	7	12	9	7	4	10	2
NS	Wins	0	11	0	0	13	14	12	12	12	12	12	13	12	12	12	12
NS	Losses	36	36	36	36	1	1	3	1	1	1	2	0	1	1	1	2
NS	Diff	-36	-25	-36	-36	12	13	9	11	11	11	10	13	11	11	11	10
NS	Rank	14	13	14	14	3	**1**	12	4	4	4	10	**1**	4	4	4	10
all	Wins	1	11	1	0	49	49	58	52	62	51	52	50	49	51	54	58
all	Losses	150	148	125	123	14	22	13	4	1	2	20	6	3	4	11	2
all	Diff	-149	-137	-124	-123	35	27	45	48	61	49	32	44	46	47	43	56
all	Rank	16	15	14	13	10	12	7	4	**1**	3	11	8	6	5	9	2

With regards to *stab*, the following observations are made:

- Similar to *acc*, the best rank was obtained by p_r-g_s. The worst rank was obtained by p_s-g_n.
- Similar to their performance with regards to *acc*, all p_s combinations performed poorly. With regards to the g_s combinations, all g_s combinations obtained a good performance, except p_s-g_s.
- All p_n combinations performed well, except p_n-g_n that performed badly. Similar to

acc, p_d-g_n and p_r-g_n performed well, while the other two g_n combinations obtained a poor performance.

- A good performance was obtained by all p_r combinations, except p_r-g_d that performed poorly. Similarly, all g_r combinations performed well, except p_s-g_r that obtained a poor rank.
- All p_d combinations performed well, except p_d-g_d that performed average. The g_d combinations obtained mixed results. A good performance was obtained by p_n-g_d, an average performance by p_d-g_d and a poor performance by p_r-g_d and p_s-g_d.

The following observations are made with regards to NS:

- The best performance was obtained by p_r-g_r.
- Once again, all p_s combinations performed poorly. However, all g_s combinations performed well, except p_s-g_s that performed badly.
- Similar to *acc* and *stab*, all p_n combinations obtained a good performance, except p_n-g_n that performed poorly. The same trend was observed for g_d combinations, with all performing well, except p_s-g_n that performed badly.
- All p_r combinations performed well, with the exception of p_r-g_d that performed average. Two g_r combinations, p_r-g_r and p_n-g_r, performed well, p_d-g_r performed average and p_s-g_r performed poorly.
- With the exception of p_d-g_r that performed average, all p_r combinations performed well. In contrast, p_d-g_d performed well, p_r-g_d performed average and the other two g_d combinations performed badly.

Table 9.4 presents the wins and losses measured over all performance measures for the various n_t-τ_t combinations for Type I DMOOPs.
The following observations are made with regards to the obtained results:

- All p_s combinations performed poorly for all n_t-τ_t combinations, except p_s-g_s that performed well for $n_t = 20$ and $\tau_t = 10$. For the g_s combinations, three combinations performed well with one performing poorly for $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 50$, and $n_t = 1$ and $\tau_t = 10$, two g_s combinations performed well, and two combinations performed badly.

Table 9.4: Overall Wins and Losses solving Type I DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	Results	pbest-gbest combination															
			s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
10	10	Wins	0	0	0	0	22	22	21	20	24	23	22	22	22	23	27	33
10	10	Losses	63	80	68	63	0	0	2	3	0	0	0	0	0	0	2	0
10	10	Diff	-63	-80	-68	-63	22	22	19	17	24	23	22	22	22	23	25	33
10	10	Rank	13	16	15	13	6	6	11	12	3	4	6	6	6	4	2	**1**
10	25	Wins	0	0	1	0	9	6	6	11	7	5	5	7	5	4	7	7
10	25	Losses	30	14	12	22	0	0	0	0	0	0	0	0	0	2	0	0
10	25	Diff	-30	-14	-11	-22	9	6	6	11	7	5	5	7	5	2	7	7
10	25	Rank	16	14	13	15	2	7	7	**1**	3	9	9	3	9	12	3	3
10	50	Wins	0	11	0	0	6	5	7	6	6	6	11	8	7	7	6	6
10	50	Losses	21	24	13	13	2	7	1	1	1	1	2	2	1	1	1	1
10	50	Diff	-21	-13	-13	-13	4	-2	6	5	5	5	9	6	6	6	5	5
10	50	Rank	16	13	13	13	11	12	2	6	6	6	**1**	2	2	2	6	6
1	10	Wins	0	0	0	0	9	10	24	11	24	12	14	10	15	11	10	11
1	10	Losses	36	30	25	25	12	15	2	0	0	1	1	4	0	1	8	1
1	10	Diff	-36	-30	-25	-25	-3	-5	22	11	24	11	13	6	15	10	2	10
1	10	Rank	16	15	13	13	11	12	2	5	**1**	5	4	9	3	7	10	7
20	10	Wins	1	0	0	0	3	6	0	4	1	5	0	3	0	6	4	1
20	10	Losses	0	0	7	0	0	0	8	0	0	0	17	0	2	0	0	0
20	10	Diff	1	0	-7	0	3	6	-8	4	1	5	-17	3	-2	6	4	1
20	10	Rank	8	11	14	11	6	**1**	15	4	8	3	16	6	13	**1**	4	8
all	all	Wins	1	11	1	0	49	49	58	52	62	51	52	50	49	51	54	58
all	all	Losses	150	148	125	123	14	22	13	4	1	2	20	6	3	4	11	2
all	all	Diff	-149	-137	-124	-123	35	27	45	48	61	49	32	44	46	47	43	56
all	all	Rank	16	15	14	13	10	12	7	4	**1**	3	11	8	6	5	9	2

- Two p_n combinations performed well and two poorly for $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 1$ and $\tau_t = 10$. All p_n combinations performed well for $n_t = 10$ and $\tau_t = 25$. For $n_t = 20$ and $\tau_t = 10$, three p_n combinations performed well and only one performed badly. The g_n combinations also obtained mixed results. For $n_t = 10$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$, three combinations performed well and one performed badly. On the other hand, for the other n_t-τ_t combinations two g_n combinations performed well and two poorly.
- All p_r combinations performed well for all n_t-τ_t combinations, except p_r-g_d that performed poorly for $n_t = 20$ and $\tau_t = 10$. For all n_t-τ_t combinations, three g_r combinations obtained a good performance and one combination obtained a poor performance, except for $n_t = 10$ and $\tau_t = 10$ where two combinations performed

badly.

- For the p_d combinations, all performed well for $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 1$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$, all p_d combinations, except p_s-g_d, obtained a good performance. Three of the g_d combinations obtained a good performance and one combination obtained a poor performance for $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$. For $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$, two combinations performed well and two performed poorly. Furthermore, for $n_t = 20$ and $\tau_t = 10$, only one g_d combination obtained a good performance and the other three obtained a bad performance.

Type II DMOOPs

The wins and losses for Type II DMOOPs with regards to the performance measures over all n_t-τ_t combinations are presented in Table 9.5. The Type II DMOOPs are FDA1$_{Zhou}$, FDA2, FDA3, FDA3$_{Camara}$, dMOP2, dMOP2$_{iso}$ and dMOP2$_{dec}$.

Table 9.5: Overall Wins and Losses solving Type II DMOOPs for Various Performance Measures

PM	**Results**	**pbest-gbest combination**															
		s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
acc	Wins	62	95	87	144	47	76	73	47	56	82	53	81	51	70	45	86
acc	Losses	181	144	155	73	64	48	30	68	58	32	58	35	64	48	62	35
acc	Diff	-119	-49	-68	71	-17	28	43	-21	-2	50	-5	46	-13	22	-17	51
acc	Rank	16	14	15	**1**	11	6	5	13	8	3	9	4	10	7	11	2
stab	Wins	100	109	111	92	46	33	15	46	50	16	46	13	49	16	45	16
stab	Losses	52	37	34	68	30	71	69	43	27	59	36	70	28	80	28	71
stab	Diff	48	72	77	24	16	-38	-54	3	23	-43	10	-57	21	-64	17	-55
stab	Rank	3	2	**1**	4	8	11	13	10	5	12	9	15	6	16	7	14
NS	Wins	52	83	67	126	32	109	141	43	45	107	47	139	25	107	49	146
NS	Losses	147	110	125	81	132	45	28	73	127	44	72	38	135	56	75	30
NS	Diff	-95	-27	-58	45	-100	64	113	-30	-82	63	-25	101	-110	51	-26	116
NS	Rank	14	10	12	7	15	4	2	11	13	5	8	3	16	6	9	**1**
all	Wins	214	287	265	362	125	218	229	136	151	205	146	233	125	193	139	248
all	Losses	380	291	314	222	226	164	127	184	212	135	166	143	227	184	165	136
all	Diff	-166	-4	-49	140	-101	54	102	-48	-61	70	-20	90	-102	9	-26	112
all	Rank	16	8	12	**1**	14	6	3	11	13	5	9	4	15	7	10	2

The following are observed with regards to *acc*:

- The best performance was achieved by p_s-g_r and the worst by p_s-g_s.

- All p_s combinations performed poorly, except p_s-g_r that performed really well. Two g_s combinations, p_r-g_s and p_d-g_s performed average and the other two g_s combinations obtained a poor performance.
- Two p_n combinations, p_n-g_s and p_n-g_r performed well and the other two performed badly. All g_n combinations performed well, except p_s-g_n that performed poorly.
- A good performance was obtained by all p_r combinations. With the exception of p_n-g_r that performed poorly, all g_r combinations obtained a good performance.
- The p_d combinations obtained mixed results. A good performance was obtained by p_d-g_n and p_d-g_r, an average performance by p_d-g_s and a poor performance by p_d-g_d. Two g_s combinations obtained a good performance, namely p_n-g_d and p_r-g_d. The other two g_d combinations performed badly.

The following observations are made with regards to *stab*:

- The best rank was obtained by p_s-g_d and the worst by p_d-g_n.
- In contrast to *acc*, all p_s combinations performed really well with regards to *stab*, obtaining the top four ranks. Furthermore, all g_s combinations obtained a good performance.
- Two p_n combinations, p_n-g_s and p_n-g_r obtained an average performance. The other two p_n combinations performed poorly. All g_n combinations performed badly with the exception of p_s-g_n that performed very good.
- For the p_r combinations, p_r-g_s and p_r-g_d performed well, but the other two combinations performed badly. The g_r combinations obtained mixed results, with p_s-g_r performing well, p_n-g_r performing average and the other two g_r combinations performing poorly.
- Two p_d combinations obtained a good performance, namely p_d-g_r and p_d-g_n. Furthermore, an average performance was obtained by p_d-g_s and a poor performance by p_d-g_d. In contrast, all g_d combinations performed well, except p_s-g_d that performed badly.

The following are observed with regards to NS:

- The best performance was achieved by p_d-g_r and the worst by p_d-g_s.
- Two p_s combinations, p_s-g_r and p_s-g_n, performed well. The other two p_s combina-

tions performed poorly. A bad performance was also obtained by all g_s combinations.

- For the p_n combinations, p_n-g_n and p_n-g_d performed well, but the other two p_n combinations performed badly. All g_n combinations performed well or average.
- All p_r combinations, except p_r-g_s, performed well. Similarly, all g_r combinations obtained a good performance, except p_n-g_r that performed poorly.
- For the p_d combinations, all performed well, except p_d-g_s that performed badly. Similarly, all g_d combinations obtained a good rank, except p_s-g_d that obtained a poor rank.

Table 9.6 presents the wins and losses measured over all performance measures for the various n_t-τ_t combinations for Type II DMOOPs.

Table 9.6: Overall Wins and Losses solving Type II DMOOPs for Various Frequencies and Severities of Change

n_t	τ_t	Results	pbest-gbest combination															
			s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	Wins	58	64	67	89	31	35	46	32	28	38	26	46	36	40	32	48
10	10	Losses	85	76	67	41	45	48	30	38	38	39	40	32	38	33	39	27
10	10	Diff	-27	-12	0	48	-14	-13	16	-6	-10	-1	-14	14	-2	7	-7	21
10	10	Rank	16	12	6	**1**	14	13	3	9	11	7	14	4	8	5	10	2
10	25	Wins	72	75	81	75	14	39	29	22	23	30	41	34	21	20	22	44
10	25	Losses	44	39	30	41	62	33	32	38	58	33	33	34	61	34	36	34
10	25	Diff	28	36	51	34	-48	6	-3	-16	-35	-3	8	0	-40	-14	-14	10
10	25	Rank	4	2	**1**	3	16	7	9	13	14	9	6	8	15	11	11	5
10	50	Wins	25	54	39	83	34	76	90	37	45	78	38	90	25	77	45	96
10	50	Losses	103	75	93	82	77	41	33	40	71	32	41	41	82	54	37	30
10	50	Diff	-78	-21	-54	1	-43	35	57	-3	-26	46	-3	49	-57	23	8	66
10	50	Rank	16	11	14	8	13	5	2	9	12	4	9	3	15	6	7	**1**
1	10	Wins	42	65	57	67	24	35	32	26	27	31	26	30	22	34	24	32
1	10	Losses	78	64	77	38	21	27	18	46	28	16	31	21	22	33	31	23
1	10	Diff	-36	1	-20	29	3	8	14	-20	-1	15	-5	9	0	1	-7	9
1	10	Rank	16	8	14	**1**	7	6	3	14	11	2	12	4	10	8	13	4
20	10	Wins	17	29	21	48	22	33	32	19	28	28	15	33	21	22	16	28
20	10	Losses	70	37	47	20	21	15	14	22	17	15	21	15	24	30	22	22
20	10	Diff	-53	-8	-26	28	1	18	18	-3	11	13	-6	18	-3	-8	-6	6
20	10	Rank	16	13	15	**1**	8	2	2	9	6	5	11	2	9	13	11	7
all	all	Wins	214	287	265	362	125	218	229	136	151	205	146	233	125	193	139	248
all	all	Losses	380	291	314	222	226	164	127	184	212	135	166	143	227	184	165	136
all	all	Diff	-166	-4	-49	140	-101	54	102	-48	-61	70	-20	90	-102	9	-26	112
all	all	Rank	16	8	12	**1**	14	6	3	11	13	5	9	4	15	7	10	2

The following observations are made with regards to the obtained results:

- All p_s combinations performed well for $n_t = 10$ and $\tau_t = 25$. For $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$, two p_s combinations performed well and two performed poorly. For the other two n_t-τ_t combinations, only one p_s combination performed well and the other p_s combinations performed badly. For $n_t = 20$ and $\tau_t = 10$, all g_s combinations performed well, except one that performed really bad. For $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, three p_s combinations performed poorly and only one performed well. Furthermore, for $n_t = 20$ and $\tau_t = 10$, all g_s combinations obtained a poor performance.
- For $n_t = 20$ and $\tau_t = 10$, all p_n combinations obtained a good performance. Three p_n combinations performed well and one performed poorly for $n_t = 10$ and $\tau_t = 50$, and $n_t = 1$ and $\tau_t = 10$. However, for $n_t = 10$ and $\tau_t = 10$, and $n_t = 10$ and $\tau_t = 25$, two p_n combinations obtained a good performance and the other two a poor performance. All g_n combinations performed well for $n_t = 1$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 25$, and $n_t = 10$ and $\tau_t = 50$, all g_n combinations obtained a good performance, except one that performed badly. For the other two n_t-τ_t combinations, two g_n combinations obtained good ranks and the other two obtained poor ranks.
- All p_r combinations, except one, obtained a good performance for $n_t = 10$ and $\tau_t = 25$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 20$ and $\tau_t = 10$. For $n_t = 10$ and $\tau_t = 10$, and $n_t = 1$ and $\tau_t = 10$, two p_r combinations performed well and two performed badly. For the g_r combinations, all performed well for $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 20$ and $\tau_t = 10$. Furthermore, for $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, only one g_r combination performed poorly and all the other obtained a good performance.
- For $n_t = 10$ and $\tau_t = 10$, all p_d combinations performed well or average. Three p_d combinations performed well with only one p_d combination performing badly for $n_t = 10$ and $\tau_t = 25$. For $n_t = 20$ and $\tau_t = 10$, two p_d combinations obtained a good performance and two obtained a poor performance. Only one p_d combination performed well for $n_t = 10$ and $\tau_t = 25$, with three p_d combinations obtaining a poor performance. For $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 25$, and $n_t =$

10 and $\tau_t = 50$, all g_d combinations performed well, except one that performed poorly. However, three g_d combinations obtained a poor performance and only one performed well for $n_t = 1$ and $\tau_t = 10$, and $n_t = 20$ and $\tau_t = 10$.

Type III DMOOPs

The wins and losses of the guide update approaches for Type III DMOOPs with regards to the performance measures over all n_t-τ_t combinations are presented in Table 9.7. The Type III DMOOPs are FDA2$_{Camara}$, HE1, HE2, HE6, HE7 and HE9.

Table 9.7: Overall Wins and Losses solving Type III DMOOPs for Various Performance Measures

PM	**Results**	**pbest-gbest combination**															
		s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
acc	Wins	160	124	141	143	77	43	58	72	75	48	72	53	91	40	67	70
acc	Losses	84	125	88	80	93	101	71	58	91	87	56	84	86	101	69	60
acc	Diff	76	-1	53	63	-16	-58	-13	14	-16	-39	16	-31	5	-61	-2	10
acc	Rank	**1**	8	3	2	11	15	10	5	11	14	4	13	7	16	9	6
stab	Wins	196	137	189	175	5	9	11	6	6	8	4	6	7	8	5	4
stab	Losses	27	37	16	13	57	56	55	69	57	54	57	54	56	55	57	56
stab	Diff	169	100	173	162	-52	-47	-44	-63	-51	-46	-53	-48	-49	-47	-52	-52
stab	Rank	2	4	**1**	3	12	7	5	16	11	6	15	9	10	7	12	12
NS	Wins	215	302	281	323	22	103	90	77	24	117	82	89	25	114	74	93
NS	Losses	128	66	61	57	258	105	100	123	259	91	101	121	249	90	113	109
NS	Diff	87	236	220	266	-236	-2	-10	-46	-235	26	-19	-32	-224	24	-39	-16
NS	Rank	4	2	3	**1**	16	7	8	13	15	5	10	11	14	6	12	9
all	Wins	571	563	611	641	104	155	159	155	105	173	158	148	123	162	146	167
all	Losses	239	228	165	150	408	262	226	250	407	232	214	259	391	246	239	225
all	Diff	332	335	446	491	-304	-107	-67	-95	-302	-59	-56	-111	-268	-84	-93	-58
all	Rank	4	3	2	**1**	16	12	8	11	15	7	5	13	14	9	10	6

The following observations are made with regards to *acc*:

- The best performance was obtained by p_s-g_s and the worst performance by p_d-g_n.
- All p_s combinations performed well. Two g_s combinations, p_s-g_s and p_d-g_s, obtained a good performance, while the other two g_s combinations performed poorly.
- The p_n combinations obtained mixed results, with p_n-g_r performing well, p_n-g_d performing average and the other two p_n combinations performing badly. All g_n combinations obtained a poor performance, except p_s-g_n that performed well.
- All p_r combinations obtained a poor performance, except p_r-g_d that obtained a

good performance. In contrast, all g_r combinations performed well, except p_r-g_r that performed poorly.

- A good performance was obtained by all p_d combinations, except p_d-g_n that performed badly. Furthermore, all g_d combinations performed well or average.

With regards to *stab*, the following observations are made:

- A p_s combination, p_s-g_d, obtained the best performance. The worst performance was obtained by p_n-g_r.
- All p_s combinations performed very well, obtaining the top four ranks. However, only one g_s combination, p_s-g_s performed well, while the others performed poorly.
- Two p_n combinations obtained a good or average performance, namely p_n-g_r and p_n-g_d. The other two p_n combinations perfromed badly. In contrast, all g_n combinations obtained a good rank.
- For the p_r combinations, p_r-g_d and p_r-g_r performed well, while the other two p_r combinations performed poorly. Similarly, p_s-g_r and p_r-g_r obtained a good performance, while the other g_r combinations obtained a poor performance.
- Only one p_d combination, p_d-g_n, performed well. All other p_d combinations performed badly. For the g_d combinations, p_s-g_d and p_n-g_d obtained a good performance, but the other g_d combinations obtained a poor performance.

The following observations are made with regards to NS:

- The best performance was obtained by p_s-g_r and the worst performance by p_n-g_s.
- Similar to *stab*, all p_s combinations performed well, obtaining the top four ranks. However, similar to *stab*, for g_s only p_s-g_s obtained a good performance and the other g_s combinations performed poorly.
- Similar to *stab*, p_n-g_r and p_n-g_d performed well, while the other p_n combinations performed badly. Furthermore, also similar to *stab*, all g_n combinations obtained a good performance.
- Two p_r combinations, p_r-g_n and p_r-g_d, obtained a good or average performance. The other two p_r combinations obtained a poor performance. For g_r, p_s-g_r and p_d-g_r performed well, while the other g_r combinations performed badly.
- For p_d, p_d-g_n and p_d-g_r performed well, while p_d-g_s and p_d-g_d obtained a poor

performance. Two g_d combinations, p_s-g_d and p_n-g_d obtained a good performance. Furthermore, p_r-g_d obtained an average performance and p_d-g_d performed poorly.

Table 9.8 presents the wins and losses measured over all performance measures for the various n_t-τ_t combinations for Type III DMOOPs.

Table 9.8: Overall Wins and Losses solving Type III DMOOPs for Various Frequencies and Severities of Change

$\mathbf{n_t}$	τ_t	**Results**	**pbest-gbest combination**															
			s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
10	10	Wins	121	115	132	136	14	24	33	34	12	31	35	30	16	30	30	26
10	10	Losses	35	48	20	31	85	58	46	49	86	44	40	53	78	50	47	49
10	10	Diff	86	67	112	105	-71	-34	-13	-15	-74	-13	-5	-23	-62	-20	-17	-23
10	10	Rank	3	4	**1**	2	15	13	6	8	16	6	5	11	14	10	9	11
10	25	Wins	110	122	121	107	9	18	20	26	11	21	21	24	12	21	22	20
10	25	Losses	27	21	16	21	72	44	40	41	69	45	39	48	75	43	40	44
10	25	Diff	83	101	105	86	-63	-26	-20	-15	-58	-24	-18	-24	-63	-22	-18	-24
10	25	Rank	4	2	**1**	3	15	13	8	5	14	10	6	10	15	9	6	10
10	50	Wins	101	107	112	136	15	43	39	29	17	49	23	32	26	32	24	46
10	50	Losses	58	49	39	35	87	47	37	52	83	45	41	45	81	48	50	34
10	50	Diff	43	58	73	101	-72	-4	2	-23	-66	4	-18	-13	-55	-16	-26	12
10	50	Rank	4	3	2	**1**	16	8	7	12	15	6	11	9	14	10	13	5
1	10	Wins	132	104	123	133	49	40	45	39	49	44	45	40	52	51	41	52
1	10	Losses	83	83	74	43	84	63	51	63	88	54	51	57	78	59	57	51
1	10	Diff	49	21	49	90	-35	-23	-6	-24	-39	-10	-6	-17	-26	-8	-16	1
1	10	Rank	2	4	2	**1**	15	12	6	13	16	9	6	11	14	8	10	5
20	10	Wins	107	115	123	129	17	30	22	27	16	28	34	22	17	28	29	23
20	10	Losses	36	27	16	20	80	50	52	45	81	44	43	56	79	46	45	47
20	10	Diff	71	88	107	109	-63	-20	-30	-18	-65	-16	-9	-34	-62	-18	-16	-24
20	10	Rank	4	3	2	**1**	15	10	12	8	16	6	5	13	14	8	6	11
all	all	Wins	571	563	611	641	104	155	159	155	105	173	158	148	123	162	146	167
all	all	Losses	239	228	165	150	408	262	226	250	407	232	214	259	391	246	239	225
all	all	Diff	332	335	446	491	-304	-107	-67	-95	-302	-59	-56	-111	-268	-84	-93	-58
all	all	Rank	4	3	2	**1**	16	12	8	11	15	7	5	13	14	9	10	6

The following are observed with regards to the obtained results:

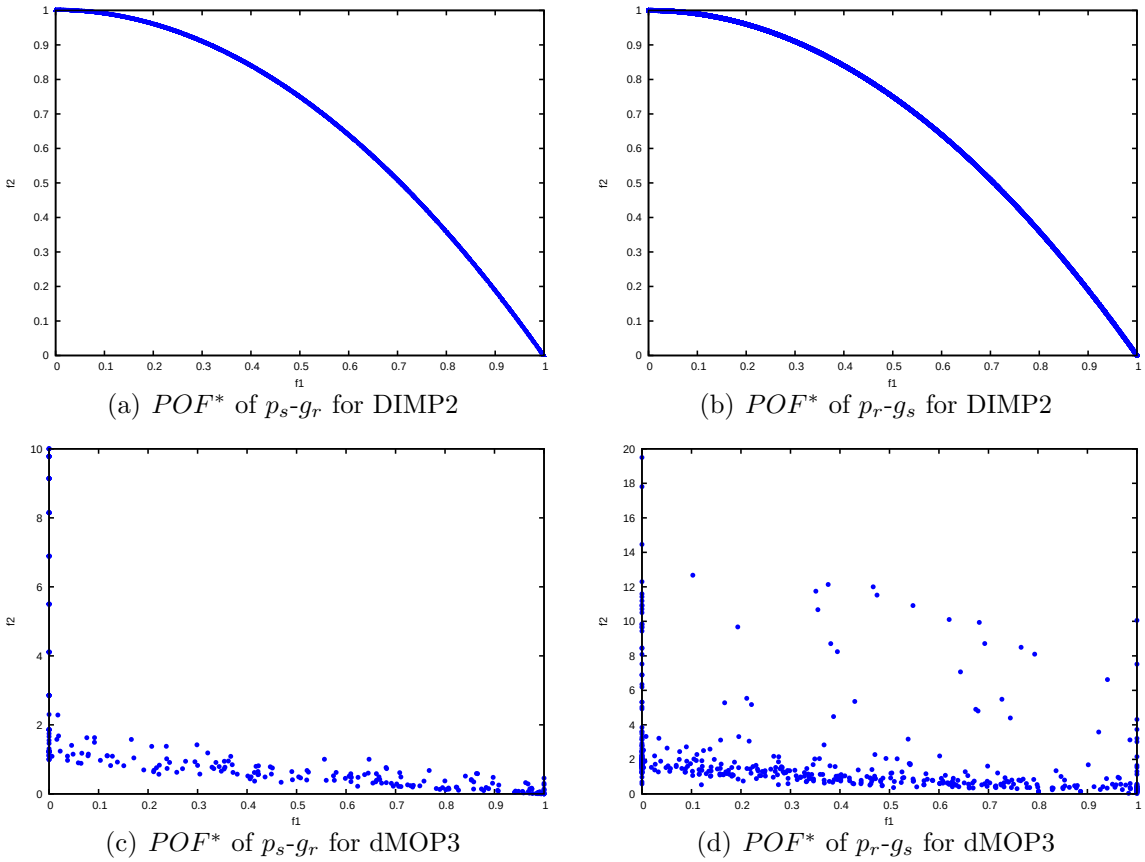
- All p_s combinations performed really well, obtaining the top four ranks for all n_t-τ_t combinations. For g_s mixed results were obtained. For $n_t = 1$ and $\tau_t = 10$, two g_s combinations obtained a good performance, while the other two performed badly. For the other n_t-τ_t combinations, only one g_s combination performed well, while the other three g_s combinations obtained a poor performance.

- For $n_t = 1$ and $\tau_t = 10$, only one p_n combination obtained a good performance, while the other p_n combinations performed poorly. Two p_n combinations performed well for the other n_t-τ_t combinations, while the other two p_n combinations obtained a bad rank. For $n_t = 10$ and $\tau_t = 50$, and $n_t = 1$ and $\tau_t = 10$, all g_n combinations obtained a good or average performance. Three g_n combinations performed well and one performed poorly for all the other n_t-τ_t combinations.
- Three p_r combinations obtained a good performance and one, p_r-g_s, a poor performance for $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$. For the other n_t-τ_t combinations, two p_r combinations produced good ranks and two produced poor ranks. All g_r combinations performed good or average for $n_t = 10$ and $\tau_t = 25$. For $n_t = 10$ and $\tau_t = 10$, two g_r combinations obtained a good performance and two obtained a poor performance. Three g_r combinations obtained good ranks and one obtained a poor rank for all the other n_t-τ_t combinations.
- A similar trend to p_r was observed for p_d. For $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $\tau_t = 25$, and $n_t = 1$ and $\tau_t = 10$, all g_d combinations performed good or average. Three g_d combinations obtained a good rank and one obtained a poor rank for $n_t = 20$ and $\tau_t = 10$. Furthermore, for $n_t = 10$ and $\tau_t = 50$, two g_d combinations performed well, and two performed badly.

General Observations with regards to DMOOP Types

It is interesting to note the difference in performance obtained by the p_s combinations for the three types of DMOOPs. It should be noted that although the p_s combinations ranked poorly for the Type I DMOOPs and three p_s combinations ranked poorly for the Type II DMOOPs, it does not indicate that they didn't successfully track the changing POF. It only indicates that the other guide-update approaches' performance measure values were statistically better, resulting in more wins. This is indicated in Figure 9.2. Figure 9.2 illustrates the approximated POF of the best performing p_s combination (p_s-g_r) and the best performing guide update approach for the Type I DMOOPs (p_r-g_s). When solving DIMP2, both guide update approaches successfully found the POF. However, over the various runs, p_s-g_r did find more outlier solutions that were further away from the true POF than p_r-g_s. Therefore, p_r-g_s obtained a better rank. For dMOP3, both guide update approaches found solutions close to the true POF. Even

though p_r-g_s found more outlier solutions than p_s-g_r in some of the runs, p_r-g_s found much more solutions with a better spread than p_s-g_r. Therefore, p_r-g_s obtained better perfomance measure values.



(a) POF^* of p_s-g_r for DIMP2

(b) POF^* of p_r-g_s for DIMP2

(c) POF^* of p_s-g_r for dMOP3

(d) POF^* of p_r-g_s for dMOP3

Figure 9.2: POF^* of p_s-g_r on the right and p_r-g_s on the left for $n_t = 10$ and $\tau_t = 10$

The next section discusses the overall performance of the various guide update approaches. This overall perfromance is measured over all performance measures and all n_t-τ_t combinations.

Overall Performance

The overall wins and losses obtained by the various guide update approaches are presented in Table 9.9.

With regards to the overall performance of the guide update approaches, the following observations are made:

- The best overall rank was obtained by p_s-g_r and the worst by p_n-g_s.
- All p_s combinations obtained a good rank. However, all g_s combinations, except p_s-g_s, ranked the worst.
- Two p_n combinations, namely p_n-g_d and p_n-g_n, obtained a good and average rank respectively. The other two p_n combinations obtained a bad rank. On the other hand, an average or good performance was obtained by all g_n combinations.
- Two p_d combinations performed well, namely p_d-g_n and p_d-g_r. The other two p_d combinations performed poorly. Similarly, two g_d combinations, namely p_s-g_d and p_n-g_d, performed well, while the other two performed poorly.
- For the p_r combinations, p_r-g_n and p_r-g_r performed average and the other two p_r combinations obtained a poor rank. In contrast, all g_r combinations, except p_n-g_r, performed well.

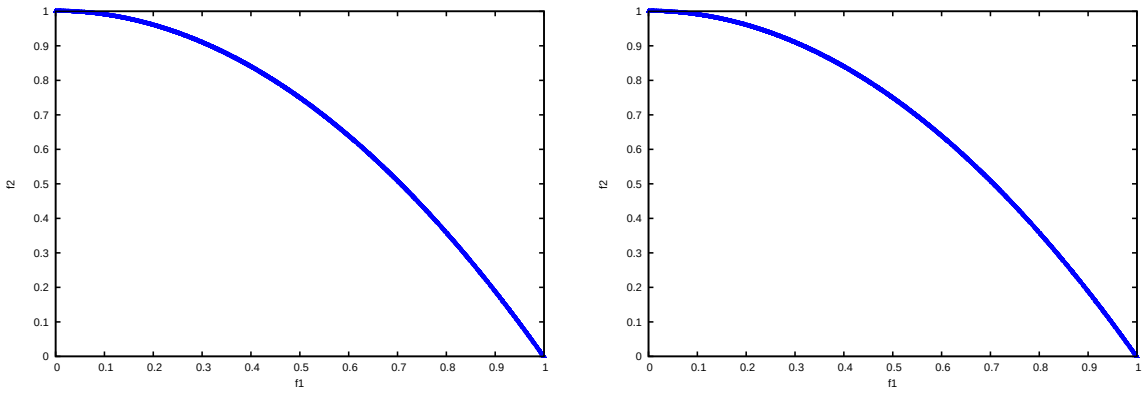
Table 9.9: Overall Wins and Losses by the various guide update approaches

Results	**pbest-gbest combination**															
	s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
Wins	786	861	877	1003	278	422	446	343	318	429	356	431	297	406	339	473
Losses	761	660	594	492	641	445	366	431	613	366	390	407	613	428	408	363
Diff	25	201	283	511	-363	-23	80	-88	-295	63	-34	24	-316	-22	-69	110
Rank	7	3	2	**1**	16	10	5	13	14	6	11	8	15	9	12	4

The best performing guide update approach, p_s-g_r uses no Pareto-dominance information for the pbest update. This enables the swarm to focus on optimising its specific objective, without taking the other objectives into account. However, for the gbest update Pareto-dominance information is taken into account. When a pbest is non-dominated with regards to the gbest, either the pbest or the current gbest is randomly selected as the new gbest. Therefore, Pareto-dominance is not required for a gbest update.

POFs found by DVEPSO using the p_s-g_r guide update approach during a single run for DIMP2 are illustrated in Figure 9.3. In Figure 9.3, POFs found for $n_t = 10$ and $\tau_t = 10$ (fast changing environment) are shown on the left and for $n_t = 1$ and $\tau_t = 10$ (severely changing environment) are shown on the right. The figures indicate

that DVEPSO successfully tracked the changing POF of DIMP2 in both a fast changing and severely changing environment.

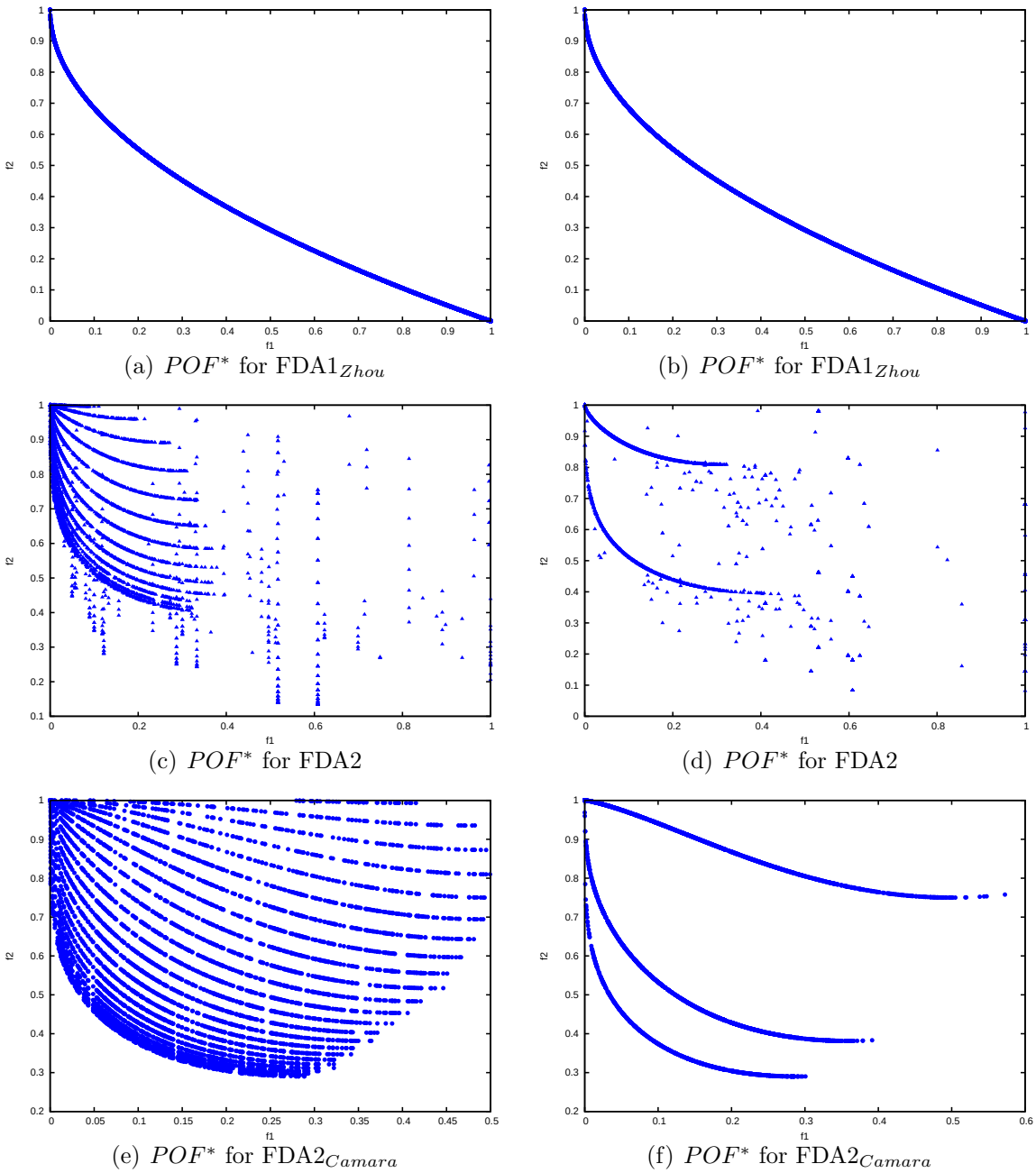


(a) (b)

Figure 9.3: POF^* for DIMP2 of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right

Figures 9.4 and 9.5 illustrate POFs found by DVEPSO using the p_s-g_r guide update approach during a single run for the FDA DMOOPs. Figures 9.4(a) and 9.4(b) indicate that DVEPSO successfully tracked the changing POF over time for FDA1$_{Zhou}$ in both a fast changing and severely changing environment. For FDA2, DVEPSO struggled to track the changing POF for every change in the environment, but did find a POF^* close to POF for many time steps, even though the spread of solutions were not that good. This is illustrated in Figures 9.4(c) and 9.4(d). However, for FDA2$_{Camara}$, DVEPSO successfully tracked the changing POF^* over time with a good spread of solutions as indicated in Figures 9.4(e) and 9.4(f). From Figure 9.5 it can be seen that DVEPSO successfully tracked the changing POF, finding a good spread of solutions for both $FDA3$ and FDA3$_{Camara}$ with a fast changing environment. However, with a severely changing environment, DVEPSO did not find as good a spread of solutions as in the case with a fast changing environment.

POFs found by DVEPSO using the p_s-g_r guide update approach during a single run for the dMOP DMOOPs are illustrated in Figures 9.6 and 9.7. When solving dMOP2, DVEPSO successfully tracked the changing POF over time for both a fast changing,

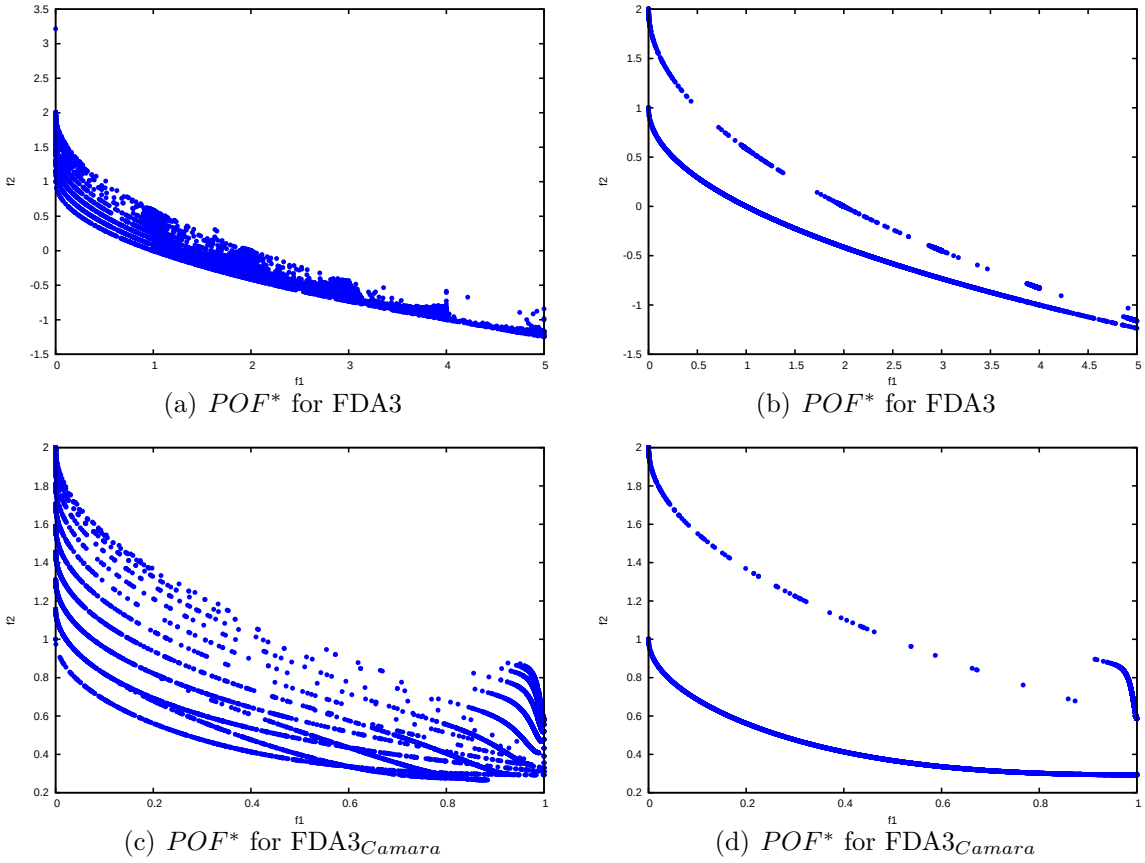


(a) POF^* for FDA1$_{Zhou}$ (b) POF^* for FDA1$_{Zhou}$

(c) POF^* for FDA2 (d) POF^* for FDA2

(e) POF^* for FDA2$_{Camara}$ (f) POF^* for FDA2$_{Camara}$

Figure 9.4: POF^* for FDA1 and FDA2 functions of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right



(a) POF^* for FDA3 (b) POF^* for FDA3

(c) POF^* for FDA3$_{Camara}$ (d) POF^* for FDA3$_{Camara}$

Figure 9.5: POF^* for FDA3 functions of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right

and severely changing environment, as illustrated in Figure 9.6. However, DVEPSO also found many outlier solutions. Figure 9.6 indicates the outlier solutions found by DVEPSO while solving the dMOP2 functions. The POF^* found by DVEPSO for the dMOP2 functions without the outliers are shown in Figure 9.7. When dMOP2 had a deceptive POF, DVEPSO found outlier solutions that were very far away from POF as can be seen in Figure 9.6(h). These outlier solutions caused large reference vectors being used to calculate the HV values and therefore very large acc values were reported (refer to Appendix D).

When solving dMOP3 with a fast changing environment, DVEPSO found a good

spread of solutions in the area of POF. However, with a severely changing environment, DVEPSO found a reasonable spread of solutions reasonably close to POF. This is illustrated in Figures 9.6(a) and 9.6(b).

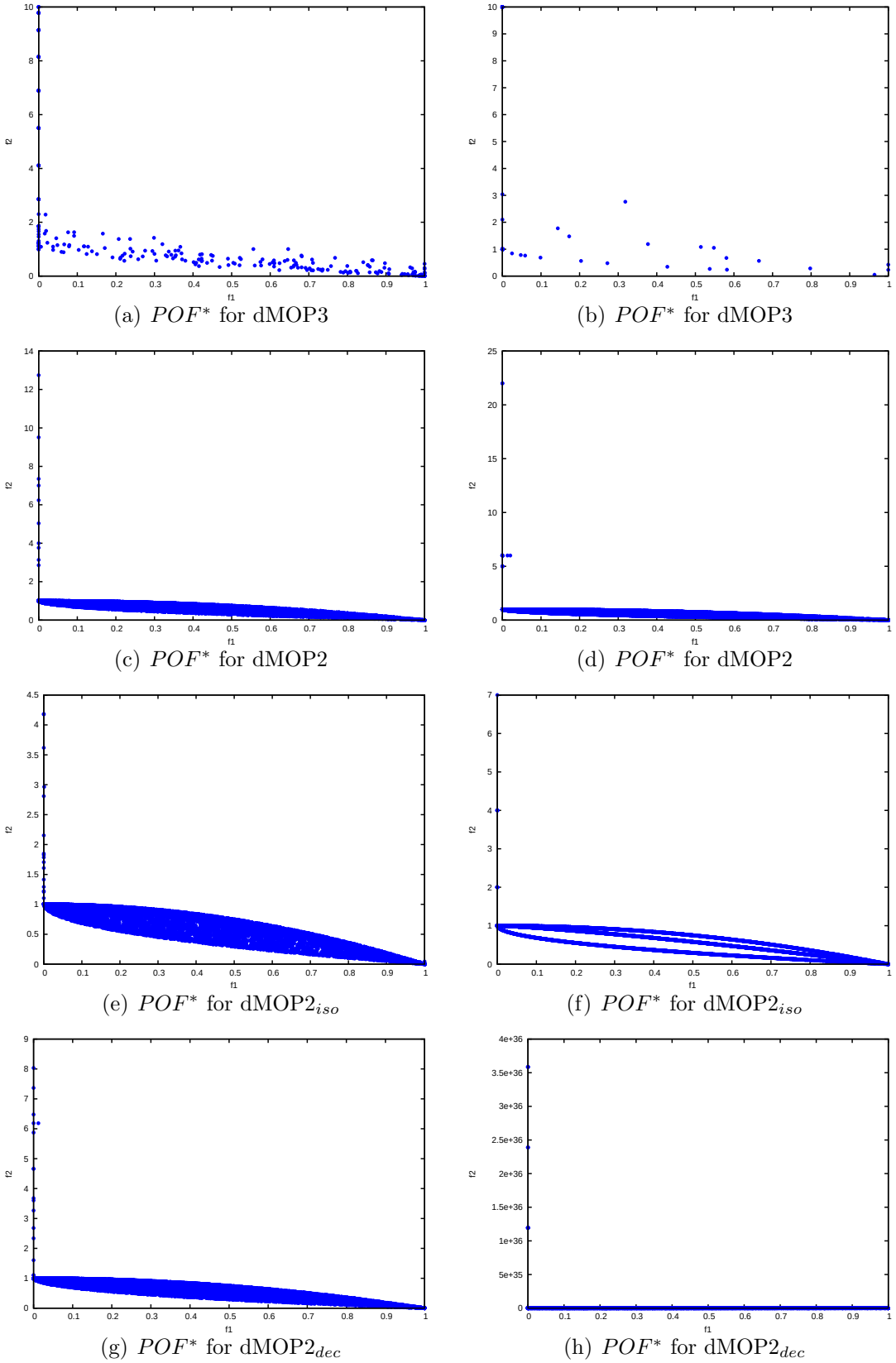
POFs found by DVEPSO using the p_s-g_r guide update approach during a single run for the HE DMOOPs are illustrated in Figures 9.8 and 9.9. Figure 9.8 indicates that DVEPSO struggled to converge to the discontinuous POFs of HE1 and HE2. When solving HE6 and HE7, DVEPSO found a POF^* that was close to POF. However, DVEPSO also found many solutions further away from POF as can be seen in Figure 9.9. When solving HE9, DVEPSO only found a few solutions and struggled to converge to the POF, as illustrated in Figures 9.9(e) and 9.9(f).

The original VEPSO algorithm's guide update approach, p_s-g_s, obtained the seventh overall rank. The other three p_s combinations obtained the top three overall ranks. Therefore, the results indicate that using Pareto-dominance information to update the guides, enhances the performance of DVEPSO.

The next section discusses general observations with regards to the performance of the various guide update approaches solving the various DMOOPs.

General Observations

It is interesting to note that p_s-g_r, which ranked the best of all guide update approaches, performed much better solving FDA2 than the modified FDA2 function, FDA2_{Camara}. The FDA2 DMOOP was adapted because the POF of the original DMOOP changes from convex to concave for only specific decision variable values (refer to Section 3.2.1). However, it should be noted that the results only indicate that relative to the other guide update approaches, p_s-g_r, performed better for FDA2 than for FDA2_{Camara}. Table 9.10 presents the wins and losses of the various guide update approaches for FDA2. When solving FDA2, p_s-g_r obtained the best performance with regards to acc over all n_t-τ_t, the tenth rank with regards to $stab$ and the seventh rank with regards to NS. This lead to an overall rank of four.



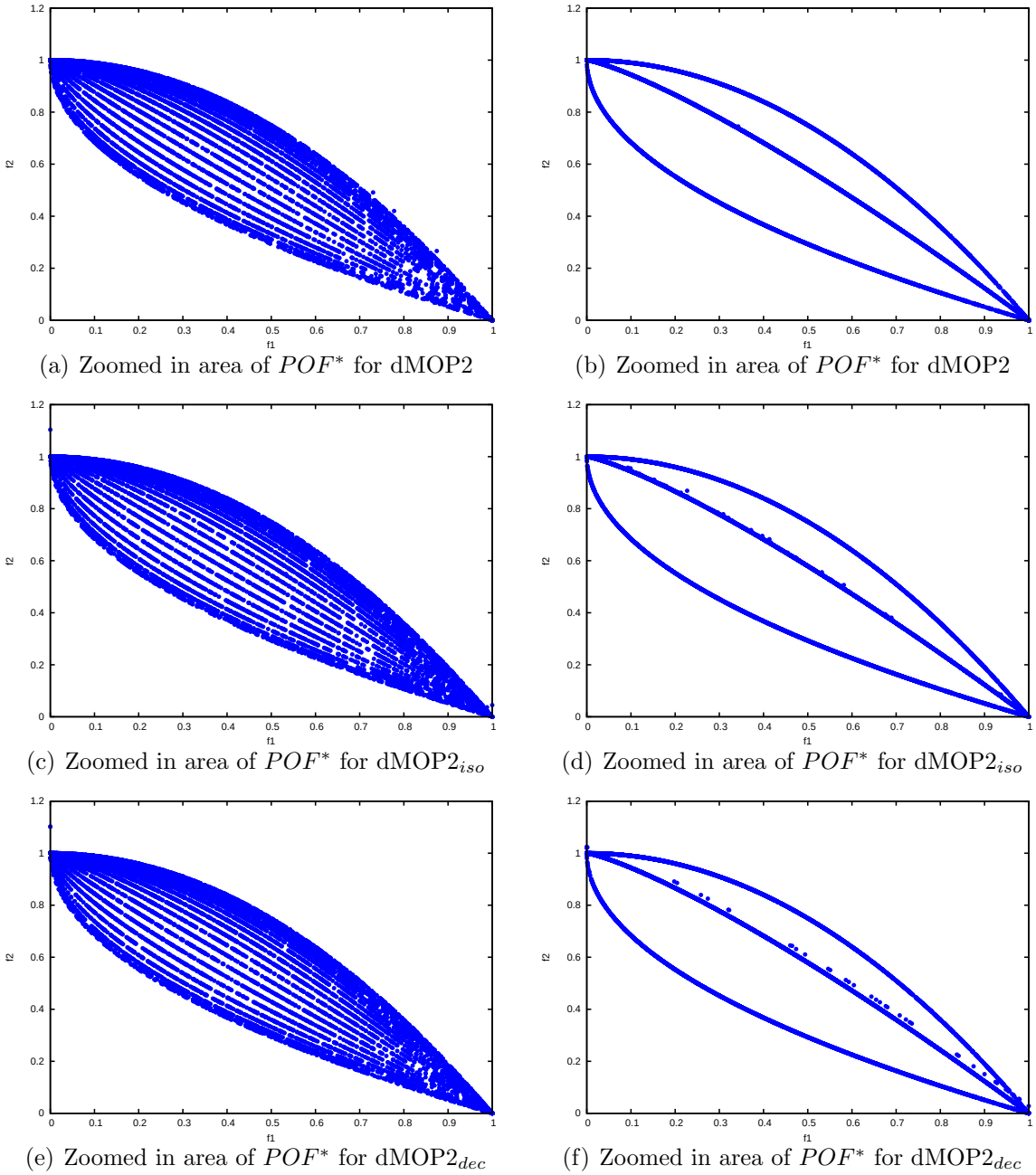
(a) POF^* for dMOP3 (b) POF^* for dMOP3

(c) POF^* for dMOP2 (d) POF^* for dMOP2

(e) POF^* for dMOP2$_{iso}$ (f) POF^* for dMOP2$_{iso}$

(g) POF^* for dMOP2$_{dec}$ (h) POF^* for dMOP2$_{dec}$

Figure 9.6: POF^* for dMOP functions of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right



(a) Zoomed in area of POF^* for dMOP2

(b) Zoomed in area of POF^* for dMOP2

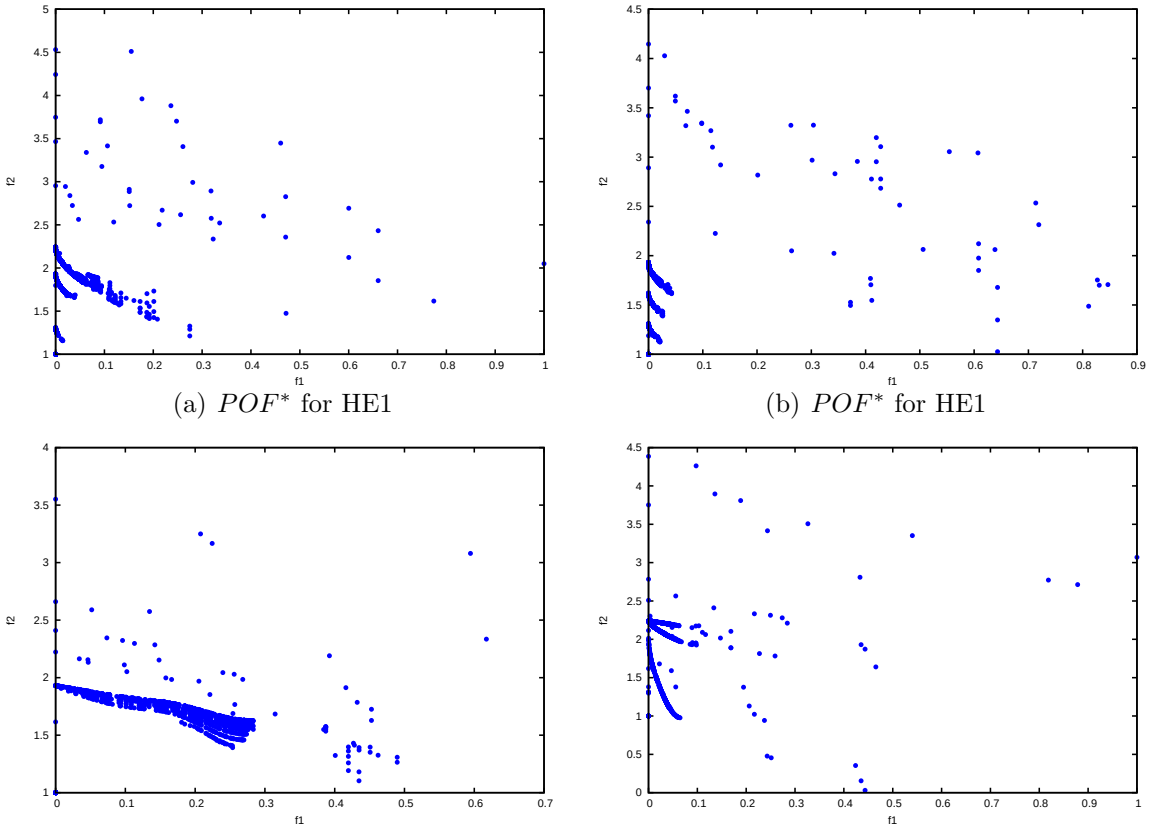
(c) Zoomed in area of POF^* for dMOP2$_{iso}$

(d) Zoomed in area of POF^* for dMOP2$_{iso}$

(e) Zoomed in area of POF^* for dMOP2$_{dec}$

(f) Zoomed in area of POF^* for dMOP2$_{dec}$

Figure 9.7: Zoomed in areas of POF^* for dMOP functions of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right



(a) POF^* for HE1 (b) POF^* for HE1

(c) POF^* for HE2 (d) POF^* for HE2

Figure 9.8: POF^* for HE1 and HE2 of DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right

Table 9.10: Wins and Losses of FDA2

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*acc*	Wins	0	0	1	12	4	9	9	3	3	9	3	12	3	10	3	12
10	10	*acc*	Losses	14	13	13	0	7	3	1	7	7	3	8	0	7	3	7	0
10	10	*acc*	Diff	-14	-13	-12	12	-3	6	8	-4	-4	6	-5	12	-4	7	-4	12
10	10	*acc*	Rank	16	15	14	**1**	8	6	4	9	9	6	13	**1**	9	5	9	**1**
10	25	*acc*	Wins	1	1	0	14	3	8	8	3	4	9	4	8	3	5	3	9
10	25	*acc*	Losses	13	13	15	0	9	1	1	6	6	1	2	1	7	2	6	0
10	25	*acc*	Diff	-12	-12	-15	14	-6	7	7	-3	-2	8	2	7	-4	3	-3	9
10	25	*acc*	Rank	14	14	16	**1**	13	4	4	10	9	3	8	4	12	7	10	2
10	50	*acc*	Wins	1	1	0	10	2	10	13	2	2	10	2	13	2	9	2	14
10	50	*acc*	Losses	13	7	15	3	7	3	0	7	7	3	7	1	7	6	7	0

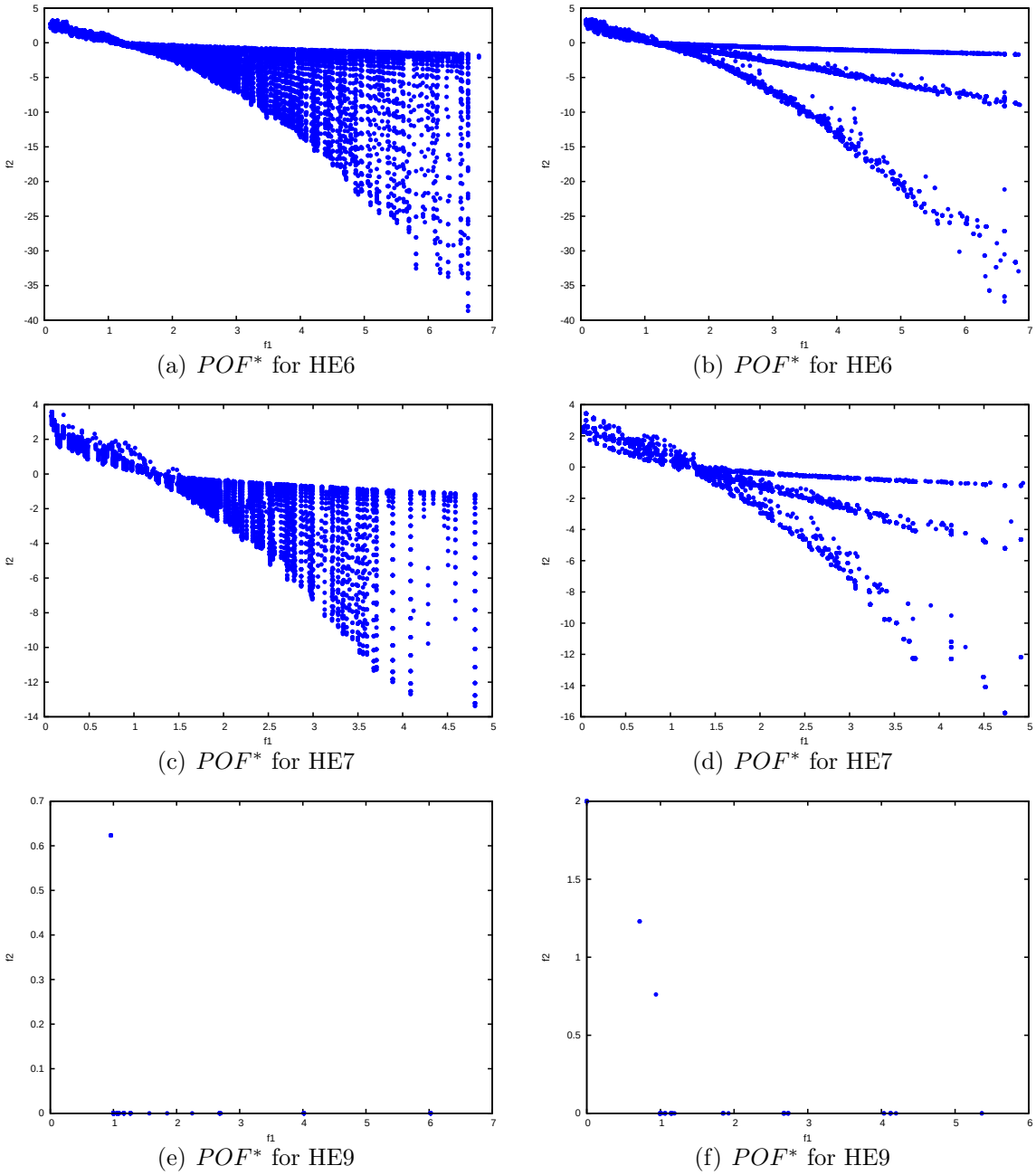
Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	50	*acc*	Diff	-12	-6	-15	7	-5	7	13	-5	-5	7	-5	12	-5	3	-5	14
10	50	*acc*	Rank	15	14	16	4	8	4	2	8	8	4	8	3	8	7	8	**1**
1	10	*acc*	Wins	2	0	1	15	6	10	5	5	5	12	4	5	5	12	3	5
1	10	*acc*	Losses	13	15	14	0	3	1	3	5	4	1	11	4	4	1	12	4
1	10	*acc*	Diff	-11	-15	-13	15	3	9	2	0	1	11	-7	1	1	11	-9	1
1	10	*acc*	Rank	14	16	15	**1**	5	4	6	11	7	2	12	7	7	2	13	7
20	10	*acc*	Wins	0	2	0	12	4	10	9	4	4	9	4	12	3	9	4	10
20	10	*acc*	Losses	14	13	14	0	7	2	0	7	7	2	7	0	12	3	7	1
20	10	*acc*	Diff	-14	-11	-14	12	-3	8	9	-3	-3	7	-3	12	-9	6	-3	9
20	10	*acc*	Rank	15	14	15	**1**	8	5	3	8	8	6	8	**1**	13	7	8	3
all	all	*acc*	Wins	4	4	2	63	19	47	44	17	18	49	17	50	16	45	15	50
all	all	*acc*	Losses	67	61	71	3	33	10	5	32	31	10	35	6	37	15	39	5
all	all	*acc*	Diff	-63	-57	-69	60	-14	37	39	-15	-13	39	-18	44	-21	30	-24	45
all	all	*acc*	Rank	15	14	16	**1**	9	6	4	10	8	4	11	3	12	7	13	2
10	10	*stab*	Wins	14	7	13	6	6	0	0	6	6	0	6	0	6	0	6	0
10	10	*stab*	Losses	0	1	0	2	2	10	10	2	2	10	2	10	3	10	2	10
10	10	*stab*	Diff	14	6	13	4	4	-10	-10	4	4	-10	4	-10	3	-10	4	-10
10	10	*stab*	Rank	**1**	3	2	4	4	11	11	4	4	11	4	11	10	11	4	11
10	25	*stab*	Wins	7	7	9	0	7	0	0	7	6	0	7	0	7	0	7	0
10	25	*stab*	Losses	0	0	0	9	0	9	9	0	1	8	0	9	0	9	1	9
10	25	*stab*	Diff	7	7	9	-9	7	-9	-9	7	5	-8	7	-9	7	-9	6	-9
10	25	*stab*	Rank	2	2	**1**	11	2	11	11	2	9	10	2	11	2	11	8	11
10	50	*stab*	Wins	7	7	7	0	7	0	0	7	7	0	7	0	7	4	7	0
10	50	*stab*	Losses	0	0	0	9	0	9	10	0	0	10	0	10	0	9	0	10
10	50	*stab*	Diff	7	7	7	-9	7	-9	-10	7	7	-10	7	-10	7	-5	7	-10
10	50	*stab*	Rank	**1**	**1**	**1**	11	**1**	11	13	**1**	**1**	13	**1**	13	**1**	10	**1**	13
1	10	*stab*	Wins	2	4	2	2	2	0	0	2	1	2	1	0	2	0	4	0
1	10	*stab*	Losses	1	0	1	0	0	10	1	0	0	0	0	0	0	10	0	1
1	10	*stab*	Diff	1	4	1	2	2	-10	-1	2	1	2	1	0	2	-10	4	-1
1	10	*stab*	Rank	8	**1**	8	3	3	15	13	3	8	3	8	12	3	15	**1**	13
20	10	*stab*	Wins	14	8	13	3	7	1	0	7	6	1	4	0	7	0	7	0
20	10	*stab*	Losses	0	1	0	7	2	8	12	2	2	8	3	10	2	9	2	10
20	10	*stab*	Diff	14	7	13	-4	5	-7	-12	5	4	-7	1	-10	5	-9	5	-10
20	10	*stab*	Rank	**1**	3	2	10	4	11	16	4	8	11	9	14	4	13	4	14
all	all	*stab*	Wins	44	33	44	11	29	1	0	29	26	3	25	0	29	4	31	0
all	all	*stab*	Losses	1	2	1	27	4	46	42	4	5	36	5	39	5	47	5	40
all	all	*stab*	Diff	43	31	43	-16	25	-45	-42	25	21	-33	20	-39	24	-43	26	-40
all	all	*stab*	Rank	**1**	3	**1**	10	5	16	14	5	8	11	9	12	7	15	4	13
10	10	*NS*	Wins	0	1	1	3	3	10	13	3	3	10	4	13	3	10	3	13
10	10	*NS*	Losses	14	14	13	7	6	3	0	6	6	3	6	0	6	3	6	0
10	10	*NS*	Diff	-14	-13	-12	-4	-3	7	13	-3	-3	7	-2	13	-3	7	-3	13
10	10	*NS*	Rank	16	15	14	13	8	4	**1**	8	8	4	7	**1**	8	4	8	**1**
10	25	*NS*	Wins	1	0	0	7	0	2	9	0	0	1	0	9	0	1	0	9
10	25	*NS*	Losses	5	4	7	0	5	0	0	4	3	0	3	0	4	0	4	0
10	25	*NS*	Diff	-4	-4	-7	7	-5	2	9	-4	-3	1	-3	9	-4	1	-4	9
10	25	*NS*	Rank	10	10	16	4	15	5	**1**	10	8	6	8	**1**	10	6	10	**1**
10	50	*NS*	Wins	0	0	0	10	2	10	13	0	4	10	0	13	0	9	2	14
10	50	*NS*	Losses	8	7	10	3	7	3	0	7	7	3	10	1	8	6	7	0
10	50	*NS*	Diff	-8	-7	-10	7	-5	7	13	-7	-3	7	-10	12	-8	3	-5	14
10	50	*NS*	Rank	13	11	15	4	9	4	2	11	8	4	15	3	13	7	9	**1**

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
1	10	NS	Wins	1	0	1	2	4	10	11	4	4	5	4	10	4	9	4	11
1	10	NS	Losses	13	15	12	12	5	0	0	5	4	2	5	0	5	0	6	0
1	10	NS	Diff	-12	-15	-11	-10	-1	10	11	-1	0	3	-1	10	-1	9	-2	11
1	10	NS	Rank	15	16	14	13	8	3	**1**	8	7	6	8	3	8	5	12	**1**
20	10	NS	Wins	0	0	0	6	1	8	13	4	3	8	3	13	2	10	2	13
20	10	NS	Losses	13	10	12	3	7	4	0	6	4	3	6	0	7	3	8	0
20	10	NS	Diff	-13	-10	-12	3	-6	4	13	-2	-1	5	-3	13	-5	7	-6	13
20	10	NS	Rank	16	14	15	7	12	6	**1**	9	8	5	10	**1**	11	4	12	**1**
all	all	NS	Wins	2	1	2	28	10	40	59	11	14	34	11	58	9	39	11	60
all	all	NS	Losses	53	50	54	25	30	10	0	28	24	11	30	1	30	12	31	0
all	all	NS	Diff	-51	-49	-52	3	-20	30	59	-17	-10	23	-19	57	-21	27	-20	60
all	all	NS	Rank	15	14	16	7	11	4	2	9	8	6	10	3	13	5	11	**1**
10	10	all	Wins	14	8	15	21	13	19	22	12	12	19	13	25	12	20	12	25
10	10	all	Losses	28	28	26	9	15	16	11	15	15	16	16	10	16	16	15	10
10	10	all	Diff	-14	-20	-11	12	-2	3	11	-3	-3	3	-3	15	-4	4	-3	15
10	10	all	Rank	15	16	14	3	8	6	4	9	9	6	9	**1**	13	5	9	**1**
10	25	all	Wins	9	8	9	21	10	10	17	10	10	10	11	17	10	6	10	18
10	25	all	Losses	18	17	22	9	14	10	10	10	10	9	5	10	11	11	11	9
10	25	all	Diff	-9	-9	-13	12	-4	0	7	0	0	1	6	7	-1	-5	-1	9
10	25	all	Rank	14	14	16	**1**	12	7	3	7	7	6	5	3	10	13	10	2
10	50	all	Wins	8	8	7	20	11	20	26	9	13	20	9	26	9	22	11	28
10	50	all	Losses	21	14	25	15	14	15	10	14	14	16	17	12	15	21	14	10
10	50	all	Diff	-13	-6	-18	5	-3	5	16	-5	-1	4	-8	14	-6	1	-3	18
10	50	all	Rank	15	12	16	4	9	4	2	11	8	6	14	3	12	7	9	**1**
1	10	all	Wins	5	4	4	19	12	20	16	11	10	19	9	15	11	21	11	16
1	10	all	Losses	27	30	27	12	8	11	4	10	8	3	16	4	9	11	18	5
1	10	all	Diff	-22	-26	-23	7	4	9	12	1	2	16	-7	11	2	10	-7	11
1	10	all	Rank	14	16	15	7	8	6	2	11	9	**1**	12	3	9	5	12	3
20	10	all	Wins	14	10	13	21	12	19	22	15	13	18	11	25	12	19	13	23
20	10	all	Losses	27	24	26	10	16	14	12	15	13	13	16	10	21	15	17	11
20	10	all	Diff	-13	-14	-13	11	-4	5	10	0	0	5	-5	15	-9	4	-4	12
20	10	all	Rank	14	16	14	3	10	5	4	8	8	5	12	**1**	13	7	10	2
all	all	all	Wins	50	38	48	102	58	88	103	57	58	86	53	108	54	88	57	110
all	all	all	Losses	121	113	126	55	67	66	47	64	60	57	70	46	72	74	75	45
all	all	all	Diff	-71	-75	-78	47	-9	22	56	-7	-2	29	-17	62	-18	14	-18	65
all	all	all	Rank	14	15	16	4	10	6	3	9	8	5	11	2	12	7	12	**1**

The wins and losses of the various guide update approaches for $FDA2_{Camara}$ are presented in Table 9.11. When solving $FDA2_{Camara}$, there was no statistical significant difference between the performance of the various guide update approaches with regards to *acc* for $n_t = 10$ and $\tau_t = 10$, $n_t = 10$ and $n_t = 25$, $n_t = 10$ and $\tau_t = 50$, and $n_t = 20$ and $\tau_t = 10$. With regards to *stab*, there was no statistical significant difference for $n_t = 10$



(a) POF^* for HE6 (b) POF^* for HE6

(c) POF^* for HE7 (d) POF^* for HE7

(e) POF^* for HE9 (f) POF^* for HE9

Figure 9.9: POF^* of HE6, HE7 and HE9 DVEPSO using p_s-g_r for $n_t = 10$ and $\tau_t = 10$ on the left and for $n_t = 1$ and $\tau_t = 10$ on the right

and $\tau_t = 10$, and $n_t = 10$ and $n_t = 25$. For $n_t = 10$ and $\tau_t = 25$, and $n_t = 20$ and $\tau_t = 10$, there was no statistical significant difference between the performance of the various guide update approaches with regards to NS. Over all performance measures for $n_t = 10$ and $\tau_t = 25$, there was no statistical significant difference between the performance of the various guide update approaches. Therefore, even though p_s-g_r performed quite poor for FDA2$_{Camara}$, there was no statistical significant difference between the performance of the various guide update approaches when solving FDA2$_{Camara}$. A similar trend was observed for FDA3 and FDA3$_{Camara}$.

Table 9.11: Wins and Losses of FDA2$_{Camara}$

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	**s-n**	**s-d**	**s-r**	**n-s**	**n-n**	**n-d**	**n-r**	**r-s**	**r-n**	**r-d**	**r-r**	**d-s**	**d-n**	**d-d**	**d-r**
10	10	*acc*	Wins	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
10	10	*acc*	Losses	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*acc*	Diff	0	-13	0	1	1	1	1	1	1	1	1	1	1	1	1	1
10	10	*acc*	Rank	14	16	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	10	*acc*	Wins	0	0	0	2	3	3	3	3	3	3	3	3	3	3	3	3
1	10	*acc*	Losses	13	13	12	0	0	0	0	0	0	0	0	0	0	0	0	0
1	10	*acc*	Diff	-13	-13	-12	2	3	3	3	3	3	3	3	3	3	3	3	3
1	10	*acc*	Rank	15	15	14	13	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
all	all	*acc*	Wins	0	0	0	3	4	4	4	4	4	4	4	4	4	4	4	4
all	all	*acc*	Losses	13	26	12	0	0	0	0	0	0	0	0	0	0	0	0	0
all	all	*acc*	Diff	-13	-26	-12	3	4	4	4	4	4	4	4	4	4	4	4	4
all	all	*acc*	Rank	15	16	14	13	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	10	*stab*	Wins	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*stab*	Losses	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*stab*	Diff	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*stab*	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	25	*stab*	Wins	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	25	*stab*	Losses	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	25	*stab*	Diff	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	25	*stab*	Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	50	*stab*	Wins	0	0	1	0	1	4	4	3	3	1	0	0	3	1	0	0
10	50	*stab*	Losses	9	5	0	5	0	0	0	0	0	0	2	0	0	0	0	0
10	50	*stab*	Diff	-9	-5	1	-5	1	4	4	3	3	1	-2	0	3	1	0	0
10	50	*stab*	Rank	16	14	6	14	6	**1**	**1**	3	3	6	13	10	3	6	10	10
1	10	*stab*	Wins	0	0	0	2	3	3	3	3	3	3	3	3	3	3	3	3
1	10	*stab*	Losses	13	13	12	0	0	0	0	0	0	0	0	0	0	0	0	0
1	10	*stab*	Diff	-13	-13	-12	2	3	3	3	3	3	3	3	3	3	3	3	3
1	10	*stab*	Rank	15	15	14	13	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
20	10	*stab*	Wins	0	0	0	0	0	1	0	0	0	2	0	0	0	2	1	0
20	10	*stab*	Losses	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	10	*stab*	Diff	-4	-2	0	0	0	1	0	0	0	2	0	0	0	2	1	0
20	10	*stab*	Rank	16	15	5	5	5	3	5	5	5	**1**	5	5	5	**1**	3	5
all	all	*stab*	Wins	0	0	1	2	4	8	7	6	6	6	3	3	6	6	4	3
all	all	*stab*	Losses	26	20	12	5	0	0	0	0	0	0	2	0	0	0	0	0

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
all	all	*stab*	Diff	-26	-20	-11	-3	4	8	7	6	6	6	1	3	6	6	4	3
all	all	*stab*	Rank	16	15	14	13	8	**1**	2	3	3	3	12	10	3	3	8	10
10	10	*NS*	Wins	0	0	0	0	2	1	2	2	2	2	0	1	2	1	2	2
10	10	*NS*	Losses	0	8	0	11	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*NS*	Diff	0	-8	0	-11	2	1	2	2	2	2	0	1	2	1	2	2
10	10	*NS*	Rank	12	15	12	16	**1**	9	**1**	**1**	**1**	**1**	12	9	**1**	9	**1**	**1**
10	50	*NS*	Wins	0	0	0	10	2	10	13	0	4	10	0	13	0	9	2	14
10	50	*NS*	Losses	8	7	10	3	7	3	0	7	7	3	10	1	8	6	7	0
10	50	*NS*	Diff	-8	-7	-10	7	-5	7	13	-7	-3	7	-10	12	-8	3	-5	14
10	50	*NS*	Rank	13	11	15	4	9	4	2	11	8	4	15	3	13	7	9	**1**
1	10	*NS*	Wins	0	2	0	0	4	4	4	4	4	4	4	4	4	4	4	4
1	10	*NS*	Losses	12	12	13	13	0	0	0	0	0	0	0	0	0	0	0	0
1	10	*NS*	Diff	-12	-10	-13	-13	4	4	4	4	4	4	4	4	4	4	4	4
1	10	*NS*	Rank	14	13	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
all	all	*NS*	Wins	0	2	0	10	8	15	19	6	10	16	4	18	6	14	8	20
all	all	*NS*	Losses	20	27	23	27	7	3	0	7	7	3	10	1	8	6	7	0
all	all	*NS*	Diff	-20	-25	-23	-17	1	12	19	-1	3	13	-6	17	-2	8	1	20
all	all	*NS*	Rank	14	16	15	13	8	5	2	10	7	4	12	3	11	6	8	**1**
10	10	all	Wins	0	0	0	1	3	2	3	3	3	3	1	2	3	2	3	3
10	10	all	Losses	0	21	0	11	0	0	0	0	0	0	0	0	0	0	0	0
10	10	all	Diff	0	-21	0	-10	3	2	3	3	3	3	1	2	3	2	3	3
10	10	all	Rank	13	16	13	15	**1**	9	**1**	**1**	**1**	**1**	12	9	**1**	9	**1**	**1**
10	50	all	Wins	0	0	1	10	3	14	17	3	7	11	0	13	3	10	2	14
10	50	all	Losses	17	12	10	8	7	3	0	7	7	3	12	1	8	6	7	0
10	50	all	Diff	-17	-12	-9	2	-4	11	17	-4	0	8	-12	12	-5	4	-5	14
10	50	all	Rank	16	14	13	7	9	4	**1**	9	8	5	14	3	11	6	11	2
1	10	all	Wins	0	2	0	4	10	10	10	10	10	10	10	10	10	10	10	10
1	10	all	Losses	38	38	37	13	0	0	0	0	0	0	0	0	0	0	0	0
1	10	all	Diff	-38	-36	-37	-9	10	10	10	10	10	10	10	10	10	10	10	10
1	10	all	Rank	16	14	15	13	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
20	10	all	Wins	0	0	0	0	0	1	0	0	0	2	0	0	0	2	1	0
20	10	all	Losses	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	10	all	Diff	-4	-2	0	0	0	1	0	0	0	2	0	0	0	2	1	0
20	10	all	Rank	16	15	5	5	5	3	5	5	5	**1**	5	5	5	**1**	3	5
all	all	all	Wins	0	2	1	15	16	27	30	16	20	26	11	25	16	24	16	27
all	all	all	Losses	59	73	47	32	7	3	0	7	7	3	12	1	8	6	7	0
all	all	all	Diff	-59	-71	-46	-17	9	24	30	9	13	23	-1	24	8	18	9	27
all	all	all	Rank	15	16	14	13	8	3	**1**	8	7	5	12	3	11	6	8	2

The average performance measure values at each iteration just before a change in the environment occurred obtained by DVEPSO using either p_s-g_s or p_s-g_r guide update approaches, are illustrated in Figures 9.10 to 9.12. In Figures 9.10 to 9.12 the values obtained by p_s-g_s and p_s-g_r are illustrated with a magenta triangle and blue circle re-

spectively. The wins and losses of Table 9.11 are calculated based on these performance measure values. Similar figures for the other DMOOPs can be found in Appendix D.

Figure 9.10 shows that p_s-g_r outperformed p_s-g_s with regards to *acc* for all n_t-τ_t combinations. This is confirmed in Table 9.11 where p_s-g_r obtained the highest rank of all guide update approaches for the wins and losses with regards to *acc*.

Figure 9.11 indicates that p_s-g_s outperformed p_s-g_r with regards to *stab* for all n_t-τ_t combinations. Table 9.11 confirms this observation, since p_s-g_s obtained the highest rank of all guide update approaches for the wins and losses with regards to *stab*.

Figure 9.12 shows that p_s-g_r outperformed p_s-g_s with regards to NS for all n_t-τ_t combinations. This is confirmed in Table 9.11 where p_s-g_r obtained a higher rank than p_s-g_r for the wins and losses with regards to NS.

When solving DMOOPs with discontinuous POFs, the p_s combinations outperformed the other guide update approaches. Table 9.12 presents the wins and losses for HE1 of the various guide update approaches. A similar trend was observed for HE2.

Table 9.12: Wins and Losses of HE1

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*acc*	Wins	15	12	14	13	6	1	3	9	3	6	8	1	8	0	8	2
10	10	*acc*	Losses	0	3	1	2	8	13	10	4	10	8	5	12	4	15	4	10
10	10	*acc*	Diff	15	9	13	11	-2	-12	-7	5	-7	-2	3	-11	4	-15	4	-8
10	10	*acc*	Rank	**1**	4	2	3	9	15	11	5	11	9	8	14	6	16	6	13
10	25	*acc*	Wins	14	12	14	13	4	1	3	7	6	0	7	11	2	2	2	7
10	25	*acc*	Losses	0	3	0	2	8	14	9	5	5	15	5	4	9	11	10	5
10	25	*acc*	Diff	14	9	14	11	-4	-13	-6	2	1	-15	2	7	-7	-9	-8	2
10	25	*acc*	Rank	**1**	4	**1**	3	10	15	11	6	9	16	6	5	12	14	13	6
10	50	*acc*	Wins	14	12	13	14	5	0	0	4	5	1	1	0	11	0	3	6
10	50	*acc*	Losses	0	3	2	0	5	12	10	5	5	8	6	9	4	10	5	5
10	50	*acc*	Diff	14	9	11	14	0	-12	-10	-1	0	-7	-5	-9	7	-10	-2	1
10	50	*acc*	Rank	**1**	4	3	**1**	7	16	14	9	7	12	11	13	5	14	10	6
1	10	*acc*	Wins	14	12	15	13	7	0	6	4	7	0	5	3	5	0	4	6
1	10	*acc*	Losses	1	3	0	2	4	13	4	8	4	13	6	12	4	13	10	4
1	10	*acc*	Diff	13	9	15	11	3	-13	2	-4	3	-13	-1	-9	1	-13	-6	2
1	10	*acc*	Rank	2	4	**1**	3	5	14	7	11	5	14	10	13	9	14	12	7
20	10	*acc*	Wins	14	12	14	13	7	3	4	9	4	0	9	1	7	0	9	3
20	10	*acc*	Losses	0	3	0	2	7	11	9	4	9	13	4	13	7	14	4	9
20	10	*acc*	Diff	14	9	14	11	0	-8	-5	5	-5	-13	5	-12	0	-14	5	-6
20	10	*acc*	Rank	**1**	4	**1**	3	8	13	10	5	10	15	5	14	8	16	5	12
all	all	*acc*	Wins	71	60	70	66	29	5	16	33	25	7	30	16	33	2	26	24
all	all	*acc*	Losses	1	15	3	8	32	63	42	26	33	57	26	50	28	63	33	33
all	all	*acc*	Diff	70	45	67	58	-3	-58	-26	7	-8	-50	4	-34	5	-61	-7	-9
all	all	*acc*	Rank	**1**	4	2	3	8	15	12	5	10	14	7	13	6	16	9	11

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*stab*	Wins	12	12	12	13	0	1	0	0	0	0	1	0	0	1	0	0
10	10	*stab*	Losses	0	1	0	0	4	4	4	7	4	4	4	4	4	4	4	4
10	10	*stab*	Diff	12	11	12	13	-4	-3	-4	-7	-4	-4	-3	-4	-4	-3	-4	-4
10	10	*stab*	Rank	2	4	2	**1**	8	5	8	16	8	8	5	8	8	5	8	8
10	25	*stab*	Wins	12	12	12	12	1	0	1	0	0	1	0	0	0	1	0	0
10	25	*stab*	Losses	0	0	0	0	4	4	4	8	4	4	4	4	4	4	4	4
10	25	*stab*	Diff	12	12	12	12	-3	-4	-3	-8	-4	-3	-4	-4	-4	-3	-4	-4
10	25	*stab*	Rank	**1**	**1**	**1**	**1**	5	9	5	16	9	5	9	9	9	5	9	9
10	50	*stab*	Wins	12	12	12	12	0	0	0	0	0	0	0	0	0	0	0	0
10	50	*stab*	Losses	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4	4
10	50	*stab*	Diff	12	12	12	12	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
10	50	*stab*	Rank	**1**	**1**	**1**	**1**	5	5	5	5	5	5	5	5	5	5	5	5
1	10	*stab*	Wins	15	12	13	13	0	0	3	0	0	1	0	3	1	0	1	1
1	10	*stab*	Losses	0	3	1	1	6	4	4	10	6	4	4	4	4	4	4	4
1	10	*stab*	Diff	15	9	12	12	-6	-4	-1	-10	-6	-3	-4	-1	-3	-4	-3	-3
1	10	*stab*	Rank	**1**	4	2	2	14	11	5	16	14	7	11	5	7	11	7	7
20	10	*stab*	Wins	12	12	12	12	0	0	0	0	0	0	0	0	0	0	0	0
20	10	*stab*	Losses	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4	4
20	10	*stab*	Diff	12	12	12	12	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4
20	10	*stab*	Rank	**1**	**1**	**1**	**1**	5	5	5	5	5	5	5	5	5	5	5	5
all	all	*stab*	Wins	63	60	61	62	1	1	4	0	0	2	1	3	1	2	1	1
all	all	*stab*	Losses	0	4	1	1	22	20	20	33	22	20	20	20	20	20	20	20
all	all	*stab*	Diff	63	56	60	61	-21	-19	-16	-33	-22	-18	-19	-17	-19	-18	-19	-19
all	all	*stab*	Rank	**1**	4	3	2	14	9	5	16	15	7	9	6	9	7	9	9
10	10	*NS*	Wins	15	13	14	12	1	1	8	0	1	3	4	9	0	8	0	0
10	10	*NS*	Losses	0	2	1	3	8	7	4	12	7	7	5	4	7	4	9	9
10	10	*NS*	Diff	15	11	13	9	-7	-6	4	-12	-6	-4	-1	5	-7	4	-9	-9
10	10	*NS*	Rank	**1**	3	2	4	12	10	6	16	10	9	8	5	12	6	14	14
10	25	*NS*	Wins	13	15	13	12	0	2	2	2	0	6	0	0	5	4	3	0
10	25	*NS*	Losses	1	0	2	2	6	4	4	6	6	4	8	11	4	4	4	11
10	25	*NS*	Diff	12	15	11	10	-6	-2	-2	-4	-6	2	-8	-11	1	0	-1	-11
10	25	*NS*	Rank	2	**1**	3	4	12	9	9	11	12	5	14	15	6	7	8	15
10	50	*NS*	Wins	14	12	12	15	0	5	0	0	0	4	0	0	0	1	0	0
10	50	*NS*	Losses	1	2	2	0	4	4	6	4	7	4	4	5	6	4	4	6
10	50	*NS*	Diff	13	10	10	15	-4	1	-6	-4	-7	0	-4	-5	-6	-3	-4	-6
10	50	*NS*	Rank	2	3	3	**1**	8	5	13	8	16	6	8	12	13	7	8	13
1	10	*NS*	Wins	13	13	15	12	1	9	4	3	0	8	3	4	1	9	3	1
1	10	*NS*	Losses	1	1	0	3	12	4	7	7	15	4	7	6	9	4	7	12
1	10	*NS*	Diff	12	12	15	9	-11	5	-3	-4	-15	4	-4	-2	-8	5	-4	-11
1	10	*NS*	Rank	2	2	**1**	4	14	5	9	10	16	7	10	8	13	5	10	14
20	10	*NS*	Wins	15	13	14	12	3	6	0	0	8	7	2	3	3	8	0	1
20	10	*NS*	Losses	0	2	1	3	8	4	13	12	4	4	8	7	6	4	11	8
20	10	*NS*	Diff	15	11	13	9	-5	2	-13	-12	4	3	-6	-4	-3	4	-11	-7
20	10	*NS*	Rank	**1**	3	2	4	11	8	16	15	5	7	12	10	9	5	14	13
all	all	*NS*	Wins	70	66	68	63	5	23	14	5	9	28	9	16	9	30	6	2
all	all	*NS*	Losses	3	7	6	11	38	23	34	41	39	23	32	33	32	20	35	46
all	all	*NS*	Diff	67	59	62	52	-33	0	-20	-36	-30	5	-23	-17	-23	10	-29	-44
all	all	*NS*	Rank	**1**	3	2	4	14	7	9	15	13	6	10	8	10	5	12	16
10	10	all	Wins	42	37	40	38	7	3	11	9	4	9	13	10	8	9	8	2
10	10	all	Losses	0	6	2	5	20	24	18	23	21	19	14	20	15	23	17	23

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	all	Diff	42	31	38	33	-13	-21	-7	-14	-17	-10	-1	-10	-7	-14	-9	-21
10	10	all	Rank	**1**	4	2	3	11	15	6	12	14	9	5	9	6	12	8	15
10	25	all	Wins	39	39	39	37	5	3	6	9	6	7	7	11	7	7	5	7
10	25	all	Losses	1	3	2	4	18	22	17	19	15	23	17	19	17	19	18	20
10	25	all	Diff	38	36	37	33	-13	-19	-11	-10	-9	-16	-10	-8	-10	-12	-13	-13
10	25	all	Rank	**1**	3	2	4	12	16	10	7	6	15	7	5	7	11	12	12
10	50	all	Wins	40	36	37	41	5	5	0	4	5	5	1	0	11	1	3	6
10	50	all	Losses	1	5	4	0	13	20	20	13	16	16	14	18	14	18	13	15
10	50	all	Diff	39	31	33	41	-8	-15	-20	-9	-11	-11	-13	-18	-3	-17	-10	-9
10	50	all	Rank	2	4	3	**1**	6	13	16	7	10	10	12	15	5	14	9	7
1	10	all	Wins	42	37	43	38	8	9	13	7	7	9	8	10	7	9	8	8
1	10	all	Losses	2	7	1	6	22	21	15	25	25	21	17	22	17	21	21	20
1	10	all	Diff	40	30	42	32	-14	-12	-2	-18	-18	-12	-9	-12	-10	-12	-13	-12
1	10	all	Rank	2	4	**1**	3	14	8	5	15	15	8	6	8	7	8	13	8
20	10	all	Wins	41	37	40	37	10	9	4	9	12	7	11	4	10	8	9	4
20	10	all	Losses	0	5	1	5	19	19	26	20	17	21	16	24	17	22	19	21
20	10	all	Diff	41	32	39	32	-9	-10	-22	-11	-5	-14	-5	-20	-7	-14	-10	-17
20	10	all	Rank	**1**	3	2	3	8	9	16	11	5	12	5	15	7	12	9	14
all	all	all	Wins	204	186	199	191	35	29	34	38	34	37	40	35	43	34	33	27
all	all	all	Losses	4	26	10	20	92	106	96	100	94	100	78	103	80	103	88	99
all	all	all	Diff	200	160	189	171	-57	-77	-62	-62	-60	-63	-38	-68	-37	-69	-55	-72
all	all	all	Rank	**1**	4	2	3	8	16	10	10	9	12	6	13	5	14	7	15

When solving DMOOPs where each decision variable has its own POS and the POS is a non-linear function, the p_s combinations outperformed the other guide update approaches. This was observed for HE6, HE7 and HE9. The only exception is with p_s-g_s, that performed poorly for HE7. The wins and losses obtained by the various guide update approaches for HE7 is presented in Table 9.13.

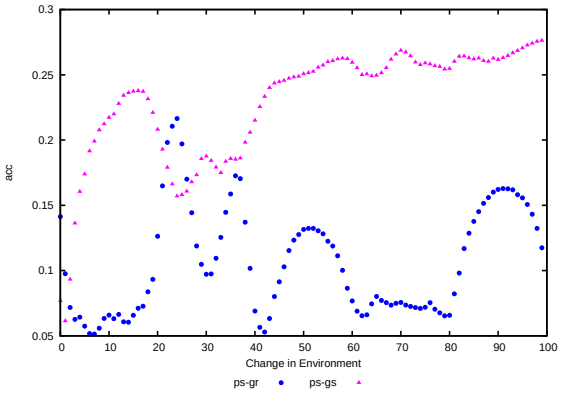
Table 9.13: Wins and Losses of HE7

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
1	10	*acc*	Wins	10	0	0	0	12	0	0	0	12	0	0	0	12	0	0	0
1	10	*acc*	Losses	0	4	4	4	0	4	4	3	0	4	4	4	0	4	4	3
1	10	*acc*	Diff	10	-4	-4	-4	12	-4	-4	-3	12	-4	-4	-4	12	-4	-4	-3
1	10	*acc*	Rank	4	7	7	7	**1**	7	7	5	**1**	7	7	7	**1**	7	7	5
all	all	*acc*	Wins	10	0	0	0	12	0	0	0	12	0	0	0	12	0	0	0
all	all	*acc*	Losses	0	4	4	4	0	4	4	3	0	4	4	4	0	4	4	3
all	all	*acc*	Diff	10	-4	-4	-4	12	-4	-4	-3	12	-4	-4	-4	12	-4	-4	-3
all	all	*acc*	Rank	4	7	7	7	**1**	7	7	5	**1**	7	7	7	**1**	7	7	5
10	10	*NS*	Wins	3	13	13	13	0	4	4	4	0	4	4	5	0	4	4	4
10	10	*NS*	Losses	12	0	0	0	13	4	3	3	13	3	3	3	13	3	3	3
10	10	*NS*	Diff	-9	13	13	13	-13	0	1	1	-13	1	1	2	-13	1	1	1
10	10	*NS*	Rank	13	**1**	**1**	**1**	14	12	5	5	14	5	5	4	14	5	5	5
10	25	*NS*	Wins	1	7	5	7	0	4	4	4	0	4	4	3	0	3	4	3
10	25	*NS*	Losses	9	0	0	0	12	0	0	0	12	0	0	3	13	2	0	2

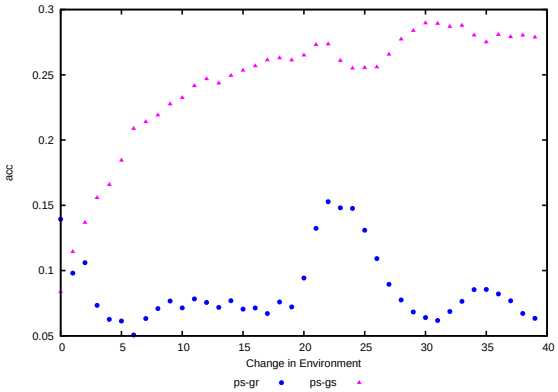
Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	25	NS	Diff	-8	7	5	7	-12	4	4	4	-12	4	4	0	-13	1	4	1
10	25	NS	Rank	13	**1**	3	**1**	14	4	4	4	14	4	4	12	16	10	4	10
10	50	NS	Wins	3	8	5	9	0	3	3	3	0	4	4	3	0	3	3	3
10	50	NS	Losses	5	0	0	0	13	0	0	3	13	0	0	2	13	2	2	1
10	50	NS	Diff	-2	8	5	9	-13	3	3	0	-13	4	4	1	-13	1	1	2
10	50	NS	Rank	13	2	3	**1**	14	6	6	12	14	4	4	9	14	9	9	8
1	10	NS	Wins	3	13	13	13	0	4	4	4	0	4	4	4	0	4	4	4
1	10	NS	Losses	12	0	0	0	13	3	3	3	13	3	3	3	13	3	3	3
1	10	NS	Diff	-9	13	13	13	-13	1	1	1	-13	1	1	1	-13	1	1	1
1	10	NS	Rank	13	**1**	**1**	**1**	14	4	4	4	14	4	4	4	14	4	4	4
20	10	NS	Wins	3	13	13	13	0	4	4	4	0	4	4	4	1	4	4	4
20	10	NS	Losses	12	0	0	0	14	3	3	3	13	3	3	3	13	3	3	3
20	10	NS	Diff	-9	13	13	13	-14	1	1	1	-13	1	1	1	-12	1	1	1
20	10	NS	Rank	13	**1**	**1**	**1**	16	4	4	4	15	4	4	4	14	4	4	4
all	all	NS	Wins	13	54	49	55	0	19	19	19	0	20	20	19	1	18	19	18
all	all	NS	Losses	50	0	0	0	65	10	9	12	64	9	9	14	65	13	11	12
all	all	NS	Diff	-37	54	49	55	-65	9	10	7	-64	11	11	5	-64	5	8	6
all	all	NS	Rank	13	2	3	**1**	16	7	6	9	14	4	4	11	14	11	8	10
10	10	all	Wins	3	13	13	13	0	4	4	4	0	4	4	5	0	4	4	4
10	10	all	Losses	12	0	0	0	13	4	3	3	13	3	3	3	13	3	3	3
10	10	all	Diff	-9	13	13	13	-13	0	1	1	-13	1	1	2	-13	1	1	1
10	10	all	Rank	13	**1**	**1**	**1**	14	12	5	5	14	5	5	4	14	5	5	5
10	25	all	Wins	1	7	5	7	0	4	4	4	0	4	4	3	0	3	4	3
10	25	all	Losses	9	0	0	0	12	0	0	0	12	0	0	3	13	2	0	2
10	25	all	Diff	-8	7	5	7	-12	4	4	4	-12	4	4	0	-13	1	4	1
10	25	all	Rank	13	**1**	3	**1**	14	4	4	4	14	4	4	12	16	10	4	10
10	50	all	Wins	3	8	5	9	0	3	3	3	0	4	4	3	0	3	3	3
10	50	all	Losses	5	0	0	0	13	0	0	3	13	0	0	2	13	2	2	1
10	50	all	Diff	-2	8	5	9	-13	3	3	0	-13	4	4	1	-13	1	1	2
10	50	all	Rank	13	2	3	**1**	14	6	6	12	14	4	4	9	14	9	9	8
1	10	all	Wins	13	13	13	13	12	4	4	4	12	4	4	4	12	4	4	4
1	10	all	Losses	12	4	4	4	13	7	7	6	13	7	7	7	13	7	7	6
1	10	all	Diff	1	9	9	9	-1	-3	-3	-2	-1	-3	-3	-3	-1	-3	-3	-2
1	10	all	Rank	4	**1**	**1**	**1**	5	10	10	8	5	10	10	10	5	10	10	8
20	10	all	Wins	3	13	13	13	0	4	4	4	0	4	4	4	1	4	4	4
20	10	all	Losses	12	0	0	0	14	3	3	3	13	3	3	3	13	3	3	3
20	10	all	Diff	-9	13	13	13	-14	1	1	1	-13	1	1	1	-12	1	1	1
20	10	all	Rank	13	**1**	**1**	**1**	16	4	4	4	15	4	4	4	14	4	4	4
all	all	all	Wins	23	54	49	55	12	19	19	19	12	20	20	19	13	18	19	18
all	all	all	Losses	50	4	4	4	65	14	13	15	64	13	13	18	65	17	15	15
all	all	all	Diff	-27	50	45	51	-53	5	6	4	-52	7	7	1	-52	1	4	3
all	all	all	Rank	13	2	3	**1**	16	7	6	8	14	4	4	11	14	11	8	10

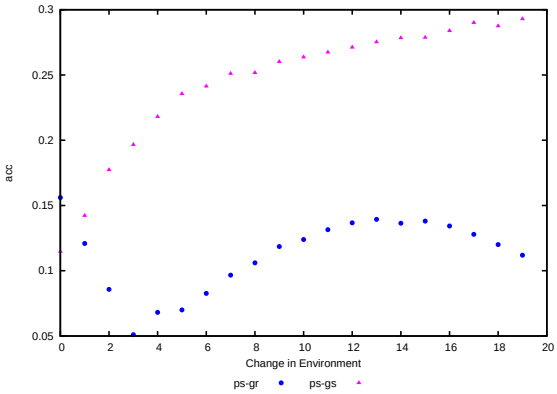
For dMOP2 it was interesting to observe the difference in performance of the p_s combinations when solving dMOP2, dMOP2 with an isolated POF, dMOP2$_{iso}$, and



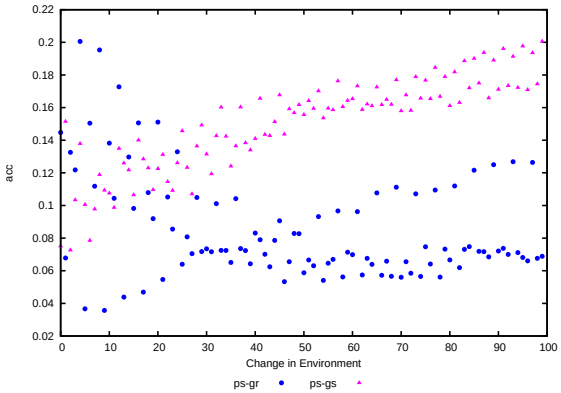
(a) *acc* values for $n_t = 10$ and $\tau_t = 10$



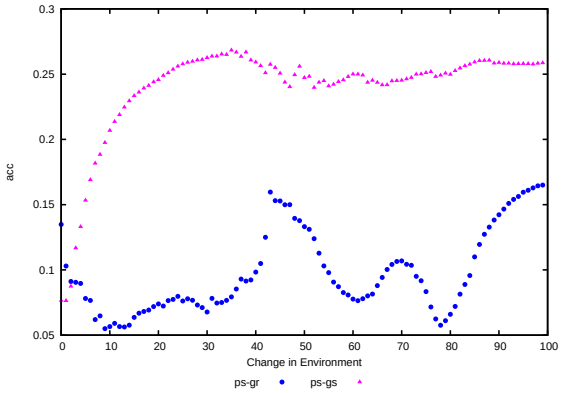
(b) *acc* values for $n_t = 10$ and $\tau_t = 25$



(c) *acc* values for $n_t = 10$ and $\tau_t = 50$

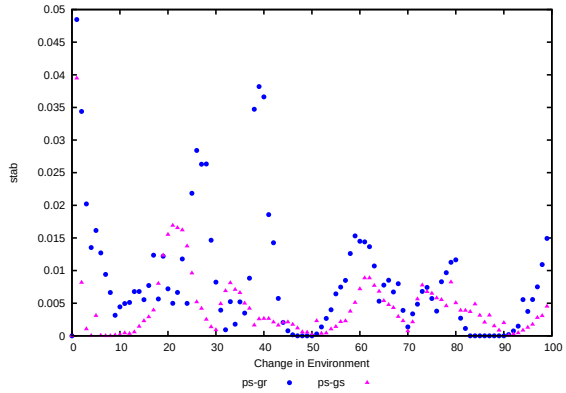


(d) *acc* values for $n_t = 1$ and $\tau_t = 10$

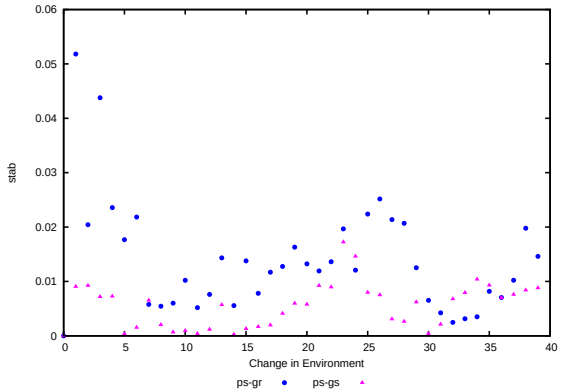


(e) *acc* values for $n_t = 20$ and $\tau_t = 10$

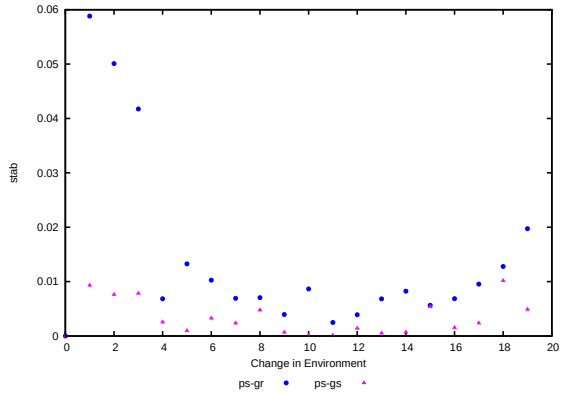
Figure 9.10: Average values of *acc* obtained by DVEPSO using either p_s-g_s or p_s-g_r solving FDA2



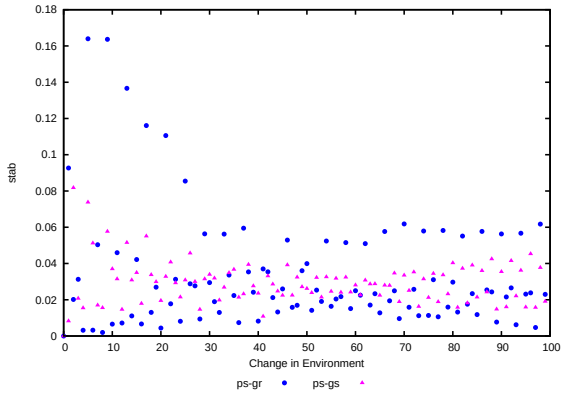
(a) *stab* values for $n_t = 10$ and $\tau_t = 10$



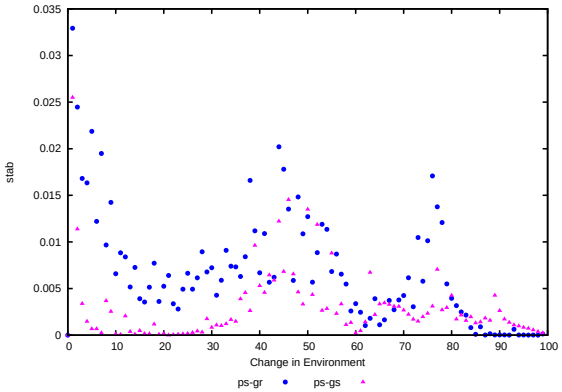
(b) *stab* values for $n_t = 10$ and $\tau_t = 25$



(c) *stab* values for $n_t = 10$ and $\tau_t = 50$

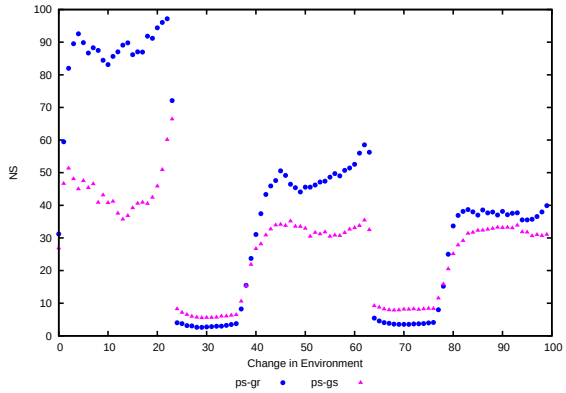


(d) *stab* values for $nfda2_t = 1$ and $\tau_t = 10$

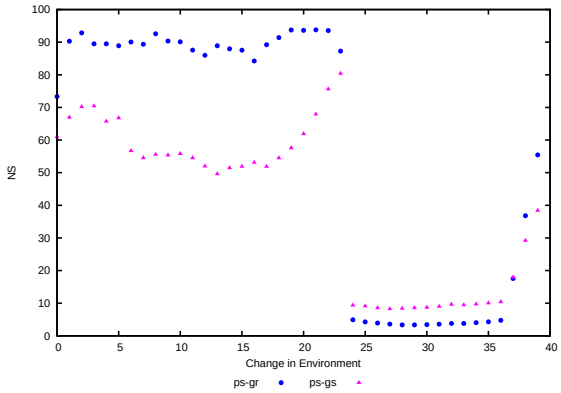


(e) *stab* values for $n_t = 20$ and $\tau_t = 10$

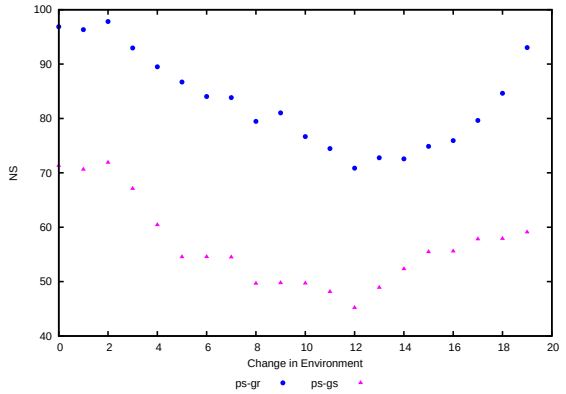
Figure 9.11: Average values of *stab* obtained by DVEPSO using either p_s-g_s or p_s-g_r solving FDA2



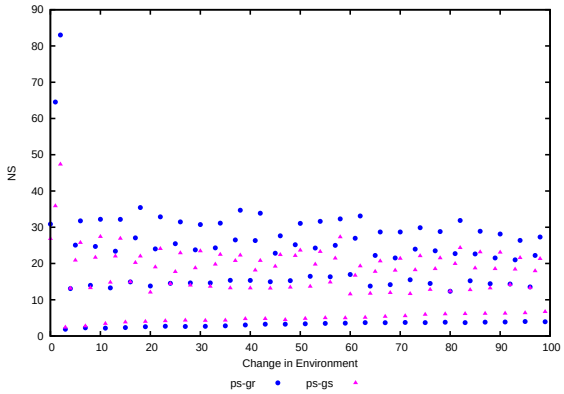
(a) NS values for $n_t = 10$ and $\tau_t = 10$



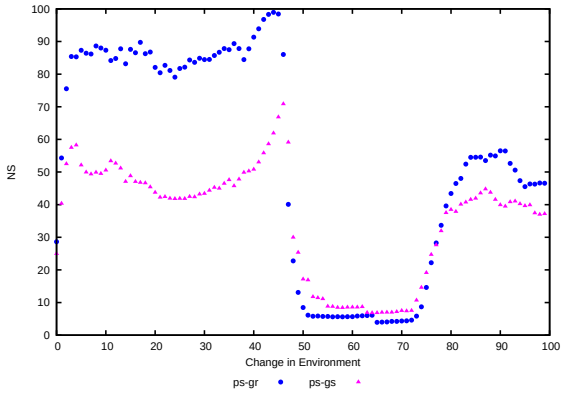
(b) NS values for $n_t = 10$ and $\tau_t = 25$



(c) NS values for $n_t = 10$ and $\tau_t = 50$



(d) NS values for $n_t = 1$ and $\tau_t = 10$



(e) NS values for $n_t = 20$ and $\tau_t = 10$

Figure 9.12: Average values of NS obtained by DVEPSO using either p_s-g_s or p_s-g_r solving FDA2

dMOP2 with a deceptive POF, dMOP2$_{dec}$. When solving dMOP2, all p_s combinations performed average, except p_s-g_s that performed the worst. Solving dMOP2$_{iso}$, p_s-g_r and p_s-g_n performed well, while the other two p_s combinations performed poorly. Once again, p_s-g_s obtained the worst performance when solving dMOP2$_{iso}$. However, for dMOP2$_{dec}$, all p_s combinations performed really well, obtaining the top four overall ranks. Tables 9.14 to 9.16 present the wins and losses for dMOP2, dMOP2$_{iso}$ and dMOP2$_{dec}$ respectively.

Table 9.14: Wins and Losses of dMOP2

n_t	τ_t	PM	Results	pbest-gbest combination															
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*acc*	Wins	7	1	1	1	4	0	4	3	4	3	3	3	3	3	4	3
10	10	*acc*	Losses	0	12	12	12	1	7	1	0	1	0	0	0	0	0	1	0
10	10	*acc*	Diff	7	-11	-11	-11	3	-7	3	3	3	3	3	3	3	3	3	3
10	10	*acc*	Rank	**1**	14	14	14	2	13	2	2	2	2	2	2	2	2	2	2
10	25	*acc*	Wins	2	3	2	0	1	1	0	0	1	6	3	0	2	1	1	6
10	25	*acc*	Losses	2	3	4	12	0	2	0	3	0	0	1	2	0	0	0	0
10	25	*acc*	Diff	0	0	-2	-12	1	-1	0	-3	1	6	2	-2	2	1	1	6
10	25	*acc*	Rank	9	9	13	16	5	12	9	15	5	**1**	3	13	3	5	5	**1**
10	50	*acc*	Wins	0	2	0	1	4	4	4	5	5	4	5	5	5	5	4	4
10	50	*acc*	Losses	13	12	14	12	0	6	0	0	0	0	0	0	0	0	0	0
10	50	*acc*	Diff	-13	-10	-14	-11	4	-2	4	5	5	4	5	5	5	5	4	4
10	50	*acc*	Rank	15	13	16	14	7	12	7	**1**	**1**	7	**1**	**1**	**1**	**1**	7	7
1	10	*acc*	Wins	5	5	5	4	0	2	3	0	3	0	1	2	0	0	1	3
1	10	*acc*	Losses	2	0	10	0	2	3	0	6	3	1	0	2	1	2	0	2
1	10	*acc*	Diff	3	5	-5	4	-2	-1	3	-6	0	-1	1	0	-1	-2	1	1
1	10	*acc*	Rank	3	**1**	15	2	13	10	3	16	8	10	5	8	10	13	5	5
20	10	*acc*	Wins	0	6	4	3	3	1	1	1	2	1	1	1	1	1	1	2
20	10	*acc*	Losses	15	4	1	0	1	0	1	2	1	1	0	0	1	0	1	1
20	10	*acc*	Diff	-15	2	3	3	2	1	0	-1	1	0	1	1	0	1	0	1
20	10	*acc*	Rank	16	3	**1**	**1**	3	5	11	15	5	11	5	5	11	5	11	5
all	all	*acc*	Wins	14	17	12	9	12	8	12	9	15	14	13	11	11	10	11	18
all	all	*acc*	Losses	32	31	41	36	4	18	2	11	5	2	1	4	2	2	2	3
all	all	*acc*	Diff	-18	-14	-29	-27	8	-10	10	-2	10	12	12	7	9	8	9	15
all	all	*acc*	Rank	14	13	16	15	8	12	4	11	4	2	2	10	6	8	6	**1**
10	10	*stab*	Wins	6	4	3	2	1	0	1	0	1	0	0	0	0	0	1	0
10	10	*stab*	Losses	0	1	1	1	1	5	3	0	2	0	1	0	0	0	3	1
10	10	*stab*	Diff	6	3	2	1	0	-5	-2	0	-1	0	-1	0	0	0	-2	-1
10	10	*stab*	Rank	**1**	2	3	4	5	16	14	5	11	5	11	5	5	5	14	11
10	25	*stab*	Wins	1	0	1	0	0	0	0	0	0	2	0	0	0	0	0	2
10	25	*stab*	Losses	0	0	0	4	0	0	0	2	0	0	0	0	0	0	0	0
10	25	*stab*	Diff	1	0	1	-4	0	0	0	-2	0	2	0	0	0	0	0	2
10	25	*stab*	Rank	3	5	3	16	5	5	5	15	5	**1**	5	5	5	5	5	**1**
10	50	*stab*	Wins	0	0	0	0	4	4	4	4	5	4	4	4	5	5	4	4
10	50	*stab*	Losses	12	12	12	12	0	3	0	0	0	0	0	0	0	0	0	0
10	50	*stab*	Diff	-12	-12	-12	-12	4	1	4	4	5	4	4	4	5	5	4	4
10	50	*stab*	Rank	13	13	13	13	4	12	4	4	**1**	4	4	4	**1**	**1**	4	4

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
1	10	*stab*	Wins	1	3	2	3	0	1	0	0	2	0	0	0	0	0	0	1
1	10	*stab*	Losses	0	0	3	0	0	0	0	5	1	0	0	0	0	3	0	1
1	10	*stab*	Diff	1	3	-1	3	0	1	0	-5	1	0	0	0	0	-3	0	0
1	10	*stab*	Rank	3	**1**	14	**1**	6	3	6	16	3	6	6	6	6	15	6	6
20	10	*stab*	Wins	3	1	4	11	3	0	0	0	0	0	0	0	0	0	0	0
20	10	*stab*	Losses	3	1	0	0	1	1	1	3	1	1	3	3	1	1	1	1
20	10	*stab*	Diff	0	0	4	11	2	-1	-1	-3	-1	-1	-3	-3	-1	-1	-1	-1
20	10	*stab*	Rank	4	4	2	**1**	3	6	6	14	6	6	14	14	6	6	6	6
all	all	*stab*	Wins	11	8	10	16	8	5	5	4	8	6	4	4	5	5	5	7
all	all	*stab*	Losses	15	14	16	17	2	9	4	10	4	1	4	3	1	4	4	3
all	all	*stab*	Diff	-4	-6	-6	-1	6	-4	1	-6	4	5	0	1	4	1	1	4
all	all	*stab*	Rank	12	14	14	11	**1**	12	6	14	3	2	10	6	3	6	6	3
10	25	*NS*	Wins	12	13	13	12	0	3	3	3	1	3	3	3	0	3	3	3
10	25	*NS*	Losses	2	0	0	0	13	4	4	4	13	4	4	4	14	4	4	4
10	25	*NS*	Diff	10	13	13	12	-13	-1	-1	-1	-12	-1	-1	-1	-14	-1	-1	-1
10	25	*NS*	Rank	4	**1**	**1**	3	15	5	5	5	14	5	5	5	16	5	5	5
10	50	*NS*	Wins	3	8	5	9	0	3	3	3	0	4	4	3	0	3	3	3
10	50	*NS*	Losses	5	0	0	0	13	0	0	3	13	0	0	2	13	2	2	1
10	50	*NS*	Diff	-2	8	5	9	-13	3	3	0	-13	4	4	1	-13	1	1	2
10	50	*NS*	Rank	13	2	3	**1**	14	6	6	12	14	4	4	9	14	9	9	8
1	10	*NS*	Wins	3	4	10	0	2	2	1	0	3	2	0	0	2	2	2	2
1	10	*NS*	Losses	9	8	1	0	1	1	2	3	0	1	3	3	1	1	1	0
1	10	*NS*	Diff	-6	-4	9	0	1	1	-1	-3	3	1	-3	-3	1	1	1	2
1	10	*NS*	Rank	16	15	**1**	10	4	4	11	12	2	4	12	12	4	4	4	3
all	all	*NS*	Wins	18	25	28	21	2	8	7	6	4	9	7	6	2	8	8	8
all	all	*NS*	Losses	16	8	1	0	27	5	6	10	26	5	7	9	28	7	7	5
all	all	*NS*	Diff	2	17	27	21	-25	3	1	-4	-22	4	0	-3	-26	1	1	3
all	all	*NS*	Rank	7	3	**1**	2	15	5	8	13	14	4	11	12	16	8	8	5
10	10	all	Wins	13	5	4	3	5	0	5	3	5	3	3	3	3	3	5	3
10	10	all	Losses	0	13	13	13	2	12	4	0	3	0	1	0	0	0	4	1
10	10	all	Diff	13	-8	-9	-10	3	-12	1	3	2	3	2	3	3	3	1	2
10	10	all	Rank	**1**	13	14	15	2	16	11	2	8	2	8	2	2	2	11	8
10	25	all	Wins	15	16	16	12	1	4	3	3	2	11	6	3	2	4	4	11
10	25	all	Losses	4	3	4	16	13	6	4	9	13	4	5	6	14	4	4	4
10	25	all	Diff	11	13	12	-4	-12	-2	-1	-6	-11	7	1	-3	-12	0	0	7
10	25	all	Rank	3	**1**	2	12	15	10	9	13	14	4	6	11	15	7	7	4
10	50	all	Wins	3	10	5	10	8	11	11	12	10	12	13	12	10	13	11	11
10	50	all	Losses	30	24	26	24	13	9	0	3	13	0	0	2	13	2	2	1
10	50	all	Diff	-27	-14	-21	-14	-5	2	11	9	-3	12	13	10	-3	11	9	10
10	50	all	Rank	16	13	15	13	12	9	3	7	10	2	**1**	5	10	3	7	5
1	10	all	Wins	9	12	17	7	2	5	4	0	8	2	1	2	2	2	3	6
1	10	all	Losses	11	8	14	0	3	4	2	14	4	2	3	5	2	6	1	3
1	10	all	Diff	-2	4	3	7	-1	1	2	-14	4	0	-2	-3	0	-4	2	3
1	10	all	Rank	12	2	4	**1**	11	8	6	16	2	9	12	14	9	15	6	4
20	10	all	Wins	3	7	8	14	6	1	1	1	2	1	1	1	1	1	1	2
20	10	all	Losses	18	5	1	0	2	1	2	5	2	2	3	3	2	1	2	2
20	10	all	Diff	-15	2	7	14	4	0	-1	-4	0	-1	-2	-2	-1	0	-1	0
20	10	all	Rank	16	4	2	**1**	3	5	9	15	5	9	13	13	9	5	9	5
all	all	all	Wins	43	50	50	46	22	21	24	19	27	29	24	21	18	23	24	33
all	all	all	Losses	63	53	58	53	33	32	12	31	35	8	12	16	31	13	13	11

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination															
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
all	all	all	Diff	-20	-3	-8	-7	-11	-11	12	-12	-8	21	12	5	-13	10	11	22
all	all	all	Rank	16	8	10	9	12	12	3	14	10	2	3	7	15	6	5	**1**

Table 9.15: Wins and Losses of dMOP2$_{iso}$

n_t	τ_t	PM	Results	pbest-gbest combination															
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*acc*	Wins	4	9	11	3	0	0	0	1	1	1	1	0	1	1	2	1
10	10	*acc*	Losses	11	1	1	0	3	3	2	1	2	2	1	3	2	2	2	0
10	10	*acc*	Diff	-7	8	10	3	-3	-3	-2	0	-1	-1	0	-3	-1	-1	0	1
10	10	*acc*	Rank	16	2	**1**	3	13	13	12	5	8	8	5	13	8	8	5	4
10	25	*acc*	Wins	0	0	3	1	3	3	2	3	3	3	9	3	4	3	2	5
10	25	*acc*	Losses	13	13	3	11	0	1	2	0	1	0	0	1	0	0	2	0
10	25	*acc*	Diff	-13	-13	0	-10	3	2	0	3	2	3	9	2	4	3	0	5
10	25	*acc*	Rank	15	15	11	14	4	8	11	4	8	4	**1**	8	3	4	11	2
10	50	*acc*	Wins	0	2	3	0	5	4	4	5	5	4	4	4	4	4	6	4
10	50	*acc*	Losses	14	13	12	14	0	0	0	0	0	0	0	0	4	0	0	1
10	50	*acc*	Diff	-14	-11	-9	-14	5	4	4	5	5	4	4	4	0	4	6	3
10	50	*acc*	Rank	15	14	13	15	2	5	5	2	2	5	5	5	12	5	**1**	11
1	10	*acc*	Wins	0	1	4	10	3	3	4	5	3	4	6	4	2	2	3	3
1	10	*acc*	Losses	15	14	11	5	1	1	0	0	1	0	0	0	2	3	1	3
1	10	*acc*	Diff	-15	-13	-7	5	2	2	4	5	2	4	6	4	0	-1	2	0
1	10	*acc*	Rank	16	15	14	2	7	7	4	2	7	4	**1**	4	11	13	7	11
20	10	*acc*	Wins	0	8	0	8	2	6	4	2	4	6	2	2	6	2	2	2
20	10	*acc*	Losses	14	4	14	4	2	0	0	2	2	0	2	2	0	4	2	4
20	10	*acc*	Diff	-14	4	-14	4	0	6	4	0	2	6	0	0	6	-2	0	-2
20	10	*acc*	Rank	15	4	15	4	8	**1**	4	8	7	**1**	8	8	**1**	13	8	13
all	all	*acc*	Wins	4	20	21	22	13	16	14	16	16	18	22	13	17	12	15	15
all	all	*acc*	Losses	67	45	41	34	6	5	4	3	6	2	3	6	8	9	7	8
all	all	*acc*	Diff	-63	-25	-20	-12	7	11	10	13	10	16	19	7	9	3	8	7
all	all	*acc*	Rank	16	15	14	13	9	4	5	3	5	2	**1**	9	7	12	8	9
10	10	*stab*	Wins	0	10	4	12	0	0	0	0	0	0	0	0	0	0	0	0
10	10	*stab*	Losses	2	0	0	0	2	0	3	3	2	2	3	2	1	2	1	3
10	10	*stab*	Diff	-2	10	4	12	-2	0	-3	-3	-2	-2	-3	-2	-1	-2	-1	-3
10	10	*stab*	Rank	7	2	3	**1**	7	4	13	13	7	7	13	7	5	7	5	13
10	25	*stab*	Wins	0	1	3	0	0	0	0	0	0	0	9	0	3	1	0	4
10	25	*stab*	Losses	4	5	0	4	0	1	1	1	1	1	0	1	0	0	2	0
10	25	*stab*	Diff	-4	-4	3	-4	0	-1	-1	-1	-1	-1	9	-1	3	1	-2	4
10	25	*stab*	Rank	14	14	3	14	6	7	7	7	7	7	**1**	7	3	5	13	2
10	50	*stab*	Wins	0	0	1	0	4	4	4	5	5	4	4	2	2	4	5	4
10	50	*stab*	Losses	12	10	10	13	0	0	0	0	0	0	0	0	3	0	0	0
10	50	*stab*	Diff	-12	-10	-9	-13	4	4	4	5	5	4	4	2	-1	4	5	4
10	50	*stab*	Rank	15	14	13	16	4	4	4	**1**	**1**	4	4	11	12	4	**1**	4
1	10	*stab*	Wins	6	8	3	3	0	0	0	3	0	0	3	0	0	0	0	0
1	10	*stab*	Losses	3	0	0	0	0	2	3	3	2	2	3	2	0	2	2	2
1	10	*stab*	Diff	3	8	3	3	0	-2	-3	0	-2	-2	0	-2	0	-2	-2	-2
1	10	*stab*	Rank	2	**1**	2	2	5	9	16	5	9	9	5	9	5	9	9	9
20	10	*stab*	Wins	0	4	0	5	0	2	0	0	1	1	0	0	2	0	0	0
20	10	*stab*	Losses	2	0	2	0	1	0	0	0	0	0	0	0	0	6	0	4
20	10	*stab*	Diff	-2	4	-2	5	-1	2	0	0	1	1	0	0	2	-6	0	-4

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
20	10	*stab*	Rank	13	2	13	**1**	12	3	7	7	5	5	7	7	3	16	7	15
all	all	*stab*	Wins	6	23	11	20	4	6	4	8	6	5	16	2	7	5	5	8
all	all	*stab*	Losses	23	15	12	17	3	3	7	7	5	5	6	5	4	10	5	9
all	all	*stab*	Diff	-17	8	-1	3	1	3	-3	1	1	0	10	-3	3	-5	0	-1
all	all	*stab*	Rank	16	2	11	3	6	3	13	6	6	9	**1**	13	3	15	9	11
10	25	*NS*	Wins	12	13	13	12	0	3	3	3	1	3	3	3	0	3	3	3
10	25	*NS*	Losses	2	0	0	0	13	4	4	4	13	4	4	4	14	4	4	4
10	25	*NS*	Diff	10	13	13	12	-13	-1	-1	-1	-12	-1	-1	-1	-14	-1	-1	-1
10	25	*NS*	Rank	4	**1**	**1**	3	15	5	5	5	14	5	5	5	16	5	5	5
10	50	*NS*	Wins	3	8	5	9	0	3	3	3	0	4	4	3	0	3	3	3
10	50	*NS*	Losses	5	0	0	0	13	0	0	3	13	0	0	2	13	2	2	1
10	50	*NS*	Diff	-2	8	5	9	-13	3	3	0	-13	4	4	1	-13	1	1	2
10	50	*NS*	Rank	13	2	3	**1**	14	6	6	12	14	4	4	9	14	9	9	8
all	all	*NS*	Wins	15	21	18	21	0	6	6	6	1	7	7	6	0	6	6	6
all	all	*NS*	Losses	7	0	0	0	26	4	4	7	26	4	4	6	27	6	6	5
all	all	*NS*	Diff	8	21	18	21	-26	2	2	-1	-25	3	3	0	-27	0	0	1
all	all	*NS*	Rank	4	**1**	3	**1**	15	7	7	13	14	5	5	10	16	10	10	9
10	10	all	Wins	4	19	15	15	0	0	0	1	1	1	1	0	1	1	2	1
10	10	all	Losses	13	1	1	0	5	3	5	4	4	4	4	5	3	4	3	3
10	10	all	Diff	-9	18	14	15	-5	-3	-5	-3	-3	-3	-3	-5	-2	-3	-1	-2
10	10	all	Rank	16	**1**	3	2	13	7	13	7	7	7	7	13	5	7	4	5
10	25	all	Wins	12	14	19	13	3	6	5	6	4	6	21	6	7	7	5	12
10	25	all	Losses	19	18	3	15	13	6	7	5	15	5	4	6	14	4	8	4
10	25	all	Diff	-7	-4	16	-2	-10	0	-2	1	-11	1	17	0	-7	3	-3	8
10	25	all	Rank	13	12	2	9	15	7	9	5	16	5	**1**	7	13	4	11	3
10	50	all	Wins	3	10	9	9	9	11	11	13	10	12	12	9	6	11	14	11
10	50	all	Losses	31	23	22	27	13	0	0	3	13	0	0	2	20	2	2	2
10	50	all	Diff	-28	-13	-13	-18	-4	11	11	10	-3	12	12	7	-14	9	12	9
10	50	all	Rank	16	12	12	15	11	4	4	6	10	**1**	**1**	9	14	7	**1**	7
1	10	all	Wins	6	9	7	13	3	3	4	8	3	4	9	4	2	2	3	3
1	10	all	Losses	18	14	11	5	1	3	3	3	3	2	3	2	2	5	3	5
1	10	all	Diff	-12	-5	-4	8	2	0	1	5	0	2	6	2	0	-3	0	-2
1	10	all	Rank	16	15	14	**1**	4	8	7	3	8	4	2	4	8	13	8	12
20	10	all	Wins	0	12	0	13	2	8	4	2	5	7	2	2	8	2	2	2
20	10	all	Losses	16	4	16	4	3	0	0	2	2	0	2	2	0	10	2	8
20	10	all	Diff	-16	8	-16	9	-1	8	4	0	3	7	0	0	8	-8	0	-6
20	10	all	Rank	15	2	15	**1**	12	2	6	8	7	5	8	8	2	14	8	13
all	all	all	Wins	25	64	50	63	17	28	24	30	23	30	45	21	24	23	26	29
all	all	all	Losses	97	60	53	51	35	12	15	17	37	11	13	17	39	25	18	22
all	all	all	Diff	-72	4	-3	12	-18	16	9	13	-14	19	32	4	-15	-2	8	7
all	all	all	Rank	16	9	12	5	15	3	6	4	13	2	**1**	9	14	11	7	8

Table 9.16: Wins and Losses of dMOP2$_{dec}$

n_t	τ_t	PM	Results	pbest-gbest combination															
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	10	*acc*	Wins	12	12	12	12	0	0	0	4	0	0	0	0	6	0	2	0
10	10	*acc*	Losses	0	0	0	0	7	6	4	4	4	7	5	6	4	4	4	5
10	10	*acc*	Diff	12	12	12	12	-7	-6	-4	0	-4	-7	-5	-6	2	-4	-2	-5
10	10	*acc*	Rank	**1**	**1**	**1**	**1**	15	13	8	6	8	15	11	13	5	8	7	11
10	25	*acc*	Wins	12	12	12	9	0	1	1	0	4	0	0	4	0	0	0	0
10	25	*acc*	Losses	0	0	0	0	4	4	3	4	3	6	8	3	4	6	4	6
10	25	*acc*	Diff	12	12	12	9	-4	-3	-2	-4	1	-6	-8	1	-4	-6	-4	-6
10	25	*acc*	Rank	**1**	**1**	**1**	4	9	8	7	9	5	13	16	5	9	13	9	13
10	50	*acc*	Wins	5	10	7	3	0	0	0	0	0	0	0	0	0	0	0	0
10	50	*acc*	Losses	0	0	0	0	4	1	3	2	0	0	2	4	3	4	1	1
10	50	*acc*	Diff	5	10	7	3	-4	-1	-3	-2	0	0	-2	-4	-3	-4	-1	-1
10	50	*acc*	Rank	3	**1**	2	4	14	7	12	10	5	5	10	14	12	14	7	7
1	10	*acc*	Wins	9	12	12	12	0	1	0	0	0	0	0	1	0	1	1	0
1	10	*acc*	Losses	0	0	0	0	4	3	4	8	4	4	4	4	4	3	3	4
1	10	*acc*	Diff	9	12	12	12	-4	-2	-4	-8	-4	-4	-4	-3	-4	-2	-2	-4
1	10	*acc*	Rank	4	**1**	**1**	**1**	9	5	9	16	9	9	9	8	9	5	5	9
20	10	*acc*	Wins	0	6	4	3	3	1	1	1	2	1	1	1	1	1	1	2
20	10	*acc*	Losses	15	4	1	0	1	0	1	2	1	1	0	0	1	0	1	1
20	10	*acc*	Diff	-15	2	3	3	2	1	0	-1	1	0	1	1	0	1	0	1
20	10	*acc*	Rank	16	3	**1**	**1**	3	5	11	15	5	11	5	5	11	5	11	5
all	all	*acc*	Wins	38	52	47	39	3	3	2	5	6	1	1	6	7	2	4	2
all	all	*acc*	Losses	15	4	1	0	20	14	15	20	12	18	19	17	16	17	13	17
all	all	*acc*	Diff	23	48	46	39	-17	-11	-13	-15	-6	-17	-18	-11	-9	-15	-9	-15
all	all	*acc*	Rank	4	**1**	2	3	14	8	10	11	5	14	16	8	6	11	6	11
10	10	*stab*	Wins	12	12	12	12	0	0	0	4	0	0	0	0	6	0	3	0
10	10	*stab*	Losses	0	0	0	0	7	6	4	4	4	7	5	7	4	4	4	5
10	10	*stab*	Diff	12	12	12	12	-7	-6	-4	0	-4	-7	-5	-7	2	-4	-1	-5
10	10	*stab*	Rank	**1**	**1**	**1**	**1**	14	13	8	6	8	14	11	14	5	8	7	11
10	25	*stab*	Wins	12	12	12	7	0	0	0	0	2	0	0	1	2	0	0	0
10	25	*stab*	Losses	0	0	0	0	4	3	3	4	3	4	6	3	3	4	4	7
10	25	*stab*	Diff	12	12	12	7	-4	-3	-3	-4	-1	-4	-6	-2	-1	-4	-4	-7
10	25	*stab*	Rank	**1**	**1**	**1**	4	10	8	8	10	5	10	15	7	5	10	10	16
10	50	*stab*	Wins	3	8	6	2	0	0	0	0	0	0	0	0	0	0	0	0
10	50	*stab*	Losses	0	0	0	0	4	0	3	1	0	0	2	4	2	2	0	1
10	50	*stab*	Diff	3	8	6	2	-4	0	-3	-1	0	0	-2	-4	-2	-2	0	-1
10	50	*stab*	Rank	3	**1**	2	4	15	5	14	9	5	5	11	15	11	11	5	9
1	10	*stab*	Wins	9	12	12	12	0	1	1	0	0	0	0	1	0	1	1	0
1	10	*stab*	Losses	0	0	0	0	4	3	4	9	4	4	4	4	4	3	3	4
1	10	*stab*	Diff	9	12	12	12	-4	-2	-3	-9	-4	-4	-4	-3	-4	-2	-2	-4
1	10	*stab*	Rank	4	**1**	**1**	**1**	10	5	8	16	10	10	10	8	10	5	5	10
20	10	*stab*	Wins	3	1	4	11	3	0	0	0	0	0	0	0	0	0	0	0
20	10	*stab*	Losses	3	1	0	0	1	1	1	3	1	1	3	3	1	1	1	1
20	10	*stab*	Diff	0	0	4	11	2	-1	-1	-3	-1	-1	-3	-3	-1	-1	-1	-1
20	10	*stab*	Rank	4	4	2	**1**	3	6	6	14	6	6	14	14	6	6	6	6
all	all	*stab*	Wins	39	45	46	44	3	1	1	4	2	0	0	2	8	1	4	0
all	all	*stab*	Losses	3	1	0	0	20	13	15	21	12	16	20	21	14	14	12	18
all	all	*stab*	Diff	36	44	46	44	-17	-12	-14	-17	-10	-16	-20	-19	-6	-13	-8	-18
all	all	*stab*	Rank	4	2	**1**	2	12	8	10	12	7	11	16	15	5	9	6	14

Continued on next page

n_t	τ_t	PM	Results	pbest-gbest combination														
				s-s	s-n	s-d	s-r	n-s	n-n	n-d	n-r	r-s	r-n	r-d	r-r	d-s	d-n	d-d	d-r
10	25	*NS*	Wins	12	13	13	12	0	3	3	3	1	3	3	3	0	3	3	3
10	25	*NS*	Losses	2	0	0	0	13	4	4	4	13	4	4	4	14	4	4	4
10	25	*NS*	Diff	10	13	13	12	-13	-1	-1	-1	-12	-1	-1	-1	-14	-1	-1	-1
10	25	*NS*	Rank	4	**1**	**1**	3	15	5	5	5	14	5	5	5	16	5	5	5
10	50	*NS*	Wins	3	8	5	9	0	3	3	3	0	4	4	3	0	3	3	3
10	50	*NS*	Losses	5	0	0	0	13	0	0	3	13	0	0	2	13	2	2	1
10	50	*NS*	Diff	-2	8	5	9	-13	3	3	0	-13	4	4	1	-13	1	1	2
10	50	*NS*	Rank	13	2	3	**1**	14	6	6	12	14	4	4	9	14	9	9	8
1	10	*NS*	Wins	2	12	0	2	2	0	2	2	1	1	2	2	2	2	0	2
1	10	*NS*	Losses	10	0	0	8	1	3	1	1	1	1	1	1	1	1	3	1
1	10	*NS*	Diff	-8	12	0	-6	1	-3	1	1	0	0	1	1	1	1	-3	1
1	10	*NS*	Rank	16	**1**	10	15	2	13	2	2	10	10	2	2	2	2	13	2
all	all	*NS*	Wins	17	33	18	23	2	6	8	8	2	8	9	8	2	8	6	8
all	all	*NS*	Losses	17	0	0	8	27	7	5	8	27	5	5	7	28	7	9	6
all	all	*NS*	Diff	0	33	18	15	-25	-1	3	0	-25	3	4	1	-26	1	-3	2
all	all	*NS*	Rank	10	**1**	2	3	14	12	5	10	14	5	4	8	16	8	13	7
10	10	all	Wins	24	24	24	24	0	0	0	8	0	0	0	0	12	0	5	0
10	10	all	Losses	0	0	0	0	14	12	8	8	8	14	10	13	8	8	8	10
10	10	all	Diff	24	24	24	24	-14	-12	-8	0	-8	-14	-10	-13	4	-8	-3	-10
10	10	all	Rank	**1**	**1**	**1**	**1**	15	13	8	6	8	15	11	14	5	8	7	11
10	25	all	Wins	36	37	37	28	0	4	4	3	7	3	3	8	2	3	3	3
10	25	all	Losses	2	0	0	0	21	11	10	12	19	14	18	10	21	14	12	17
10	25	all	Diff	34	37	37	28	-21	-7	-6	-9	-12	-11	-15	-2	-19	-11	-9	-14
10	25	all	Rank	3	**1**	**1**	4	16	7	6	8	12	10	14	5	15	10	8	13
10	50	all	Wins	11	26	18	14	0	3	3	3	0	4	4	3	0	3	3	3
10	50	all	Losses	5	0	0	0	21	1	6	6	13	0	4	10	18	8	3	3
10	50	all	Diff	6	26	18	14	-21	2	-3	-3	-13	4	0	-7	-18	-5	0	0
10	50	all	Rank	4	**1**	2	3	16	6	10	10	14	5	7	13	15	12	7	7
1	10	all	Wins	20	36	24	26	2	2	3	2	1	1	2	4	2	4	2	2
1	10	all	Losses	10	0	0	8	9	9	9	18	9	9	9	9	9	7	9	9
1	10	all	Diff	10	36	24	18	-7	-7	-6	-16	-8	-8	-7	-5	-7	-3	-7	-7
1	10	all	Rank	4	**1**	2	3	8	8	7	16	14	14	8	6	8	5	8	8
20	10	all	Wins	3	7	8	14	6	1	1	1	2	1	1	1	1	1	1	2
20	10	all	Losses	18	5	1	0	2	1	2	5	2	2	3	3	2	1	2	2
20	10	all	Diff	-15	2	7	14	4	0	-1	-4	0	-1	-2	-2	-1	0	-1	0
20	10	all	Rank	16	4	2	**1**	3	5	9	15	5	9	13	13	9	5	9	5
all	all	all	Wins	94	130	111	106	8	10	11	17	10	9	10	16	17	11	14	10
all	all	all	Losses	35	5	1	8	67	34	35	49	51	39	44	45	58	38	34	41
all	all	all	Diff	59	125	110	98	-59	-24	-24	-32	-41	-30	-34	-29	-41	-27	-20	-31
all	all	all	Rank	4	**1**	2	3	16	6	6	12	14	10	13	9	14	8	5	11

9.5 Summary

This chapter discussed the DVEPSO algorithm, which is an adaptation of SMOO VEPSO for DMOO. Similar to VEPSO, DVEPSO has two layers, namely a top layer that manages the top-level tasks and a lower layer that consists of the sub-swarms that handle the lower level tasks. On the lower level, the sub-swarms of DVEPSO checks the environment to determine whether a change has occurred, in addition to optimising the objectives that are performed by the sub-swarms of VEPSO. On the top level, in addition to the tasks of knowledge sharing and archive management performed by VEPSO, DVEPSO also responds to changes in the environment that were detected by the sub-swarms.

The optimisation process of DVEPSO is guided by local and global guides. Various ways of updating the local and global guides exist. This chapter investigated the influence of the various guide update approaches on the performance of DVEPSO. The results indicated that guide update approaches that incorporate Pareto-dominance knowledge outperformed the guide update approach of the original VEPSO algorithm that does not incorporate Pareto-dominance. The guide update approach that achieved the overall best performance was p_s-g_r. With this approach, the local guide is updated in such a way that the particle's fitness is measured with regards to only the objective function that the specific swarm optimises. Only if an improvement in the fitness of the current local guide can be obtained, the guide is updated, and no Pareto-dominance information is used. The global guide is updated if the new pbest is non-dominated with respect to the global guide, by randomly selecting either the pbest or the corresponding global guide.

The next chapter investigates the influence of other parameters on the performance of DVEPSO. These parameters include knowledge sharing swarm topologies, approaches to manage boundary constraint violations and approaches to respond to environment changes.

Chapter 12

Conclusions

"Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience." – Roger Bacon

This chapter summarises the research of this thesis. Section 12.1 summarises the findings and contributions of the thesis. In addition, possible future related research are proposed in Section12.2.

12.1 Summary of Conclusions

The main objective of this thesis was to develop and analyse a PSO-based MOA for DMOO. However, in order to determine whether the algorithm efficiently solves DMOOPs, benchmark functions are required that are representative of typical real-world problems. Furthermore, performance measures are required to quantify the performance of the algorithm.

Chapter 3 provided a comprehensive overview of benchmark functions used for DMOO. The overview highlighted three limitations of current DMOOPs, namely: all benchmark functions have a POS that is defined by linear functions and all decision variables have the same POS, none of the DMOOPs have an isolated POF, and none of the DMOOPs have a deceptive POF. In order to address these shortcomings of currently used DMOOPs, new DMOOPs were proposed that have POSs defined by non-linear functions and where each

decision variable has its own POS. In addition, approaches to adapt current DMOOPs' POF to become either deceptive or isolated were presented.

A comprehensive overview of the performance measures used to quantify the performance of DMOO algorithms was provided in Chapter 4. Furthermore, the following issues of currently used performance measures were discussed:

- The effect of outliers on the value of distance based performance measures;
- Misleading HV values when the shape of the POF changes from convex to non-convex over time, since a higher HV value is obtained by algorithms that lose track of the true POF than algorithms that are tracking the POF;
- The effect of boundary constraint violations on the HV values of an algorithm; and
- How the choice of performance measures may produce misleading results when comparing various DMOAs' performance.

The original VEPSO algorithm was introduced in Chapter 7. VEPSO is a multi-swarm PSO-based algorithm, where each sub-swarm solves only one objective function that was assigned to the specific sub-swarm. Therefore, VEPSO is easy to implement and can easily be extended to solve additional objective functions. Extensions proposed to the original VEPSO algorithm were also discussed in Chapter 7, namely storing the non-dominated solutions found so far by VEPSO in an archive, managing particles that move outside the bounds of the search space and various approaches to share knowledge between the various sub-swarms.

The PSO-based MOA proposed in this thesis, DVEPSO, was introduced in Chapter 9. The required adaptations to the original VEPSO algorithm for DMOO were highlighted. These changes include detecting whether a change occurred in the environment and responding to a change in an appropriate manner. The search process of DVEPSO is guided by local and global guides. Three new approaches to update local and global guides were proposed. Furthermore, an empirical study was conducted to determine whether the new approaches that use Pareto-dominance information outperforms the orginal VEPSO guide update approach that does not use Pareto-dominance information. The benchmark functions and performance measures used in the empirical study were selected according to the findings of Chapters 3 and 4. A new approach was introduced to analyse the obtained data, namely calculating the number of statistical significant wins

or losses for each function, each combination of frequency (τ_t) and severity (n_t) of change and each performance measure. These results were then analysed for each DMOO type, each n_t-τ_t combination and each performance measure. The results indicated that the approaches that use Pareto-dominance information outperformed the original VEPSO guide update approach.

Chapter 10 investigated the effect of various parameters on the performance of DVEPSO. These parameters included approaches to manage boundary constraint violations, approaches to share knowledge between the different sub-swarms and responses to a change in the environment applied to either the particles of the sub-swarms or the non-dominated solutions in the archive. The results of the empirical study indicated that:

- The best performing approach to manage boundary constraint violations was clamping, where any particle that violates a specific boundary of the search space is placed on or close to the violated boundary.
- The knowledge sharing approach that produced the best performance was using a random topology for the sub-swarms and selecting the global guide of another sub-swarm with tournament selection.
- The response applied to particles of the sub-swarms after a change in the environment occurred that performed the best, was re-initialising 30% of the particles of only the sub-swarm(s) whose objective function changed.
- The best performing response applied to the archive was to remove all solutions from the archive after a change in the environment occurred.

In order to determine whether DVEPSO solves DMOOPs efficiently, DVEPSO was compared against four other state-of-the-art DMOO algorithms. An empirical study was conducted and the results indicated that the PSO-based algorithms (DMOPSO and DVEPSO) completely outperformed the DMOEAs (DNSGA-II-A, DNSGA-II-B and dCOEA). The DNSGA-II algorithms obtained the best performance for DMOOPs with either a deceptive or isolated POF. Furthermore, DVEPSO was the only algorithm that efficiently solved DIMP2, a Type I DMOOP where each decision variable has its own rate of change. dCOEA found solutions, but did not converge towards POF of DIMP2. However, the other three DMOAs found on average only one solution or no solutions

at all. However, the results indicated that DVEPSO performed poorly when solving DMOOPs with a discontinuous POF. Taking all of the results into consideration, it can be concluded that PSO-based DMOAs efficiently solve DMOOPs of various types, and with various frequencies and severities of change.

12.2 Future Research

The following related research is suggested for future investigation:

- DVEPSO is easy to implement and to extend for additional objectives. However, a scalability study is required to determine to which extent DVEPSO is scalable with regards to both the number of decision variables and the number of objectives. However, when increasing the number of objectives, it is important to note that the usage of Pareto-dominance to evaluate the quality of solutions is ineffective, since many solutions will be non-dominanted with regards to the other found solutions. Therefore, research in many-objective optimisation is emerging [130].
- The empirical studies discussed in this thesis investigated the influence of various parameters on the performance of DVEPSO. The knowledge gained from these studies should be used to develop a self-adapting DVEPSO algorithm, eliminating the need to optimise parameters to solve new DMOOPs.
- The results of the empirical studies indicated that DVEPSO struggles to solve DMOOPs with a discontinuous POF, i.e. where the POF consists of various separated continous parts. Various local seach approaches should be incorporated into DVEPSO and investigated to determine whether the local search algorithm improves DVEPSO's performance when solving DMOOPs with discontinuous POFs.
- The empirical study of Chapter 9 investigated various approaches to update local and global guides. In addition, various guide selection approaches should be investigated to determine whether certain guide selection approaches lead to an improved diversity or spread of the found non-dominated solutions.
- Incorporating dynamic PSOs, for example the quantum PSO or charged PSO, into DVEPSO when solving DMOOPs where the POS changes over time.
- Evaluating the performance of DVEPSO in noisy environments.

- Developing performance measures for DMOO that are not vulnerable to the issues discussed in Chapter 4 and that do not require prior knowledge about the true POF.

Bibliography

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

[2] Z. Avdagić, S. Konjicija, and S. Omanović. Evolutionary approach to solving non-stationary dynamic multi-objective problems. In A. Abraham, A-E. Hassanien, P. Siarry, and A. Engelbrecht, editors, *Foundations of Computational Intelligence Volume 3*, volume 203 of *Studies in Computational Intelligence*, pages 267–289. Springer Berlin/Heidelberg, 2009.

[3] C.R.B. Azevedo and A.F.R. Araujo. Generalized immigration schemes for dynamic evolutionary multiobjective optimization. In *Proceedings of Congress on Evolutionary Computation*, pages 2033–2040, june 2011.

[4] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. part i: background and development. *Natural Computing*, 6(4):467–484, 2007.

[5] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1):109–124, 2008.

[6] T. Bartz-Beielstein, P. Limbourg, J. Mehnen, K. Schmitt, K.E. Parsopoulos, and M.N. Vrahatis. Particle swarm optimizers for pareto optimization with enhanced archiving techniques. In *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1780–1787, dec. 2003.

[7] N. Beume, C.M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, oct. 2009.

[8] N. Beume and G. Rudolph. Faster s-metric calculation by considering dominated hypervolume as klee's measure problem. In *Proceedings of Computational Intelligence*, pages 233–238. IASTED/ACTA Press, 2007.

[9] H-G. Beyer and H-P. Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[10] Z. Bingul. Adaptive genetic algorithms applied to dynamic multiobjective problems. *Applied Soft Computing*, 7:791–799, 2007.

[11] J. Branke. Gecco tutorial: Optimization in dynamic environments. Available online at: `http://www.cs.york.ac.uk/rts/docs/GECCO_2004/Workshop%20proceedings/gecco04/TUT026.pdf`, 2004. Last accessed on: 12 May 2011.

[12] J. Branke, E. Salihoğlu, and Ş. Uyar. Towards an analysis of dynamic environments. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 1433–1440, New York, NY, USA, 2005. ACM.

[13] J. Branke and H. Schmeck. *Advances in evolutionary computing: theory and applications*, chapter Designing evolutionary algorithms for dynamic optimization problems, pages 239–262. Natural Computing Series. Springer, 2003.

[14] F.M. Burnet. *The clonal selection theory of acquired immunity*. Cambridge University Press, Cambridge, U.K., 1959.

[15] M. Cámara, J. Ortega, and F. de Toro. The parallel single front genetic algorithm (psfga) in dynamic multi-objective optimization. In F. Sandoval, A. Prieto, J. Cabestany, and M. Gra na, editors, *Computational and Ambient Intelligence*, volume 4507 of *Lecture Notes in Computer Science*, pages 300–307. Springer Berlin/Heidelberg, 2007.

[16] M. Cámara, J. Ortega, and F. de Toro. A single front genetic algorithm for parallel multi-objective optimization in dynamic environments. *Neurocomputing*, 72(16-18):3570–3579, 2009. Financial Engineering; Computational and Ambient Intelligence (IWANN 2007).

[17] M. Cámara, J. Ortega, and F. de Toro. Approaching dynamic multi-objective optimization problems by using parallel evolutionary algorithms. In C. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 63–86. Springer Berlin/Heidelberg, 2010.

[18] M. Cámara, J. Ortega, and F.J. de Toro. Parallel processing for multi-objective optimization in dynamic environments. *International Parallel and Distributed Processing Symposium*, 0:243–250, 2007.

[19] M. Cámara, J. Ortega, and F.J. de Toro. A single front genetic algorithm for parallel multi-objective optimization in dynamic environments. *Neurocomputing*, 72(16–18):3570–3579, 2007.

[20] M. Cámara, J. Ortega, and F.J. de Toro. A diversity enhanced signle front multi-objective algorithm for dynamic optimization problems. In *Proceedings of the International Conference on Metaheuristics and Nature Inspired Computing*, 2008.

[21] A. Carlisle and G. Dozier. An off-the-shelf pso. In *Proceedings of Particle Swarm Optimization Workshop*, Indianapolis, Indiana, U.S.A., 2001.

[22] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of World Automation Congress*, volume 13, pages 265–270, Orlando, Florida, U.S.A., June 2002.

[23] H. Chen, M. Li, and X. Chen. Using diversity as an additional-objective in dynamic multi-objective optimization algorithms. *Electronic Commerce and Security, International Symposium*, 1:484–487, 2009.

[24] CHPC. Sun hybrid system. http://www.chpc.ac.za/sun. Last accessed online on: 20 December 2011.

[25] W. Chu, X. Gao, and S. Sorooshian. Handling boundary constraints for particle swarm optimization in high-dimensional search space. *Information Sciences*, 2010. In Press.

[26] M. Clerc. *Particle swarm optimization*. ISTE Ltd, 2006.

[27] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, feb 2002.

[28] H.G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, Naval Research Laboratory: Navy Center for Applied Research in Articial Intelligence, Washington, DC, U.S.A., 11 December 1990.

[29] C. Coello Coello and G.T. Pulido. A micro-genetic algorithm for multiobjective optimization. In E. Zitzler, L. Thiele, K. Deb, C. Coello Coello, and D. Corne, editors, *Evolutionary multi-criterion optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 126–140. Springer Berlin/Heidelberg, 2001.

[30] C.A. Coello Coello. Evolutionary multiobjective optimization. Available online at: `http://www.lania.mx/~ccoello/`. Last accessed on: 23 May 2011.

[31] C.A. Coello Coello. *Computational Intelligence: Principles and Practice*, chapter 20 years of evolutionary multi-objective optimization: what has been done and what remains to be done, pages 73–88. IEEE Computational Intelligence Society, 2006.

[32] C.A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, feb 2006.

[33] C.A. Coello Coello and M.S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Proceedings of Congress on Evolutionary Computation*, volume 2, pages 1051–1056, 2002.

[34] C.A. Coello Coello, G.T. Pulido, and M.S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, june 2004.

[35] C.A. Coello Coello, D.A. van Veldhuizen, and G.B. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic Publishers/Plenum Publishers, 2002.

[36] Y. Collette and P. Siarry. *Multiobjective optimization*. Decision Engineering. Springer-Verlag, 2003.

[37] C. Cruz, J. González, and D. Pelta. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pages 1–22, 2010.

[38] K. Deb. Multi-objective genetic algorithms: problem difficulties and construction of test problems. *Evolutionary Computation*, (3):205–230, 1999.

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

[40] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, Ltd, 2004.

[41] K. Deb. Single and multi-objective dynamic optimization: two tales from an evolutionary perspective. Technical Report 2011004, Kalyanmoy Deb Kanpur Genetic Algorithms Laboratory (KanGAL), February 2011.

[42] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Technical Report 200001, Kanpur, India, 2000.

[43] K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[44] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395, 2002.

[45] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers, june.

[46] K. Deb, U.B. Rao N., and S. Karthik. Dynamic multi-objective optimization and decision-making using modied nsga-ii: a case study on hydro-thermal power scheduling. In *Proceedings of International Conference on Evolutionary Multi-criterion optimization*, pages 803–817, Matsushima, Japan, 2007.

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

[48] K. Deb, A. Sinha, and S. Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 1141–1148, New York, NY, USA, 2006. ACM.

[49] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of Congress on Evolutionary Computation*, pages 825–830, Honolulu, 12-17 May 2002.

[50] J. Duhain. Empirical analysis of particle swarm optimization on dynamic environments. Master's thesis, University of Pretoria. To be published.

[51] R. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, first edition, 1996. page 217.

[52] R.C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings of Congress on Evolutionary Computation*, volume 1, pages 81–86, 2001.

[53] R.C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of Congress on Evolutionary Computation*, pages 94–100, May 2001.

[54] T. Elsayed. Genetic algorithms tutorial. Last access online: 11 April 2011.

[55] A.P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley and Sons, Ltd, 2005.

[56] A.P. Engelbrecht. Fruitless wandering: Where are my particles?, 2012. Proceedings of World Congress on Computational Intelligence: Congress on Evolutionary Compuation. To be published.

[57] M. Farina. A minimal cost hybrid strategy for pareto optimal front approximation. *Evolutionary Optimization*, 3(1):41–52, 2001.

[58] M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8(5):425–442, October 2004.

[59] L.J. Fogel. *On the organization of intellect*. PhD thesis, University of California, 1964.

[60] C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the International Conference in Genetic Algorithms*, pages 416–423, San Franciso, U.S.A., 1993.

[61] C.M. Fonseca, M. López-Ibá nez, L. Paquete, and A.P. Guerreiro. Computation of the hypervolume indicator. Version 1.3. Last accessed on 4 February 2012.

[62] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *Proceedings of Congress on Evolutionary Computation*, pages 1157–1163, july 2006.

[63] F.v.d.Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science University of Pretoria, 2002.

[64] Gheorghe and Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[65] C-K. Goh, Y.S. Ong, K.C. Tan, and E.J. Teoh. An investigation on evolutionary gradient search for multi-objective optimization. In *Proceedings of World*

Congress on Computational Intelligence: Congress on Evolutionary Computation, pages 3741–3746, Hong Kong, june 2008.

[66] C-K. Goh and K. Tan. A coevolutionary paradigm for dynamic multi-objective optimization. In *Evolutionary Multi-objective Optimization in Uncertain Environments*, volume 186 of *Studies in Computational Intelligence*, pages 153–185. Springer Berlin/Heidelberg, 2009.

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

[68] D.E. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesly, 1989.

[69] D.E. Goldberg and K. Deb. *Foundations of genetic algorithms*, chapter A comparison of selection schemes used in genetic algorithms, pages 69–93. Morgan Kaufmann, 1991.

[70] D.E. Goldberg and J. Richardson. Genetic algorithm with sharing for multimodal function optimization. In *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum.

[71] M. Greeff and A. Engelbrecht. Dynamic multi-objective optimisation using pso. In Nadia Nedjah, Leandro dos Santos Coelho, and Luiza de Macedo Mourelle, editors, *Multi-Objective Swarm Intelligent Systems*, volume 261 of *Studies in Computational Intelligence*, pages 105–123. Springer Berlin/Heidelberg, 2010.

[72] M. Greeff and A.P. Engelbrecht. Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation. In *Proceedings of World Congress on Computational Intelligence (WCCI): Congress on Evoluationary Computation*, pages 2917–2924, Hong Kong, june 2008.

[73] J. Grobler and A.P. Engelbrecht. Hybridizing pso and de for improved vector evaluated multi-objective optimization. In *Proceedings of Congress on Evolutionary Computation*, pages 1255–1262, may 2009.

[74] S-U. Guan, Q. Chen, and W. Mo. Evolving dynamic multi-objective optimization problems with objective replacement. *Artificial Intelligence Review*, 23:267–293, 2005.

[75] M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 22 March 1998.

[76] I. Hatzakis and D. Wallace. Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 1201–1208, New York, NY, USA, 2006. ACM.

[77] M. Helbig and A.P. Engelbrecht. *Metaheuristics for dynamic optimization*, chapter Dynamic multi-objective optimisation using PSO. Springer Verlag. To be published.

[78] M. Helbig and A.P. Engelbrecht. Archive management for dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. In *Proceedings of Congress on Evolutionary Computation*, pages 2047–2054, New Orleans, U.S.A., june 2011.

[79] S. Helwig and R. Wanka. Particle swarm optimization in high-dimensional bounded search spaces. In *Proceedings of Swarm Intelligence Symposium*, pages 198–205, apr 2007.

[80] A.G. Hernández-Díaz, L.V. Santana-Quintero, C.A. Coello Coello, and J. Molina. Pareto-adaptive ϵ-dominance. *Evolutionary Computation*, 15(4):493–517, 2007.

[81] F. Herrera, M. Lozano, and J.L. Verdegay. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[82] J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, Oxford, UK, 1975.

[83] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of World Congress on Computational Intelligence: Congress on Evolutionary Computation*, volume 1, pages 82–87, jun 1994.

[84] L. Huang, I.H. Suh, and A. Abraham. Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants. *Information Sciences*, 181(11):2370– 2391, 2011.

[85] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, oct. 2006.

[86] E.J. Hughes. Multiple single objective pareto sampling. In *Proceedings of the Congress on Evolutionary Computation*, volume 4, pages 2678–2684, dec. 2003.

[87] A. Isaacs, V. Puttige, T. Ray, W. Smith, and S. Anavatti. Development of a memetic algorithm for dynamic multi-objective optimization and its applications for online neural network modeling of uavs. In *Proceedings of World Congress on Computational Intelligence: International Joint Conference on Neural Networks*, pages 548 –554, june 2008.

[88] A. Isaacs, T. Ray, and W. Smith. Memetic algorithm for dynamic bi-objective optimization problems. In *Proceedings of Congress on Evolutionary Computation*, pages 1707–1713, Trondheim, Norway, may 2009.

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

[90] Y. Jin and B. Sendhoff. Constructing dynamic optimization test problems using the multi-objective optimization concept. In G. Raidl et al., editors, *Applications*

of Evolutionary Computing, volume 3005 of *Lecture Notes in Computer Science*, pages 525–536. Springer Berlin/Heidelberg, 2004.

[91] K. De Jong. Evolving in a changing world. In *Proceedings of International Symposium on Foundations of Intelligent Systems*, Lecture Notes in Computer Science, pages 512–519. Springer Berlin/Heidelberg, 1999.

[92] K.A. De Jong. *Analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Computer and Communication Sciences Department, University of Michigan, aug 1975.

[93] J. Kennedy. The particle swarm: social adaptation of knowledge. In *Proceedings of the International Conference on Evolutionary Computation*, pages 303–308, apr 1997.

[94] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of International Conference on Neural Networks*, volume IV, pages 1942–1948, 1995.

[95] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of World Congress on Computational Intelligence: Congress on Evolutionary Computation*, volume 2, pages 1671–1676, Honolulu, Hawaii, May 2002.

[96] A.K.M. Khaled, A. Talukder, and M. Kirley. A pareto following variation operator for fast-converging multiobjective evolutionary algorithms. In *Proceedings of World Congress on Computational Intelligence: Congress on Evolutionary Computation*, pages 2270–2277, june 2008.

[97] K. Kim, R.I. McKay, and B-R. Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the Conference on Genetic and Evolutionary Computation*.

[98] J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105, 1999.

[99] J.D. Knowles. *Local-search and hybrid evolutionary algorithms for Pareto optimisation*. PhD thesis, Department of Computer Science The University of Reading, 2002.

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

[101] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evoluationary multiobjective optimization. *Evoluationary Computation*, 10(3):263–282, 2002.

[102] M.S. Lechuga. *Multi-objective optimisation using sharing in swarm optimisation algorithms*. PhD thesis, University of Birmingham, July 2009.

[103] Y-W. Leung and Y. Wang. U-measure: A quality measure for multiobjective programming. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(3):337–343, 2003.

[104] G. Leyland. *Multi-objective optimization applied to industrial energy problems*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2002.

[105] H. Li and Q. Zhang. A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages. In T. Runarsson, H-G. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley, and X. Yao, editors, *Parallel problem solving from nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 583–592. Springer Berlin/Heidelberg, 2006.

[106] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evoluationary Computation*, 13(2):284–302, 2009.

[107] X. Li. Better spread and convergence: particle swarm multiobjective optimization using the maximin fitness function. In K. Deb, editor, *Proceedings of Conference on*

Genetic and Evolutionary Computation, volume 3102 of *Lecture Notes in Computer Science*, pages 117–128. Springer Berlin/Heidelberg, 2004.

[108] X. Li, J. Branke, and M. Kirley. On performance metrics and particle swarm methods for dynamic multiobjective optimization problems. In *Proceedings of Congress on Evolutionary Computation*, pages 576–583, sept. 2007.

[109] Kampur Genetic Algorithms Library. Software developed at kangal. Available online at: `http://www.iitk.ac.in/kangal/codes.shtml`. Last accessed on: 12 May 2011.

[110] C-A. Liu. New dynamic multiobjective evolutionary algorithm with core estimation of distribution. *Electrical and Control Engineering, International Conference on*, 0:1345–1348, 2010.

[111] C-A. Liu and Y. Wang. New evolutionary algorithm for dynamic multiobjective optimization problems. In L. Jiao, L. Wang, X-B. Gao, J. Liu, and F. Wu, editors, *Advances in Natural Computation*, volume 4221 of *Lecture Notes in Computer Science*, pages 889–892. Springer Berlin/Heidelberg, 2006.

[112] C-A. Liu and Y. Wang. Dynamic multi-objective optimization evolutionary algorithm. *International Conference on Natural Computation*, 4:456–459, 2007.

[113] R. Liu, W. Zhang, L. Jiao, F. Liu, and J. Ma. A sphere-dominance based preference immune-inspired algorithm for dynamic multi-objective optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 423–430, New York, NY, USA, 2010. ACM.

[114] G. Lizárraga Lizárraga. *On the evaluation of the quality of non-dominated sets*. PhD thesis, Centro de Investigación en Matemáticas, A.C. (CIMAT), April 2009.

[115] K.M. Malan and A.P. Engelbrecht. Algorithm comparisons and the significance of population size. In *Proceedings of the World Congress on Computational Intelligence: Congress on Evolutionary Computation*, pages 914–920, june 2008.

[116] W. Mathyssen and A.P. Engelbrecht.

[117] J. Mehnen, G. Rudolph, and T. Wagner. Evolutionary optimization of dynamic multiobjective functions. Technical Report CI-204/06, Universität Dortmund, Universität Dortmund, Fachbereich Informatik/XI, 44221, Dortmund, Germany, May 2006.

[118] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer Berlin/Heidelberg, 1996.

[119] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical report, California Institute of Technology, Pasadena, California, U.S.A., 1999.

[120] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[121] M. Obitko. Introduction to genetic algorithms: biological background. Last accessed online: 11 April 2011.

[122] G. Pampará, A.P. Engelbrecht, and T. Cloete. Cilib: A collaborative framework for computational intelligence algorithms - part i. In *Proceedings of World Congress on Computational Intelligence*, pages 1750–1757, Hong Kong, 1-8 June 2008. Source code available at: http://www.cilib.net. Last accessed on: 6 March 2011.

[123] K.E. Parsopoulos, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis. Vector evaluated differential evolution for multiobjective optimization. In *Proceedings of Congress on Evolutionary Computation*, volume 1, pages 204–211, june 2004.

[124] K.E. Parsopoulos, D.K. Tasoulis, and M.N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *Proceedings of IASTED International Conference on Artificial Intelligence and Applications*, Innsbruck Austria, 2004.

[125] K.E. Parsopoulos and M.N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[126] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence*, 1(1):33–57, 2007.

[127] T. Ray, A. Isaacs, and W. Smith. A memetic algorithm for dynamic multiobjective optimization. In C-K. Goh, Y-S. Ong, and K. Tan, editors, *Multi-Objective memetic algorithms*, volume 171 of *Studies in Computational Intelligence*, pages 353–367. Springer Berlin/Heidelberg, 2009.

[128] R.E. Rosenthal. Principles of multiobjective optimization. *Decision Sciences*, 16:132–152, 1985.

[129] R. Roy and J. Mehnen. Dynamic multi-objective optimisation for machining gradient materials. *CIRP Annals - Manufacturing Technology*, (57):429–432, 2008.

[130] D.K. Saxena, T. Ray, K. Deb, and A.Tiwari. Constrained many-objective optimization: a way forward. In *Proceedings of Congress on Evolutionary Computation*, pages 545–552, may 2009.

[131] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[132] J.R. Schott. Fault tolerance design using single and multi-criteria genetic algorithms. Master's thesis, Department of Aeronautics and Astronautics Massachusetts Institute of Technology, 1995.

[133] D. Schuhmacher, B-T. Vo, and B-N. Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Transactions on Signal Processing*, 56(8):3447–3457, aug 2008.

[134] H-P. Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisciplinary systems research*, 26, 1977. Birkhäuser, Basel.

[135] R. Shang, L. Jiao, M. Gong, and B. Lu. Clonal selection algorithm for dynamic multiobjective optimization. In Y. Hao, J. Liu, Y. Wang, Y-M. Cheung, H. Yin,

L. Jiao, J. Ma, and Y-C. Jiao, editors, *Computational Intelligence and Security*, volume 3801 of *Lecture Notes in Computer Science*, pages 846–851. Springer Berlin/Heidelberg, 2005.

[136] Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of Congress on Evolutionary Computation*, pages 69–73, May 1998.

[137] M. Sierra and C. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. In Carlos Coello Coello, Arturo Hernndez Aguirre, and Eckart Zitzler, editors, *Evolutionary multi-criterion optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 505–519. Springer Berlin Heidelberg, 2005.

[138] M. Cámara Sola. *Parallel processing for dynamic multi-objective optimization*. PhD thesis, Universidad de Granada, Dept. of Computer Architecture and Computer Technology, Universidad de Granada, Spain, April 2010.

[139] W.M. Spears. *The role of mutation and recombination in evolutionary algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 1998.

[140] M. Srinivas and L.M. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17–26, jun 1994.

[141] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[142] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[143] G. Sywerda. Uniform crossover in genetic algorithms. In *Proceedings of International Conference on Genetic algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[144] A.K.M. Talukder and A. Khaled. Towards high speed multiobjective evolutionary optimizers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1791–1794, New York, NY, USA, 2008. ACM.

[145] A.K.M. Talukder, A. Khaled, M. Kirley, and R. Buyya. A pareto following variation operator for fast-converging multiobjective evolutionary algorithms. In *Proceedings of the annual conference on Genetic and evolutionary computation*, pages 721–728, New York, NY, USA, 2008. ACM.

[146] K. Tan and C. Goh. Handling uncertainties in evolutionary multi-objective optimization. In Jacek Zurada, Gary Yen, and Jun Wang, editors, *Computational Intelligence: Research Frontiers*, volume 5050 of *Lecture Notes in Computer Science*, pages 262–292. Springer Berlin/Heidelberg, 2008. 10.1007/978-3-540-68860-0_13.

[147] K.C. Tan, Y.H. Chew, T.H. Lee, and Y.J. Yang. A cooperative coevolutionary algorithm for multiobjective optimization. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, volume 1, pages 390–395, oct 2003.

[148] K.C. Tan, Y.J. Yang, and C.K. Goh. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):527–549, oct 2006.

[149] M. Tang, Z. Huang, and G. Chen. The construction of dynamic multi-objective optimization test functions. In *Proceedings of International Conference on Advances in Computation and Intelligence*, pages 72–79, Berlin, Heidelberg, 2007. Springer-Verlag.

[150] E. Tantar, A-A. Tantar, and P. Bouvry. On dynamic multi-objective optimization, classification and performance measures. In *Proceedings of Congress on Evolutionary Computation*, pages 2759–2766, june 2011.

[151] D.A. van Veldhuizen. *Multiobjective evolutionary algorithms: classification, analyses, and new innovations*. PhD thesis, Graduate School of Engineering Air University, 1999.

[152] F. v.d.Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.

[153] Y. Wang and C. Dang. An evolutionary algorithm for dynamic multi-objective optimization. *Applied Mathematics and Computation*, 25:6–18, 2008.

[154] Y. Wang and B. Li. Fh-moea: multi-objective evolutionary algorithm based-on fast hyper-volume contribution approach. *Journal of University of Science and Technology*, 38:802–809, 2008.

[155] Y. Wang and B. Li. Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment. In *Proceedings of Congress on Evolutionary Computation*, pages 630–637, Trondheim, Norway, may 2009.

[156] Y. Wang and B. Li. Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. *Memetic Computing*, 2(1):3–24, 2010.

[157] K. Weicker. Performance measures for dynamic environments. In J. Guervós, P. Adamidis, H-G. Beyer, H-P. Schwefel, and J-L. Fernández-Villaca nas, editors, *Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*, pages 64–73. Springer Berlin/Heidelberg, 2002.

[158] A.H. Wright. *Foundations of genetic algorithms*, volume 1, chapter Genetic algorithms for real parameter optimization, pages 205–218. Elsevier Science & Technology Books, june 1991.

[159] X. Yao. An overview of evolutionary computation, 1995. Invited tutorial at International Conference for Young Computer Scientists (ICYCS), Beijing, China. Available online at: `http://0-citeseerx.ist.psu.edu.innopac.up.ac.za/viewdoc/download?doi=10.1.1.56.9127&rep=rep1&type=pdf`. Last accessed on: 30 September 2011.

[160] S-Y. Zeng, G. Chen, L. Zheng, H. Shi, H. de Garis, L. Ding, and L. Kang. A dynamic multi-objective evolutionary algorithm based on an orthogonal design. In *Proceedings of Congress on Evolutionary Computation*, pages 573–580, Vancouver, Canada, 16-21 July 2006.

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

[162] W.-J. Zhang, X.-F. Xie, and D.-C. Bi. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In *Congress on Evolutionary Computation*, volume 2, pages 2307–2311, June 2004.

[163] Z. Zhang. Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control. *Applied Soft Computing*, 8:959–971, 2007.

[164] Z. Zhang and S. Qian. Artificial immune system in dynamic environments solving time-varying non-linear constrained multi-objective problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15:1333–1349, 2011.

[165] B. Zheng. A new dynamic multi-objective optimization evolutionary algorithm. In *Proceedings of Third International Conference on Natural Computation*, volume 5, pages 565–570, aug. 2007.

[166] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. 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 832–846. Springer Berlin/Heidelberg, 2007.

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

[168] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: on the design of pareto-compliant indicators via weighted integration. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer Berlin/Heidelberg, 2007.

[169] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[170] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 2001.

[171] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms a comparative case study. 1498:292–301, 1998.

[172] 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.

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

Appendix A

List of Symbols

Δ	spacing metric of Deb
d_i	Euclidean distance in the objective space between solution i of approximated POF and the nearest member of the sampled points of the true POF
d_k^e	distance between the extreme solutions of the approximated POF and the sampled solutions of the true POF
$\prec$	domination relational operator
$\preceq$	weak domination relational operator
$\prec_\epsilon$	ϵ-domination relational operator
F	feasible space, $F \subseteq S$
f_k	objective function
$\mathbf{f}(\mathbf{x}, \mathbf{w}(t))$	dynamic objective function vector
$f(\mathbf{x}, \mathbf{w}(t))$	dynamic objective function
$\mathbf{f}_{ref}$	reference vector for hypervolume calculation
$\mathbf{f}$	objective function vector

g	inequality constraint
γ	path between two solutions in objective space
$\mathbf{g}$	inequality constraint vector
h	equality constraint
$\mathbf{h}$	equality constraint vector
F	neighbourhood of points, $N \subseteq F$
ND	number of non-dominated solutions
n_g	number of inequality constraints
n_h	number of equality constraints
n_k	number of objective functions
n_t	severity of change
n_x	number of decision variables
O_{space}	objective space
O_C	complete outperformance
O_S	strong outperformance
O_W	weak outperformance
O	outperforms
$PF^*(t)$	Pareto-optimal front at time t
PF^*	Pareto-optimal front
PF^*_ϵ	ϵ-approximate Pareto-optimal front
P^*	Pareto-optimal set

PF_g^*	global POF
PF_l^*	local POF
POF^*	approximated POF
n_{POF^*}	number of solutions in approximated POF
POF	true POF
$n_{POF'}$	number of sampled solutions in true POF
POF'	sampled solutions of true POF
S	unrestricted search space
τ	current iteration
τ_t	frequency of change
U	Set of utility functions
$\mathbf{x}$	solution vector
$\mathbf{x}_i$	decision variable
$\mathbf{x}_{min}$	lower bound of the feasible values for decision variable $\mathbf{x}$
$\mathbf{x}_N$	local optima
$\mathbf{x}^*$	global optimum solution
$\mathbf{x}^*(t)$	global optimum solution at time step t
$\mathbf{x}_{max}$	upper bound of the feasible values for decision variable $\mathbf{x}$

Appendix B

List of Acronyms

AIS
: artificial immune system. 172, 173, 191

CCEA
: cooperative-coevolution evolutionary algorithm. 6, 143, 151, 152, 154, 155, 177, 178

CI
: computational intelligence. 28, 124, 126, 144, 156, 169, 191, 192

COEA
: competitive-cooperative evolutionary algorithm. 177

D-QMOO
: dynamic queuing multi-objective optimizer. 187, 188

dCCEA
: dynamic CCEA. 176, 178, 179

dCOEA
: dynamic COEA. 177–179, 327–329, 335–337, 339, 340, 346, 353, 362, 363, 366

DE
: differential evolution. 7, 141, 156, 165, 172, 184

dMO-EGS
: dynamic MO-EGS. 175, 176, 191

dMO-EGS-PG
: dynamic MO-EGS with prediction gradient. 191

DMOA
: dynamic MOA. 84, 327, 329, 332, 334–337, 339, 340, 346, 353, 357, 362, 363, 365–367, 399, 401

DMOEA
: dynamic MOEA. 183, 189, 336, 337, 339, 363, 366

dMOEA
: dynamic MOEA. 178, 179

DMOO
: dynamic multi-objective optimisation. 2–7, 10, 11, 24, 27–31, 41, 42, 48, 54, 72, 74, 82, 84, 87, 88, 100, 101, 103, 107, 109, 110, 113, 114, 122, 124, 142, 143, 168–170, 172, 173, 175–177, 179, 180, 183, 191, 192, 201, 202, 206, 248, 249, 311, 322–326, 364–366, 368

DMOOP
: dynamic multi-objective optimisation problem. 1–7, 20, 25–31, 41–45, 47, 48, 50–54, 56–75, 77–87, 92, 101, 102, 105, 111–113, 117, 119, 121, 123, 168–177, 179–187, 189–192, 194, 195, 197–199, 201–204, 206, 209, 211, 213, 215–218, 220, 222, 223, 225, 230, 234, 237, 239, 251, 253–261, 271, 273–280, 282, 289, 291, 293, 295–298, 302, 309, 311–313, 315, 316, 323, 326–329, 334–339, 352, 362–367, 397–399, 401

DMOPSO
: dynamic MOPSO. 325, 327, 328, 334–340, 346, 352, 353, 362, 366

DNSGA-II
: dynamic NSGA-II. 170, 171, 190, 327–329, 334–340, 346, 352, 353, 362, 366

DSOO
: dynamic single-objective optimisation. 2, 10, 11, 21, 27, 107, 110

DSOOP
: dynamic single-objective optimisation problem. 2, 11, 21, 22, 25–27, 63, 168, 169

DVEPSO

dynamic VEPSO. 194

DVEPSO
: dynamic Vector Evaluated Particle Swarm Optimisation. 2–7, 121, 126, 143, 162, 192, 194–196, 198–202, 209, 222, 223, 225, 228, 248–251, 260, 261, 270, 288, 298, 322–325, 328, 329, 334–336, 338–340, 346, 352, 353, 362, 363, 365–367, 399

EA
: evolutionary algorithm. 2, 28, 143–145, 166, 169, 173, 176, 177, 179, 181, 182, 185, 191, 192

GA
: genetic algorithm. 6, 121, 126, 136–138, 141–145, 156, 170, 176, 184

GD
: generational distance. 91, 92, 102, 173, 179, 180, 182, 186

gIDG
: generalised immigrants-based diversity generator. 185

HV
: hypervolume. 98–100, 106, 110, 113, 188, 365

HVD
: hypervolume distance. 110

HVR
: hypervolume ratio. 99, 100, 189

IDMOEA
: individual diversity multi-objective optimization evolutionary algorithm. 174, 175

IGD
: inverse generational distance. 92

MA
: memetic algorithm. 181, 182

MO-EGS
: dynamic multi-objective gradient search. 175, 191

MOA
: multi-objective algorithm. 2, 3, 364, 365

MOEA
: multi-objective evolutionary algorithm. 144, 145, 155, 156, 171–174, 176, 178, 179, 182, 183, 187, 189, 190, 334

MOGA
: multi-objective genetic algorithm. 146, 155, 170

MOO
: multi-objective optimisation. 2, 7, 10, 11, 15, 16, 20, 27–30, 43, 72, 74, 83, 84, 87, 88, 91, 92, 100, 124, 143–145, 149, 151, 152, 155, 156, 163, 166, 170, 171

MOOP
: multi-objective optimisation problem. 2, 6, 7, 11, 14–20, 24, 27, 29, 30, 32, 37, 39, 56, 59, 64, 71, 72, 74, 75, 84, 88, 89, 99, 119, 128, 144, 151–153, 155, 156, 160, 163, 165–168, 182, 183, 187, 188, 195, 324

MOPSO
: multi-objective Particle Swarm Optimisation. 6, 143, 149–151, 155, 179, 362

MSOPS
: multiple single objective Pareto sampling. 171, 172

NSGA
: non-dominated sorting genetic algorithm. 146, 149, 155

NSGA-II
: non-dominated sorting genetic algorithm II. 6, 143, 146, 147, 149, 151, 155, 170–172, 183, 185, 187

PAES
: pareto archived evolution strategy. 151, 176, 183

POF
: Pareto-optimal front. 2–5, 7, 17–20, 25–27, 29–35, 37–52, 54–56, 58–64, 66–75, 77–83, 85, 86, 88, 89, 91, 96, 98–102, 104, 105, 110–115, 117, 118, 123, 124, 143–145, 151, 154, 155, 157, 163, 164, 168, 169, 171–174, 177–179, 181–183,

VEDE
: vector evaluated differential evolution. 7, 156, 165–167

VEDEPSO
: vector evaluated differential evolution particle swarm optimisation. 166, 167

VEGA
: vector evaluated genetic algorithm. 7, 144, 156, 158, 164–167

VEPSO
: vector evaluated particle swarm optimisation. 5, 7, 156, 158–167, 192, 194–196, 198, 199, 206, 208, 228, 248, 365, 366

Appendix C

Calculating the True POS and POF

This appendix discusses how *POS* and *POF* are determined for DMOOPs. Two examples are provided, namely FDA5 and FDA2 modified by Cámara *et al.* [17] [16] [138] referred to in this section as FDA2$_{Camara}$.

C.1 Example 1: FDA2$_{Camara}$

The FDA2$_{Cámara}$ DMOOP has two objective functions (refer to Section 3.2.1) and is defined as

$$\begin{cases} f_1(\mathbf{x_I}) = x_1 \\ g(\mathbf{x_{II}}) = 1 + \sum_{x_i \in \mathbf{x_{II}}} x_i^2 \\ h(\mathbf{x_{III}}, f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ where: \\ H(t) = z^{-\cos(\pi t/4)}, \;\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ H_2(t) = H(t) + \sum_{x_i \in \mathbf{x_{III}}} (x_i - H(t)/2)^2 \\ \mathbf{x_I} \in [0,1]; \;\; \mathbf{x_{II}}, \mathbf{x_{III}} \in [-1,1] \end{cases}$$

The goal when solving FDA2$_{Camara}$ is to minimise the two objective functions, namely f_1 and $f_2 = gh$. Since f_1 only depends on x_1 the true POF depends on f_2. In order to minimise gh, both g and h have to be minimised. h will be minimised if the term $\frac{f_1}{g}^{H_2(t)}$ is maximised (since this term is subtracted from 1). The term $\frac{f_1}{g}^{H_2(t)}$ is maximised if g is minimised (since f_1 is divided by g). g is minimised if the term $\sum_{x_i \in \mathbf{x_{II}}} x_i^2$ is minimised, i.e. if $\sum_{x_i \in \mathbf{x_{II}}} x_i^2$ is zero. Therefore, the optimal values for $x_i \in \mathbf{x_{II}}$ is

$x_i = 0$. If $\sum_{x_i \in \mathbf{x_{II}}} x_i^2 = 0$, $g = 1$. Replacing $g = 1$ into $f_2 = gh$, results in $f_2^* = 1 - f_1^{H_2(t)}$. In order to minise f_2^*, $H_2(t)$ has to be minimised. $H_2(t)$ is minimised if the term $\sum_{x_i \in \mathbf{x_{III}}} (x_i - H(t)/2)^2$ is minimised, which results in $H_2^*(t) = H(t)$. Therefore, the optimal values of $x_i \in \mathbf{x_{III}}$ is $x_i = \frac{H(t)}{2}$. Replacing H_2 in f_2^* with H_2^*, results in $f_2 = 1 - f_1^{H(t)}$.

Therefore, POF is $1 - f_1^{H(t)}$. The decision variable values that lead to POF is POS, namely $x_i = 0$, $\forall x_i \in \mathbf{x_{II}}$ and $x_i = \frac{H(t)}{2}$, $\forall x_i \in \mathbf{x_{III}}$.

C.2 Example 2: FDA5

FDA5 is a three-objective DMOOP [58] (refer to Section 3.2.1), defined as

$$\text{FDA5} = \begin{cases} Minimize : \mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, g(\mathbf{x_{II}}, t)), \ldots, \\ \qquad\qquad f_k(\mathbf{x}, g(\mathbf{x_{II}}, t))) \\ f_1(\mathbf{x}, g, t) = (1 + g(\mathbf{x_{II}}, t)) \prod_{i=1}^{M-1} \cos\left(\frac{y_i \pi}{2}\right) \\ f_k(\mathbf{x}, g, t) = (1 + g(\mathbf{x_{II}}, t)) \left(\prod_{i=1}^{M-1} \cos\left(\frac{y_i \pi}{2}\right)\right) \\ \sin\left(\frac{y_{M-k+1} \pi}{2}\right), \forall k = 1, \ldots, M-1 \\ \vdots \\ f_m(\mathbf{x}, g, t) = (1 + g(\mathbf{x_{II}}, t)) \prod_{i=1}^{M-1} \sin\left(\frac{y_1 \pi}{2}\right) \\ where : \\ g(\mathbf{x_{II}}, t) = G(t) + \sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2 \\ G(t) = |sin(0.5\pi t)|, \;\; t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ y_i = x_i^{F(t)}, \;\; \forall i = 1, \ldots, (M-1) \\ F(t) = 1 + 100 \sin^4(0.5\pi t) \\ \mathbf{x_{II}} = (x_M, \ldots, x_n) \\ x_i \in [0, 1], \; \forall i = 1, \ldots, n \end{cases} \tag{C.1}$$

In order to minimise FDA5, each objective function has to be minimised. Therefore, g has to be minimised, i.e. the term $\sum_{x_i \in \mathbf{x_{II}}} (x_i - G(t))^2$ has to be minimised. This results in $g = G(t)$ and POF of $\sum_{i=1}^{m} f_i^2 = (1 + G(t))^2$ (refer to [49]). The decision variable values that minimises g is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$, resulting in POS of $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$.

Appendix D

Additional Data and Figures for Conducted Experiments

This appendix provides additional data and figures of experiments discussed in Chapters 9, 10 and 11. Section D.1 presents performance measure values, p-values and figures with regards to the data of the experiments that were conducted to investigate the influence of various guide update approaches on the performance of DVEPSO (refer to Chapter 9). The performance measure values and p-values of the sensitivity analysis discussed in Chapter 10 are presented in Section D.2. Finally, Section D.3 presents the performance measure values and p-values of the DMOAs discussed in Chapter 11.

Due to space limitations, the tables and figures can be found on the included CD. Below a list is provided of the data that is available on the CD, as well as the file that contains the specific data.

D.1 Guide Update Approaches

The following additional data is provided on the CD:

- Tables containing performance measure values that were obtained by the various guide update approaches (guide-update-approaches-performance-measure-values.pdf)
- Figures of the wins and losses values obtained by the guide update approaches for each performance measure and DMOOP (guide-update-approaches-wins-and-

losses.pdf)

- Tables presenting the p-values of the Mann-Whitney U tests that were performed on the data obtained by the guide update approaches (guide-update-approaches-pvalues.pdf)

D.2 Sensitivity Analysis

The following additional data is provided on the CD with regards to various approaches used to manage boundary constraint violations:

- Tables containing performance measure values that were obtained by the boundary constraint management approaches (boundary-performance-measure-values.pdf)
- Tables presenting the p-values of the Mann-Whitney U tests that were performed on the data obtained by the boundary constraint violation management approaches (boundary-pvalues.pdf)

Additional data with regards to knowledge sharing approaches that can be found on the CD are:

- Tables containing performance measure values that were obtained by the knowledge sharing strategies (knowledge-performance-measure-values.pdf)
- Tables presenting the p-values of the Mann-Whitney U tests that were performed on the data obtained by the knowledge sharing approaches (knowledge-pvalues.pdf)

The following additional data is provided on the CD with regards to responses after a change in the environment occurs:

- Tables containing performance measure values that were obtained by the responses applied to particles of the sub-swarms (response-particles-performance-measure-values.pdf) and responses applied to the archive (response-archive-performance-measure-values.pdf)
- Tables presenting the p-values of the Mann-Whitney U tests that were performed on the data obtained by the responses applied to the particles (response-particles-pvalues.pdf) and responses applied to the archive (response-archive-pvalues.pdf)

D.3 Comparison of Dynamic Multi-objective Optimisation Algorithms

Additional data with regards to the comparison of DMOAs solving DMOOPs that can be found on the CD are:

- Tables containing performance measure values that were obtained by the DMOAs (comparison-performance-measure-values.pdf)
- Tables presenting the p-values of the Mann-Whitney U tests that were performed on the data obtained by the DMOAs (comparison-pvalues.pdf)

Appendix E

List of Publications

This appendix lists the publications that were derived from the research conducted for this thesis.

Journal Articles

M. Helbig and A.P. Engelbrecht. Dynamic multi-objective optimisation problems. Submitted for review.

M. Helbig and A.P. Engelbrecht. Performance measures for dynamic multi-objective optimisation algorithms. Submitted for review.

Conference Papers

M. Greeff and A.P. Engelbrecht. Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation. In *Proc. of IEEE World Congress on Computational Intelligence: IEEE Congress on Evoluationary Computation*, pages 2917–2924, Hong Kong, June 2008.

M. Helbig and A.P. Engelbrecht. Archive management for dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. In *Proc. of the*

IEEE Congress on Evolutionary Computation, pages 2047–2054, New Orleans, U.S.A., June 2011.

M. Helbig and A.P. Engelbrecht. Analyses of Guide Update Approaches for Vector Evaluated Particle Swarm Optimisation on Dynamic Multi-Objective Optimisation Problems. In *Proc. of IEEE World Congress on Computational Intelligence: IEEE Congress on Evoluationary Computation*, pages 2621–2628, Brisbane, June, 2012.

Book Chapters

M. Greeff and A.P. Engelbrecht. Dynamic multi-objective optimisation using PSO. In Nadia Nedjah, Leandro dos Santos Coelho, and Luiza de Macedo Mourelle, editors, *Multi-Objective Swarm Intelligent Systems*, volume 261 of Studies in Computational Intelligence, pages 105–123, Springer Berlin/Heidelberg, 2010.

M. Greeff and A.P. Engelbrecht. Dynamic multi-objective optimisation using PSO. In *Metaheuristics for Dynamic Optimization*, Springer, To be published.