Driving dynamic multi-objective optimizations constrained by decision-makers' preferences

by

Adekoya Adekunle Rotimi

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

Jan 2019

Publication data:

Adekoya Adekunle R. Driving Dynamic Multi-Objective Optimizations Constrained by Decision-Makers' Preferences. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, Jan 2019.

Electronic, hyperlinked versions of this dissertation are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

http://upetd.up.ac.za/UPeTD.htm

Driving dynamic multi-Objective optimizations constrained by decision-makers' preferences

by

Adekoya Adekunle Rotimi
E-mail: adekunleadekoya@gmail.com

Abstract

Dynamic multi-objective optimization problems (DMOOPs) are an interesting and a relatively complex class of optimization problems where elements of the problems, such as objective functions and/or constraints, change with time. These problems are characterized with at least two objective functions in conflict with one another. Sometimes, human decision-makers seek to influence ways (by restricting the search to a specific region of the Pareto-optimal Front (POF)) in which algorithms that optimize these problems behave by incorporating personal preferences into the optimization process. This dissertation proposes approaches that enable decision-makers to influence the optimization process with their preferences. The decision-makers' imparted preferences force a reformulation of the optimization problems as constrained problems, where the constraints are defined in the objective space. Consequently, the constrained problems are then solved using variations of constraint handling techniques, such as penalization of infeasible solutions and the restriction of the search to the feasible region. The pro-posed algorithmic approaches' performance are compared using standard performance measures for dynamic multi-objective optimization (DMOO) and newly proposed mea-sures. The proposed measures estimate how well an algorithm is able to find solutions in the objective space that best reflect the decision-maker's preferences and the pareto-optimality goal of DMOO. This dissertation also proposes a new differential evolution algorithm, called dynamic differential evolution vector-evaluated non-dominated sorting (2DEVENS). 2DEVENS combines elements of the dynamic non-dominated sort genetic

algorithm version II (DNSGA-II) and the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm to drive the search for solutions.

The proposed 2DEVENS algorithm compared favorably with other nature-inspired algorithms that were used in the studies carried out for this dissertation. The proposed approaches used in incorporating decision-makers' preferences in the optimization process also demonstrated good results.

Keywords: Constrained optimization, dynamic multi-objective optimization, decision-maker preference incorporation, differential evolution, evolutionary and nature-inspired computation, benchmark functions and performance measures

Supervisor : Dr. Mardé Helbig
Department : Department of Computer Science
Degree : Master of Science

"So do all who live to see such times. But that is not for them to decide. All we have to decide is what to do with the time that is given us."

J.R.R. Tolkien, The Fellowship of the Ring.

"A wise man makes his own decisions, an ignorant man follows the public opinion. . . ."

Chinese Proverb.

Acknowledgements

I will like to express my gratitude to the following people for their assistance during the production of this dissertation:

- Dr. Mardé Helbig, my supervisor, for her insight, thorough review of my write-ups, encouragement and unrelenting efforts to find me financial support which really kept me going.
- Prof Andries Engelbrecht, the leading figure in our research group, for his unusual understanding and support regarding my software development project, and for the wealth of materials and ideas he has made available for us to learn from.
- My mother, for her prayers and belief in me.
- My wife, for her prayers and belief in me.
- My friends and research colleagues, for the beautiful and stimulating research environments they made possible for me.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Keep it simple. Eliminate the extraneous materials and focus on the critical success factors. These are the defining characters of quality decision making.

Decision-making as a process results from the fact that decision-makers (decision-makers) have to contend with many objectives. This problem is further compounded by the fact that decision-making environments are ever changing, and dynamic environments present unique challenges to decision-making.

In this research, decision-making is studied in the context of nature-inspired and evolutionary algorithms. The algorithms search for the optimal solutions to optimization problems. The optimization problems are characteristically defined by one or more objective functions, and zero or more constraints. In order to model the dynamic nature of the environment, objective functions incorporate a temporal parameter.

The class of optimization problems described above is referred to as dynamic multi-objective optimization problems (DMOOPs). There are many DMOOPs in the real-world [9] [26] [35] [47] [61] [78] [137] [96] [116] [133] [139] [169] . Sometimes, solving real-world optimization problems may be computationally expensive from an experimental point of view. Also, researchers may be interested in comparing the performance of dynamic multi-objective optimization algorithms (DMOAs) on problems that are easy to understand, whose optimal solutions are known, etc. Therefore, researchers have invented artificial problems, also called benchmark functions. Benchmark functions are well researched in [61] [73] [84] [88] [97] [106].

When solving dynamic multi-objective optimization (DMOO), formulating DMOOPs is one step. Another required step is to design the search algorithms. Search algorithms search for the optimal solutions of the formulated DMOOPs. There are a class of iterative procedures employed in the search, which draw on the mechanisms of evolution, such as biological evolution [200]. Examples of such biological evolutionary procedures are genetic algorithm (GA) [19] [69] [70] [94] [154] [197], particle swarm optimization (PSO) [112] and differential evolution (DE) algorithms [148] [180]. The evolutionary algorithms, when applied to the search of solutions to DMOOPs, are collectively referred to as DMOAs.

Solutions found by DMOAs are typically large in number and may overwhelm decision-makers, who need to select a single solution. Therefore, decision-makers seek an interesting subset of the optimal solutions which satisfy target criteria. A decision-maker may perform the subsetting after the optimal solutions have been found by the DMOAs. He may also specify the subsetting rules before the DMOAs start the search process, and lastly he may interrupt the DMOAs while computing the solutions in order to specify the interesting subset of the found optimal solutions. The three approaches discussed above and employed by a decision-maker in specifying the interesting subset of the optimal solutions are respectively called a priori, posteriori and interactive methods of preference incorporation [32] [36] [37] [47] [62] [76] [101] [162] [181] [185].

1.1 Objectives

This dissertation seeks to achieve the following objectives:

- To provide a literature review of the trend in decision-maker preference incorporation approaches - and discuss any existing applications to DMOOPs.
- To propose a new preference incorporation approach in the context of solving DMOOPs using DMOAs.
- To propose a new DE-based DMOA.
- To perform a comparative study of the performance of the proposed DE-based DMOA with other nature-inspired algorithms, such as the dynamic vector-evaluated

particle swarm optimization (DVEPSO) algorithm and the dynamic non-dominated sort genetic algorithm version II (DNSGA-II).

- To propose new algorithms exemplifying different ways of incorporating the proposed preference incorporation approach into the proposed DE-based DMOA.
- To perform a comparative study of the performance of the new algorithms that incorporate the decision-maker's preferences.
- To propose new performance measures for DMOAs. The proposed performance measures will reflect the effectiveness of a DMOA in finding optimal solutions to optimizations problems in which decision-makers' preferences are incorporated.

1.2 Contributions

The contributions of this dissertation are summarised as follows:

- The conclusion that no work has been done before now on how to incorporate decision-maker preferences into DMOAs.
- The successful design and implementation of a new DE-based DMOA called the dynamic differential evolution vector-evaluated non-dominated sorting (2DEVENS) algorithm.
- The discovery that 2DEVENS outperforms DVEPSO and DNSGA-II for the two important performance measures of accuracy and stability.
- An effective new approach to specify decision-maker preferences by means of a bounding-box framework.
- Successfully designing and implementing three new DMOAs, namely:
  - proportionate-penalty algorithm (ALG:1)
  - death-penalty algorithm (ALG:2)
  - restrict-search-to-feasible-region algorithm (ALG:3).

- ALG:1, ALG:2 and ALG:3 incorporate the bounding-box approach for decision-maker preferences. The algorithms are based on 2DEVENS.
- A full-fledged comparative analysis of the three new DMOAs. The analysis showed the relative strengths of each algorithm with regards to standard performance measures for DMOOPs and some of the new performance measures developed in this work.
- A new set of performance measures for DMOOPs that incorporate decision-maker preferences defined using the proposed bounding-box framework in this dissertation. For the purpose of these new performance measures, the word decision is equivalent to a solution. The new performance measures are as follows:
  - Number of Non-Violating Decisions (nNVD): measures the number of solutions that fall within the decision-maker's preference set. The higher the value of this measure, the better.
  - Spread of Non-Violating Decisions (sNVD): measures the spread of solutions within the preferred set. The higher the value of this measure, the better the performance of the algorithm.
  - Number of Violating Decisions (nVD): measures the number of violating solutions in the archive, i.e. that do not lie within the preferred bounding box. The lower the value of this measure, the better the performance of the algorithm.
  - Total Deviation of Violating Decisions (dVD): measures the total deviation from the preferred bounding box for all violating solutions in the archive. The lower the value of this measure, the better the algorithm's performance.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows:

- **Chapter 2** discusses mathematical optimization, and static and dynamic multi-objective optimization. It also discusses the benchmark functions, performance

measures and statistical analysis used in this research.

- **Chapter 3** covers the three classes of nature-inspired and evolutionary algorithms studied in this research.
- **Chapter 4** discusses decision making, as well as normative and positive theories of decision making. Also, it presents some formalizations of decision problems. The chapter also discusses the notions of preferences and multi-criteria decision evaluations and designs.
- **Chapter 5** covers the first experimental study carried out in this research work. In the chapter, the performance of 2DEVENS is compared with two nature-inspired algorithms for selected optimization problems.
- **Chapter 6** covers the second experimental study carried out in this research work. It presents a comparative study, like that of **Chapter 5**. However, in addition to the problems and experimental configurations studied in **Chapter 5**, a new set of benchmark functions, such as HE1 and HE3, and new experimental configurations, were studied in this chapter.
- **Chapter 7** presents the third experimental study carried out in this research work. It discusses the proposed approaches for incorporating decision-makers' preferences in DMOAs. It discusses a preference specification approach called a bounding box. It presents algorithms that use the bounding box approach to solve optimization problems that incorporate decision makers' preferences. A new set of performance measures are also presented in this chapter. Algorithms proposed in this chapter are compared based on the new measures of performance.
- **Chapter 8** presents a summary of the research, as well as possible directions for future work.

The following appendices are included, containing a number of lists with relevant information for quick referencing:

- **Appendix A** provides a list of the important acronyms used or newly defined in the course of this work, as well as their associated definitions.

- **Appendix B** lists and defines the mathematical symbols used in this work, categorised according to the relevant chapter in which they appear.
- **Appendix C** lists the publications derived from this work.
- **Appendix D** presents the detailed results of the third experimental study.

Chapter 2

Optimization

Decide not rashly. The decision made Can never be recalled. The gods implore not, Plead not, solicit not; they only offer Choice and occasion, which once being passed Return no more. Dost thou accept the gift?

Henry Wadsworth Longfellow.

Optimization problems are found in fields as diverse as science, engineering, economics, business and many others. Driven by the need to formalize the process of optimization, this dissertation focuses on mathematical optimization, which is the dominant form of optimization defined by methods from mathematics, computer science and operations research. Mathematical optimization is the selection of a best element, with regards to some criteria, relative to some other alternative elements. Put differently, mathematical optimization strives to find the elements, or characteristics, of a system that offer the best values for target criteria. The target criteria of mathematical optimization problems are typically represented by real-valued functions, whose domains are the set of allowable values for the system elements or characteristics. The real-valued functions that model the target criteria are called by various names such as: cost functions, loss functions, payoff functions, objective functions, utility functions, and energy functions. Henceforth, target criteria shall be referred to as objectives and the corresponding real-valued functions as objective functions. In some optimization problems, there is a single criterion. Such a single-criterion optimization problem is called a single-objective

problem. Notable examples of single-objective problems are: knapsack problem, travelling salesman problem, assignment problem and transportation problem [14] [71] [119] [126] [211]. These problems are well-covered in operations research literature.

There are however, a class of problems with two or more objectives, where two or more functions are required to model the objectives. This class of problems is called multi-objective optimization problems (MOOPs). MOOPs provide interesting research opportunities, and they are as widespread as their single-objective counterparts.

Optimization problems have been classified using various criteria. Some of the classifications include continuous versus discrete, constrained versus unconstrained, global versus local and stochastic versus deterministic [135]. The class of problems studied in this dissertation is continuous, unconstrained (in objective space), global and deterministic in nature.

The rest of the chapter is organized as follows: Section 2.1 discusses MOOPs, while dynamic MOOPs, or DMOOPs, are discussed in Section 2.2. Artificial DMOOPs, also called benchmark functions, are covered in Section 2.3. Performance measures are discussed in Section 2.4. A review of some of the statistical analyses approaches employed in the dissertation is presented in Section 2.5. Section 2.6 presents a generic DMOA. A summary of the chapter is presented in Section 2.7.

2.1 Multi-Objective Optimization

MOOPs, unlike single-objective problems, present an additional level of complexity as a result of the greater number of objectives in the problems. However, these objectives are usually in conflict with one another, thereby making the process of finding a single optimal solution an impossible task [33]. Trade-off solutions are therefore the norm. The Pareto-dominance relation [140] is a popular operator used to compare the trade-off solutions.

The general MOOP is defined as follows [56]:

$$\begin{aligned} &\text{Minimize:} \quad \mathbf{f(x)} = (f_1(\mathbf{x}), ..., f_{n_k}(\mathbf{x})) \\ &\text{Subject to:} \\ &g_m(\mathbf{x}) \leq 0 \quad m = 1, ...n_g \\ &h_m(\mathbf{x}) = 0 \quad m = n_g + 1, ...n_g + n_h \\ &\mathbf{x} \in [x_{min}, x_{max}]^{n_x} \end{aligned} \tag{2.1}$$

Where n_k is the number of objective functions, n_g is the number of inequality constraints and n_h is the number of equality constraints.

$\mathbf{x} = (x_1, ..., x_{n_x}) \in \mathcal{S}$ is referred to as a decision vector and $\mathcal{S} \subseteq R^{n_x}$ is the n_x dimensional search space. $\mathcal{F} \subseteq S$ is the feasible search space. With no constraints, the feasible search space is the same as the search space.

A single-objective function, f_k, is defined as follows:

$$f : R^{n_x} \longrightarrow R$$

From Equation 2.1 $\mathbf{f(x)} = (f_1(\mathbf{x}), ..., f_{n_k}(\mathbf{x})) \in \mathcal{O} \subseteq R^{n_k}$ is an objective vector containing n_k objective function evaluations, and $\mathcal{O}$ is referred to as the objective space. The search space $\mathcal{S}$ is also referred to as the decision space. $\mathbf{x} \in [x_{min}, x_{max}]$ are the boundary constraints. Solutions of the multi-objective problem, x^*, are in the feasible space, $\mathcal{F}$.

Due to the conflicting nature of the objectives of a MOOP, trade-off solutions, x^*, are usually sought [33]. Trade-off solutions are such that no objective can be improved without degrading the solutions of at least one other objective. These solutions are referred to as non-dominated solutions and the set of such solutions is called the Pareto-optimal Set (POS). The set of corresponding objective vectors of these solutions is called the Pareto-optimal Front (POF).

Following are concepts and definitions considered relevant to understanding the topic of multi-objective optimization (MOO) and the related problems of DMOOPs [48] [56] [57] [84]:

Definition 2.1. Decision Vector Domination: A decision vector x_1 dominates another decision vector x_2, denoted by $x_1 \prec x_2$, if and only if

1. x_1 is at least as good as x_2 for all the objectives, i.e. $f_k(x_1) \leq f_k(x_2)\ \forall k$.
2. x_1 is strictly better than x_2 for at least one objective, i.e. $\exists k$ such that $f_k(x_1) < f_k(x_2)$.

Definition 2.2. Weak Vector Domination: A decision vector x_1 weakly dominates another decision vector x_2, denoted by $x_1 \preceq x_2$, if and only if
x_1 is at least as good as x_2 for all the objectives, i.e. $f_k(x_1) \leq f_k(x_2)\ \forall k$.

Definition 2.3. Pareto Optimal: A decision vector x^* is Pareto optimal if there does not exist a decision vector $x \neq x^*$ that dominates x^*.

Definition 2.4. Pareto Optimal Set: The Pareto Optimal set P^* is a set of Pareto optimal decision vectors. It is mathematically defined as follows:
$P^* = \{x^* \in \mathcal{F} | \nexists x \in \mathcal{F} : x \prec x^*\}$

Definition 2.5. Local Pareto Optimal Set: Let $N \subset \mathcal{F}$ be a set of solutions in the neighborhood of P^*, the Pareto-optimal set. P^* is a local Pareto-optimal set if there exists a solution $x \in N$ such that $\mathbf{f}(x) \prec \mathbf{f}(x^*)$ for some $x^* \in P^*$.

Definition 2.6. Global Pareto Optimal Set: P^* is a global Pareto-optimal set if there does not exist a solution $x \in \mathcal{S}$ in the search space such that $\mathbf{f}(x) \prec \mathbf{f}(x^*)\ \forall\ x^* \in P^*$.

Definition 2.7. Pareto Optimal Front: The Pareto Optimal Front PF^* is a set of objective vectors that correspond to the Pareto optimal set P^*. It is mathematically defined as:
$PF^* = \{\mathbf{f} = (f_1(x^*), ..., f_{n_k}(x^*))\}, \forall x^* \in P^*$

2.2 Dynamic Multi-Objective Optimization

MOOPs that reflect temporal considerations are referred to as DMOOPs [9] [26] [35] [47] [61] [78] [137] [96] [116] [133] [139] [169]. This section discusses DMOOPs. A review of

important concepts and definitions is presented in Section 2.2.1. Section 2.2.2 presents environment types which are used in characterizing DMOOPs.

2.2.1 A Brief Review and Definitions

Finding solutions to DMOOPs poses a unique set of challenges, since algorithms that search for the solutions must be able to cope with the changing environments that characterize DMOOPs. DMOOPs have objective functions and/or constraints that change over time [10] [87] [105] [134]. Therefore, algorithms searching for solutions to these problems must be capable of tracking changes in the optimal solutions and their associated objective vectors.

The general DMOOP is formally defined as follows [84]:

$$\begin{aligned} &\text{Minimize:} \quad \mathbf{f(x,t)} = (f_1(\mathbf{x},t), ..., f_{n_k}(\mathbf{x},t)) \\ &\text{Subject to:} \\ &g_m(\mathbf{x},t) \leq 0 \quad m = 1, ...n_g \\ &h_m(\mathbf{x},t) = 0 \quad m = n_g + 1, ...n_g + n_h \\ &\mathbf{x} \in [x_{min}, x_{max}]^{n_x} \end{aligned} \tag{2.2}$$

In Equation 2.2, t represents time. All other parameters in the equation are as defined in Equation 2.1.

Definition 2.8. Changing Pareto Optimal Front: The goal of a DMOA is to track the changing POF over time. The changing POF is defined as follows:

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

2.2.2 Dynamic Environment Types

The changing environment of a DMOOP is an important classification criterion. Good DMOAs are those which are very effective in tracking the changes. The general classification of the changing environment is discussed in this section. Farina et al [61] classified the environments of a DMOOP as presented in Table 2.1:

1. Type I environment: POS changes, but the POF remains unchanged.

2. Type II environment: POS and POF change.

3. Type III environment: POS remains unchanged, but POF changes.

4. Type IV environment: Both POS and POF do not change.

Changes in the environment of a DMOOP may impact the POF in any of the following ways [84]:

1. Existing solutions in the POF become dominated and therefore are excluded from the POF.

2. The shape of the POF remains the same, but its location in the objective space changes 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:

   (a) Change shape from convex to concave, or vice versa.

   (b) Change shape from continuous to disconnected front or vice versa.

4. The density of the solutions in the POF changes over time:

   (a) The solutions in the POF may become more/less dense.

   (b) The number of solutions in the POF may become more/less.

Table 2.1: Type of DMOOP Environments.

	POS	
POF	NO CHANGE	CHANGE
NO CHANGE	TYPE IV	TYPE I
CHANGE	TYPE III	TYPE II

2.3 Benchmark Functions

This section discusses benchmark functions. Section 2.3.1 presents what they mean and some of the ideas motivating their constructions. Section 2.3.2 presents the ZDT benchmark function. DTLZ is discussed in Section 2.3.3. Dynamic benchmark functions are presented in Section 2.3.4.

2.3.1 Construction of Benchmark Functions

Benchmark functions are the artificial counterparts of the real-world MOOPs. Real-world MOOPs present various difficulties to algorithms, and algorithms' effectiveness are evaluated on their abilities to overcome those difficulties. Benchmark functions are typically used for the purpose of comparing algorithms on their abilities to overcome the difficulties presented by MOOPs [61] [73] [88] [84] [97] [106]. For instance, when there is a change in the environment, the POF may change in shape, location, or in some other ways. An effective algorithm should be able to track the changes and converge to the new POF [84].

Benchmark functions are expected to possess some ideal characteristics, which usually guide their construction. Such characteristics include the following [49]:

1. Ease of construction.
2. Scalability in terms of the number of decision variables and the number of objective functions.
3. Present a POF with a known shape and location, which is easy to understand.
4. Present known difficulties, such as flat regions and local optima, to algorithms trying to converge to the true POF.
5. Make the prospect of obtaining a well diversified set of optimal solutions difficult to achieve by algorithms.

An algorithm may experience difficulties in converging to the true POF, i.e. the theoretically correct optimal solutions to a problem, because of the following features that may be present in the problem [34] [45]:

1. Multi-modality, i.e. the presence of multiple global POFs.
2. Deception, i.e. attraction of algorithms to local POFs.
3. Isolated optimum, e.g. a flat region that may force an algorithm to get stuck in a sub-optimal region of the search space.
4. Collateral noise, e.g. a situation where the low-order building blocks, in a solution vector, that may lead to the true optimum are improperly evaluated due to the interference of some other components of a solution vector.

While convergence to the true POF is one of the main goals of any algorithm employed in the search for solutions of MOOPs, maintaining diversity of solutions in the found POF is another major goal. However, algorithms may experience the following difficulties in achieving the goal of diversity [45]:

1. Convexity or non-convexity in the POF.
2. Discontinuity in the POF, and
3. Non-uniform distribution of solutions in the POF.

2.3.2 ZDT Benchmark functions

Zitzler, Deb and Thiele [45] proposed an N-variable tunable two-objective problem as follows:

$$\begin{aligned}
&\text{Minimize:} \quad \mathbf{f} = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}})) \\
&\text{Subject to:} \\
&\quad f_2 = g(\mathbf{x_{II}}) \, . \, h(f_1(\mathbf{x_I}), g(\mathbf{x_{II}})) \\
&\quad \mathbf{x_I} = (x_1, ..., x_m) \\
&\quad \mathbf{x_{II}} = (x_{m+1}, ..., x_N)
\end{aligned} \tag{2.3}$$

Where f_1 , $g > 0$. By choosing appropriate values for f_1, g and h, problems with specific features can be created:

1. An appropriate h function affects convexity or the discontinuity in the POF.
2. A difficult g function, e.g. multi-modal, deceptive and others, affects convergence to the true POF.
3. An appropriate f_1 (non-linear or multi-dimensional) function affects diversity of the POF solutions.

Deb et al [45] consequently created six benchmark functions based on Equation 2.3 and the construction guidelines in Section 2.3.1. Also, each of the ZDT functions addresses one of the difficulties faced by algorithms searching for solutions of MOOPs. ZDT problems are, however, two objective problems [217]. For further details on the six benchmark functions, including their mathematical definitions, and their POSs and POFs, refer to [45] [84] [88].

2.3.3 DTLZ Benchmark functions

Deb, Thiele, Laumanns and Zitzler developed a set of benchmark functions, referred to as DTLZ benchmarks [49]. This section discusses the two approaches and the benchmark function generator used in developing the set of DTLZ functions.

Definition 2.9. Spherical Coordinates Approach:

Deb et al [49] created a test problem where the POF lies in the first quadrant of a sphere of radius one and all objective functions take non-negative values. Using spherical coordinates (θ, γ, $r = 1$), the POF can be defined as follows:

$$\begin{aligned} f_1(\theta,\gamma) &= cos\,\theta\, cos(\gamma + \frac{\pi}{4}) \\ f_2(\theta,\gamma) &= cos\,\theta\, sin(\gamma + \frac{\pi}{4}) \\ f_3(\theta,\gamma) &= sin\,\theta \end{aligned} \tag{2.4}$$

where:

$$0 \leq \theta \leq \frac{\pi}{2}, -\frac{\pi}{4} \leq \gamma \leq \frac{\pi}{4}$$

If all the objective functions are minimized, any two points on the surface are non-dominated to each other. If the rest of the objective search space is constructed above

this surface, the unit sphere constitutes the POF. This can be achieved by creating the rest of the search space parallel to the surface defined in Equation 2.4 as follows:

Minimize:

$$
\begin{aligned}
&f_1(\theta,\gamma) = (1+g(r))\,cos\,\theta\,cos(\gamma+\frac{\pi}{4})\\
&f_2(\theta,\gamma) = (1+g(r))\,cos\,\theta\,sin(\gamma+\frac{\pi}{4})\\
&f_3(\theta,\gamma) = (1+g(r))\,sin\,\theta
\end{aligned}
\tag{2.5}
$$

where:

$$0 \leq \theta \leq \frac{\pi}{2}, -\frac{\pi}{4} \leq \gamma \leq \frac{\pi}{4}$$

$$g(r) \geq 0$$

The POS for the problem defined by Equation 2.5 is $0 \leq \theta^* \leq \frac{\pi}{2}, -\frac{\pi}{4} \leq \gamma^* \leq \pi/4$, $g(r^*) = 0$.

Definition 2.10. Constrained Surface Approach: This is another approach employed by Deb et al to construct DTLZ benchmark functions [49]. A search space is defined as follows:

Minimize:

$$
\begin{aligned}
&f_1(\mathbf{x})\\
&.\\
&.\\
&.\\
&f_{n_k}(\mathbf{x})
\end{aligned}
\tag{2.6}
$$

Subject to:

$$0 \leq f_i^L \leq f_i(\mathbf{x}) \leq f_i^U,\ i = 1,...,n_k$$

For the problem defined by Equation 2.6, f_i^L and f_i^U refer to the lower and the upper bounds of the function f_i respectively. The POS corresponding to Equation 2.6 is $(f_1^L,...,f_{n_k}^L)^T$.

The problem can be made more interesting by adding constraints (linear and non-linear), such as the following: $g_j(f_1,...,f_{n_k}) \geq 0,\ j = 1,...,n_g$, where f_i is an objective function defined in Equation 2.6.

Definition 2.11. Benchmark Function Generator: Deb [48] suggested a generic function generator based on the constraint surface approach. The generator was used by Deb et al to develop the set of DTLZ benchmark functions [49]. The generator is defined as follows:

Minimize:

$f_1(\mathbf{x}_1)$

.

.

.

$f_{n_k-1}(\mathbf{x}_{n_k-1})$

$f_{n_k}(\mathbf{x}_{n_k}) = g(x_{n_k})\,.\,h(f_1, ..., f_{n_k}, g)$

where:

$x \equiv (x_1, ..., x_{n_k})$, i.e. x is partitioned into disjoint, n_k sets of decision variables, and

$x_i \in R^{|x_i|}$, $i = 1, ..., n_k$

(2.7)

2.3.4 Dynamic Multi-Objective Benchmark functions

Benchmark functions discussed in this section are dynamic, in addition to possessing multiple objectives. They are constructed by extending the construction rules suggested in [45] [49] [217]. Unlike the proposals in [192], the construction rules on which the dynamic benchmark functions of this section are based, facilitate a systematic construction of problems that present different difficulties (refer to Section 2.3.1) to algorithms. Dynamic problems are further characterized by the possibility of transforming from one difficulty to another as the problem environment changes.

The following is a generic two-objective benchmark function based on ZDT [45]:

$$\begin{aligned} &\text{minimize:} \quad \mathbf{f} = (f_1(\mathbf{x_I}, t), f_2(\mathbf{x_{II}}), t) \\ &\text{where:} \\ &\quad f_2 = g(\mathbf{x_{II}}, t)\,.\,h(\mathbf{x_{III}}, f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}), t) \end{aligned} \tag{2.8}$$

Farina et al [61] created the FDA suite of benchmark functions. The FDA functions are based on the generic test problem defined by Equation 2.8. The functions exhibit the desirable properties of test problems: ease of construction, scalability in terms of decision variables and objective functions, and a known POF, among others. Further details of the FDA functions can be found in [61] [88]. However, the FDA functions used in this research are briefly presented next.

$$\text{FDA1} = \begin{cases} \text{minimize} \quad \mathbf{f} = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}}), t) \\ \quad f_1(\mathbf{x_I}) = x_I \\ \quad f_2 = g(\mathbf{x_{II}}, t)\,.\,h(f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}), t) \\ \quad g(\mathbf{x_{II}}, t) = 1 + \sum\limits_{x_i \in x_{\mathbf{II}}} (x_i - G(t))^2 \\ \quad h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \\ \quad \text{where:} \\ \quad G(t) = sin(0.5\pi t),\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \quad x_i \in [0,1],\ x_{\mathbf{II}} = \mathbf{x} \setminus \mathbf{x_I},\ x_{\mathbf{II}} \in [-1,1]^{n_x - 1} \end{cases} \tag{2.9}$$

FDA1 is a type I DMOOP. Its POS changes with time, while the POF remains unchanged. The true POF is $1-\sqrt{f_1}$ and is convex shaped. The POS is $x_i = G(t), \forall x_i \in \mathbf{x_{II}}$.

$$\text{FDA2} = \begin{cases} \text{minimize} \quad \mathbf{f} = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}}), t) \\ f_1(\mathbf{x_I}) = x_I \\ f_2 = g(\mathbf{x_{II}}, t) \,.\, h(\mathbf{x_{III}}, f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}), t) \\ g(\mathbf{x_{II}}) = 1 + \sum\limits_{x_i \in x_{\mathbf{II}}} x_i^2 \\ h(f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H_2(t)} \\ \text{where:} \\ H(t) = 0.75\,(1 + sin(0.5\pi t))\,,\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ H_2(t) = (H(t) + \sum\limits_{x_i \in x_{\mathbf{III}}} x_i - H(t)^2)^{-1} \\ x_i \in [0,1],\; x_{\mathbf{II}},\; x_{\mathbf{II}} \subseteq \mathbf{x} \setminus \mathbf{x_I} \in [-1,1]^{n_x - 1} \end{cases} \tag{2.10}$$

The POS and POF of FDA2 change over time. The benchmark function is a type II DMOOP. Therefore, FDA2's POF changes from convex to concave. The POS is $x_i = 0,\ \forall x_i \in \mathbf{x_{II}}$ and $x_i = H(t),\ \forall x_i \in \mathbf{x_{III}}$. The POF is $1 - f_1^{H(t)^{-1}}$.

$$\text{FDA3} = \begin{cases} \text{minimize} \quad \mathbf{f} = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}}), t) \\ f_1(\mathbf{x_I}) = \sum\limits_{x_i \in x_{\mathbf{I}}} x_i^{F(t)} \\ f_2 = g(\mathbf{x_{II}}, t) \,.\, h(f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}), t) \\ g(\mathbf{x_{II}}, t) = 1 + G(t) + \sum\limits_{x_i \in x_{\mathbf{II}}} (x_i - G(t))^2 \\ h(f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{0.5} \\ \text{where:} \\ G(t) = |sin(0.5\pi t)| \\ F(t) = 10^{2sin(0.5\pi t)},\; t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ \mathbf{x_{I}}_j\,, \mathbf{x_{II}}_j \in [0,1] \end{cases} \tag{2.11}$$

FDA3 is a type II DMOOP, where both the POF and the POS change over time. The POF is convex and is defined as $(1+G(t))\left(1-\sqrt{\frac{f_1}{(1+G(t))}}\right)$. The POS is $x_i = G(t),\ \forall x_i \in \mathbf{x_{II}}$.

$$
\text{FDA5} = \begin{cases}
\text{minimize} \quad \mathbf{f} = (f_1(\mathbf{x}, g(\mathbf{x_{II}}, t), ..., f_k(\mathbf{x}, g(\mathbf{x_{II}}, t))) \\
f_1 = (1 + g(\mathbf{x_{II}}, t))\left(\prod_{i=1}^{M-1} cos(y_i \frac{\pi}{2})\right) \\
f_k = (1 + g(\mathbf{x_{II}}, t))\left(\prod_{i=1}^{M-1} cos(y_i \frac{\pi}{2})\right) \\
sin(y_{M-k+1}\frac{\pi}{2}),\ \forall k = 2, ..., M-1 \\
... \\
f_M = (1 + g(\mathbf{x_{II}}, t))sin(y_1 \frac{\pi}{2}) \\
\text{where:} \\
g(\mathbf{x_{II}}, t) = G(t) + \sum_{x_i \in x_{\mathbf{II}}} (x_i - G(t))^2 \\
G(t) = |sin(0.5\pi t)| \\
y_i = x_i^{F(t)},\ \forall i = 1, ..., M-1 \\
F(t) = 1 + 100 sin^4(0.5\pi t) \\
\mathbf{x_{II}} = (x_M, ..., x_{n_x}) \\
x_i \in [0, 1],\ \forall i = 1, ..., n_x
\end{cases} \tag{2.12}
$$

FDA5 is a type II problem, with POS and POF changing over time. The density of solutions also change over time. The non-convex POF of a three-objective FDA5 is $\sum_{i=1}^{3} f_i^2 = (1 + G(t))^2$. The POS is $x_i = G(t),\ \forall x_i \in \mathbf{x_{II}}$.

Goh and Tan [73] proposed dMOP2, a type I dynamic multi-objective benchmark function whose POF changes from convex to concave over time. The function is defined as follows:

$$\text{dMOP2} = \begin{cases} \text{minimize} \quad \mathbf{f} = (f_1(\mathbf{x_I}), f_2(\mathbf{x_{II}}), t) \\ f_1(\mathbf{x_I}) = x_I \\ f_2 = g(\mathbf{x_{II}}, t) \,.\, h(f_1(\mathbf{x_I}, t), g(\mathbf{x_{II}}), t) \\ g(\mathbf{x_{II}}, t) = 1 + 9 \sum\limits_{x_i \in x_{\mathbf{II}}} (x_i - G(t))^2 \\ h(f_1, g, t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ \text{where:} \\ H(t) = 0.75 sin(0.5\pi t) + 1.25 \\ G(t) = sin(0.5\pi t),\ t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \\ x_i \in [0, 1] \end{cases} \tag{2.13}$$

The POF of dMOP2 is $1 - f_1^{H(t)}$, while its POS is $x_i = G(t)$, $\forall x_i \in \mathbf{x_{II}}$.

2.4 Performance Measures

When algorithms find solutions of optimization problems, the solutions are usually approximations. Therefore, a means is required to evaluate the quality of the approximated solutions. A set of solutions is preferred, or of good quality, if the approximated solutions are very close to the true solutions of the problem being solved. Also, a quality solution set will be well diversified. Veldhuizen et al [193] provided one of the earliest quantitative approaches for comparing the quality of solutions in MOOPs, providing a radical departure from the earlier approaches using graphical plots. Meanwhile, the literature now contain many different proposals on how the quality can be measured.

Algorithms are evaluated on their ability to obtain quality approximations of the true solutions. Performance measures are quantitative metrics that measure how effective an algorithm is in terms of the ability to find a good set of solutions. The measures are based on defined criteria of effectiveness, such as the number of non-dominated solutions found, the closeness of the found Pareto-optimal solutions to the true Pareto-optimal solutions and the diversity of solutions in the found set of Pareto-optimal solutions.

Performance measures are mathematical operators, or functions [216] [219]. As operators, they can take one argument, and are then referred to as unary operators [44] [193]. The arguments of the operators, in this instance, are POFs. Binary operators have also been proposed and studied [81] [215]. However, Fonseca et al [77] proposed an entirely different approach, based on the attainment function, evaluating the quality of the solution set. While this dissertation focuses on DMOOPs, interested readers may refer to [6] [132] [165] for measures dealing with dynamic single-objective problems and [16] [27] [115] for measures dealing with static MOOPs.

Definition 2.12. Performance Measure: Zitzler [216] defined a performance measure as follows:

$$P : \Omega^m \longrightarrow R \tag{2.14}$$

According to Definition 2.14, a performance measure maps m approximated POFs to a real number $P(POF_1^*, POF_2^*, ..., POF_m^*)$.

In order to evaluate the quality of performance measures, Hansel and Jaszkiewicz [81] proposed an outperformance relation. Outperformance relations are related to another important concept, namely utility functions. A utility function quantifies the numerical strength associated with a Pareto-optimal vector by a decision-maker. Utility functions are quantitative estimates of a decision-maker's preferences. Mathematically, a utility function is defined as:

$$u : R^j \longrightarrow R \tag{2.15}$$

In Definition 2.15, u is the utility function. Its domain is the set of Pareto-optimal vectors. Each element in the domain of the function is of dimension j. The codomain of the function is a real number. For minimization problems, which are the focus of this dissertation, a utility function is compatible with the dominance relation [220] iff given two Pareto-optimal vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a}$ dominating $\mathbf{b}$ implies $u(\mathbf{a}) > u(\mathbf{b})$.

The set of all utility functions that are compatible with the dominance relation is denoted as U_c. Let $u^* = \max_{z \in A}\{u(z)\}$ be defined as the maximum value of a utility function on a POF, A. u^* represents a decision-maker's best compromise among other solutions in the set A.

Let A and B be two distinct POFs representing approximations of a true POF. Then,

$$U(A > B) = \{u \in U : u^*(A) > u^*(B)\} \subseteq U_c$$

is defined as the set of utility functions for which A is better than B. Consequently, the following relations and associated concepts are defined:

Definition 2.13. Outperformance Relation: Let A and B be two approximations of a true POF. Then, A outperforms B, denoted as $A\,O_U\,B$, if $U(A > B)$ is non-empty and $U(B > A)$ is empty.

Decision-makers' preferences, which are typically modelled with utility functions, are generally assumed, though in a weak sense, to be compatible with the dominance relation. Therefore, decision-makers are assumed to prefer non-dominated solutions [51], i.e. given two approximated sets A and B, a decision-maker will prefer $ND(A \cup B)$. This dominance-based assumption about a decision-maker's preferences was used by Hansel and Jaszkiewicz [81] to formulate the following three relations about POFs that approximate the true POF.

Definition 2.14. Weak Outperformance Relation: Let A and B be two approximations of a true POF. A weakly outperforms B, denoted as $A\,O_W\,B$, if $A \neq B$, $ND(A \cup B) = A$. A weakly outperforms B if a non-dominated element in B is also in A, and there is an element in A that is not in B.

Definition 2.15. Strong Outperformance Relation: Let A and B be two approximations of a true POF. A strongly outperforms B, denoted as $A\,O_S\,B$, if $A \neq B$, $ND(A \cup B) = A$, $B \setminus ND(A \cup B) \neq \phi$. A strongly outperforms B if every non-dominated element in B is also found in A and there is at least one element in B that is not found in A. In addition, every element in A is not dominated by elements in B.

Definition 2.16. Complete Outperformance Relation: Let A and B be two approximations of a true POF. A completely outperforms B, denoted as $A\,O_C\,B$, if $A \neq B$, $ND(A \cup B) = A$, $B \cap ND(A \cup B) \neq \phi$. In complete outperformance, every element in B is dominated by at least one element in A.

According to Knowles [114], reliable performance measures that will be consistent with the Pareto dominance relations must be compatible with the preceding outperformance relations.

The following definitions present the notions of compatibilities proposed in [81]:

Definition 2.17. Weak Compatibility: A performance measure is weakly compatible with an outperformance relation if for a pair of POFs, A and B, whenever A outperforms B the performance measure will evaluate A as not being worse than B.

Definition 2.18. Compatibility: A performance measure is compatible with an outperformance relation if for a pair of POFs, A and B, whenever A outperforms B the performance measure will evaluate A as being better than B.

In order to evaluate the efficiency of performance measures, Knowles [114] proposed the concepts of monotony and relativity.

Definition 2.19. Monotony: Given an approximated POF A, and let B be a new approximated POF, a monotonic performance measure will evaluate $A \cup B$ as not being worse than A.

Definition 2.20. Relativity: Given an approximated POF A, and let B be a true POF, then a performance measure will evaluate B as being better than A.

This dissertation focuses on DMOOPs. Performance measures used for evaluating DMOAs are now reviewed. These measures are broadly divided into measures that evaluate the accuracy of the found POFs, measures that evaluate the diversity of solutions in the found POFs and measures that combine both accuracy and diversity. For a comprehensive review of measures in the three categories, refer to [87] [90].

The following measures are relevant for this dissertation:

Definition 2.21. Hypervolume: The hypervolume, which is also referred to as the size covered, refers to the region of the objective space that is dominated by a non-dominated set [215] [218]. When the hypervolume of non-dominated sets are computed, the higher the value of the measure, the better the non-dominated sets, provided all non-dominated sets used the same reference vector when computing the hypervolume. For a formal definition of a domination region and the hypervolume, refer to [90].

Algorithm 1 Calculation of Wins and Losses

```
while existUnprocessed(DMOOP) do
  while existUnprocessed(n_t − τ_t combination) do
    Perform Kruskal-Wallis tests on pm
    if statistical significant difference then
      while existUnprocessed(DMOA-pair) do
        Perform Mann-Whitney U tests on pm
        if statistical significant difference then
          Assign wins and losses
        end if
      end while
    end if
    calculate Diff for the n_t − τ_t combination
  end while
  calculate Diff for the DMOOP
end while
```

Definition 2.22. Hypervolume Ratio: The hypervolume has a drawback, which is bias towards the convex region of the search space. To solve the bias problem, van Velduhizen [192] proposed hypervolume ratio, hvr, which is defined as follows:

$$hvr = HV(POF^*)/HV(POF).$$

hvr normalises the hypervolume of the approximated POF, POF^*, and its value ranges between 0 and 1. The maximum value of hvr is obtained with the true POF. The higher the value of this measure, the better the performance of the associated algorithm.

2.5 Statistical Analysis

In order to analyse the performance of multi-objective optimization algorithms (MOAs) in a changing environment, statistical approaches are employed. Two of the notable approaches which have been used for static MOOPs are as follows:

Algorithm 2 Generic Dynamic Multi-Objective Optimization Algorithm

Set population size, N
Set max archive size, SizeArchive
Initialize the iteration counter, iteration $\leftarrow 0$
Initialize time, $t \leftarrow 0$
Initialize$(P_t, freq, severity, dMOOP, t)$
{Initialize population of solutions, P_t}
AssignNonDominatedToArchive$(P_t, dMOOP, t)$
$bMORE \leftarrow TRUE$
while $bMORE$ **do**
  if $iteration \leq maxiteration$ **then**
    $t \leftarrow 1/severity * floor(iteration/freq)$
    {severity and frequency of change are inputs to the algorithm}
    Optimizer$(P_t, dMOOP, t)$
    {main optimization routine}
    {Implementation of Optimizer varies from one natured-inspired algorithm to another}
    {Subsequent chapters will elaborate on various implementations of Optimizer}
    Pick sentry solutions
    {sentry solutions are used to determine if a change in the environment has occurred}
    if ENV changes$(P_t, dMOOP, t)$ **then**
      ProcessChange$(P_t, freq, severity, dMOOP, t)$
      {if there is change in the optimization environment, algo processes the change}
    end if
    $iteration \leftarrow iteration + 1$
  else
    $bMORE \leftarrow FALSE$
  end if
end while

1. A performance measure is computed for an algorithm for each time step. The average and standard deviation of the measure for all the time steps are then used to analysis the algorithm's performance [18] [22] [23] [72] [115] [120] [206].

2. The average performance measure of each algorithm is used to compute wins and

losses. For a given problem, the performance measures of two algorithms are computed in each time step and compared using a Kruskal Wallis test. If there is a statistical significant difference between the algorithms' performance measure values, then the algorithm with a better average measure value is awarded a win, while the other algorithm is awarded a loss [82] [85] [86].

The shortcoming of the two approaches described above is that they do not account for the changing POFs. Therefore, those approaches are unable to evaluate how well an algorithm is able to track the changing POFs.

However, one popular approach that has been widely used to analyse the tracking behaviours of algorithms is to plot the performance measure value over time [5] [7] [38] [52] [83] [99] [121]. The problem with this approach is that the graph becomes too cluttered when the number of algorithms increases.

Helbig [87] proposed an approach that accounts for the ability of algorithms to track the changes in the dynamic environment of the optimization problems that are being solved. Algorithm 1 presents the procedure used in calculating wins and losses as proposed in [87]. When testing for a statistical significant difference, the average performance measures in each time step just before a change in the environment was used. In the algorithm, *pm* refers to performance measure, while Diff is the difference between the number of wins and losses. In assigning wins and losses as per Algorithm 1, Helbig and Engelbrecht [85] [86] [89] first proposed the following approach: if the Mann-Whitney test shows a statistical significant difference between the performance measure values of the pair of DMOAs, the average value of the measure is used to assign wins and losses. The algorithm with the best measured average over all time steps is the winner. This approach, named $wins - losses_A$ [87] is unable to measure how an algorithm is capable of tracking the changing POFs. Therefore, another approach, called $wins - losses_B$ was proposed in [87]. In $wins - losses_B$, average performance values of a pair of algorithms are compared, provided there is a statistical significant difference in performance measure values. The algorithm with the better average in each time step just before a change in environment occurred is assigned a win. By this method, the tracking ability of an algorithm is accounted for in the process of computing wins and losses as per Algorithm 1. Helbig and Engelbrecht[87] also proposed a normalization procedure for the wins and

losses computed - refer to [87] for more details about the normalised wins and losses.

2.6 Generic Dynamic Multi-objective Optimization Algorithm

This section presents a high-level description of a generic DMOA. This algorithm provides the framework for the experimental studies conducted in this research. The algorithm uses an archive to store the non-dominated solutions, or elites, that are found across different generations. The implementations of DNSGA-II, DVEPSO and 2DEVENS algorithms, which are considered in the following chapters, all built on this generic algorithm. The generic algorithm is presented in Algorithm 2.

2.7 Summary

This chapter discussed key concepts of mathematical optimization. Multi-objective optimization in the context of static and dynamic environments were also discussed. Important terms and concepts related to dynamic multi-objective optimization were defined and discussed. Statistical measures of performance of dynamic multi-objective optimization algorithms were motivated and a generic dynamic multi-objective optimization algorithm was presented. The next chapter will discuss the nature-inspired algorithms that were used in this research.

Chapter 3

Nature-Inspired Algorithms

Nature doesn't feel compelled to stick to a mathematically precise algorithm; in fact, nature probably can't stick to an algorithm.

Margaret Wertheim.

A lot can be learned from nature; nature efficiently solves certain problems. Therefore, algorithms can be designed to imitate nature's approaches to solving these types of problems. Algorithms [207] have been designed to mimic natures's problem-solving methodologies in areas such as the design of aircraft wings [63] [189], wind turbines [4] [166], bullet trains, bionic cars, etc. [174] Optimization problems, which require the search for the most effective ways of doing things, have been solved by traditional methods, such as those prevalent in operations research. Traditional approaches to optimization problems are either purely analytical/mathematical or iterative [135]. According to Goldberg [74], traditional approaches fall into the following three categories: calculus-based, enumerative and random. Calculus-based involves the use of derivative information, and it could be analytical or iterative. One major concern with the calculus-based approach is discontinuity and non-differentiability at some points in the search space. In the face of non-differentiability, an enumerative approach is preferred. However, enumerating all the points of the search space can prove a huge computational cost, in which case the random approach is preferred. In the long run, however, the random approach will behave like the enumerative approach. In light of the foregoing, it

is evident that the traditional approaches are well-suited for continuous, differentiable, unimodal and noiseless problems [56].

Evolutionary algorithms and swarm intelligence algorithms are two prevalent classes of nature-inspired algorithms. Computational intelligence is a bigger umbrella term for the subfields of evolutionary computation and swarm intelligence. Evolutionary algorithms, or evolutionary computation techniques, are search and optimization procedures that function by using a population of candidate solutions. Therefore, they are well-suited for finding the global optimal of optimization problems. They are inspired by biological evolution [123] as explained by Darwin's theory [40] of reproduction in order to produce offspring, mutation and the selection of the best offspring. Selection ensures that on average better offspring are selected in order to improve the desirable qualities in a population of biological entities. The selection process preserves the offspring who are able to best adapt to the changes in the competitive environment [12, 65]. For more information about evolutionary algorithms, refer to Section 3.1.

Flocking in birds, schooling in fish and a host of other herding behaviours found in gregarious animals have inspired computational procedures generally referred to as swarm intelligence algorithms. Swarming is a problem-solving technique that involves the distribution of local intelligence among a group of cooperative agents in order to solve a global problem. Local intelligence are shared in a distributed fashion and the coordination among the agents is decentralized and adaptive. The collective intelligence that exists in the swarm is never embodied in any of the agents in the swarm [92] [151].

The rest of the chapter is organized as follows: Section 3.1 presents the generic evolutionary algorithm. One of the evolutionary algorithms studied in this dissertation, the genetic algorithm, is discussed in Section 3.2. Particle swarm optimization algorithms are examined in Section 3.3. Another instance of evolutionary algorithms, differential evolution, that is studied in this dissertation, is discussed in Section 3.4, and Section 3.5 provides a summary of the chapter.

Algorithm 3 Generic Evolutionary Algorithm

1: *Let $t = 0$ be the generation counter*
2: *Create and initialize population, C(0)*
3: **while** *stopping conditions not true* **do**
4: *Evaluate the fitness, $f(x_i(t))$, of each individual, $x_i(t)$*
5: *Perform reproduction to create offspring*
6: *Select the new population, C(t+1)*
7: *Increase the generation counter: $t = t + 1$*
8: **end while**

3.1 Generic Evolutionary Algorithm

This section presents the generic evolutionary algorithm as presented in Algorithm 3. Search and optimization procedures that employ a generic evolutionary algorithm generally perform basic functions, such as:

1. Encoding of solutions as chromosomes.
2. Fitness evaluation in order to compute the survivability of an individual in the problem environment.
3. Initialization of the initial population.
4. Selection of individuals.
5. Reproduction of better and improved offspring from parents.

This algorithm may be implemented in different ways. The various implementations of the algorithm have given rise to some of the following evolutionary computation paradigms:

1. GA: this is based on genetic evolution [74, 145, 197, 207].
2. Genetic Programming (GP): it is based on GA, however individuals are programs or executable chromosomes [203].

3. Evolutionary Programming (EP): this is based on the evolution of behavioral traits, as against genotypic traits [67], which are based on genetic makeups.

4. Evolutionary Strategies (ES): evolution of phenotypic traits, which are expressed traits that are visible to the outside world, and are used in driving the genetic evolution; it is based on evolution of evolution [153].

5. Differential Evolution (DE): analogous to GA, but different in the way the reproduction is performed [148] [180].

6. Cultural Evolution (CE): changes in the cultural character of a population is used to drive the evolution of genetic and phenotypic traits of the population [157] [158] [159].

7. Co-evolution (CoE): individuals in a population may develop reciprocal relationships that facilitate co-evolution. For instance, an insect may develop specialized capacities to feed on a plant, and the plant may in turn develop abilities to ensure that its pollination is facilitated by the insect. These reciprocal relationships may be cooperative or competitive [93].

Only evolutionary computation paradigms that are used in this research are discussed in the rest of the chapter in more detail.

3.2 Genetic Algorithm

This section discusses the GA. A background to GAs is presented in Section 3.2.1. Chromosome representation schemes are discussed in Section 3.2.2. The dominant operators used in GAs are discussed in Sections 3.2.3, 3.2.4 and 3.2.5. Section 3.2.6 presents a representative DMOA that employs GA techniques to solve DMOOPs.

3.2.1 Background

GAs are stochastic and computational models of biological and genetic evolution. They were first proposed by Fraser [69, 70], Bremermann [19] and Reed et al [154]. However,

it was the work of Holland [93] that popularized GAs. It is one of the earliest evolutionary computation techniques to gain widespread adoption. Like other evolutionary algorithms, it is widely used for search and optimization. Because it is a population-based algorithm, searching for optimal solutions typically starts with a population of candidate solutions, which are randomly generated to ensure that the solutions are widely drawn from the search space. Such randomly drawn candidate solutions help to ensure the diversity of the final (best) solutions. Also, the widely drawn candidate solutions ensure that the search can advance simultaneously from different parts of the search space, which can help to avoid the possibility of the algorithm getting stuck in a local optimum. Each solution in the population is the equivalent of a chromosome in a biological evolution. The functional value of a solution is the equivalent of the phenotype in a biological evolution. The phenotypic value of a solution is also referred to as the fitness value of the solution.

GAs, including the original version that was proposed by Holland [93], are generally consistent in their procedures with the steps in Algorithm 3. The reproduction step in Algorithm 3 roughly corresponds to crossover and mutation steps in a GA. In addition, different implementations of a GA may treat the key elements/steps of Algorithm 3 in different ways. The following subsections will therefore examine some of the implementation specific differences in how solutions in the search space may be represented and how the various forms of operators used in a GA may be implemented.

3.2.2 Chromosome Representation

A chromosome is the fundamental building block in evolutionary algorithms (EAs). It is the carrier of genetic characteristics of individuals and it is a string of genetic materials. The genes, which constitute the components of a chromosome, carry important genetic information that determines the survival strengths of individuals in a competitive environment.

For optimization and search problems, individual solution in the search space requires a representation of the chromosomes that will facilitate navigating the search space. Such representations should also help to preserve quality genetic material from generation to generation. Encoding solutions is the act of finding such representations for solutions in

the search space. According to Section 3.1, finding appropriate encodings is an important first step in the workings of EAs. Different search and optimization problems may impose constraints on the choice of encodings [117]. Various encodings have been used in GAs, namely:

1. Binary encoding. where a chromosome is represented as a binary string [75]. Each bit in the string carries genetic information about an individual. The original implementation of a GA proposed by Holland [93] used this encoding.

2. Permutation encoding, where a chromosome is a sequence of numbers which are each representing positions, such as the position/location of a city in a Travelling Salesman Problem [15]. This type of encoding is used when the fitness of a chromosome depends on the order of genetic material in the chromosome [118] [168] [184] [198] [199].

3. Integer number encoding, where the constituents of a chromosome are integers [168].

4. Floating-point number encoding, where each component of a chromosome is a floating-point number [50] [60] [107] [204].

3.2.3 Selection Operator

In Algorithm 3, selection is one of the key operations. The motivation behind selection is to improve the chance of better individuals to survive to the next generation [175]. At the same time, a good selection strategy should facilitate diversity in the population by avoiding too high selection pressure, which tends to favor a few better individuals at the expense of the weaker ones. In a population, a subset will be selected to undergo reproduction in order to generate a new population of individuals for the next generation. Also, when parent individuals have been selected and mated, some of the new offspring may be selected for the next generation according to the selection method adopted by the GA. In this section some of the popular selection strategies are discussed. However, for a more comprehensive discussion of selection strategies, refer to [58] [173] [175].

The following are popular selection strategies:

1. Random: In a population of N individuals, a purely randomized process is used in making the selection. Each individual in the population has equal probability of being selected, since fitness information are not taken into account. Because the process of selection is purely random, no member of the population is favored. Therefore, selective pressure is the least in this selection strategy [58].

2. Fitness Proportional: The probability of selection is proportional to the individual's fitness. There are different implementations of this strategy that vary in how they compute the probability of selection [175]. Roulette wheel is one popular implementation [95] of the fitness proportional strategy, and it proceeds by first computing the sum of the fitness for all individuals in the population and then divides the fitness of each individual by the sum. The values obtained are the selection probability of each individual. Then an iterative procedure is kick-started by setting a sum of selection probabilities, S, to zero. The procedure also generates a random number, r, between 0 and 1. The procedure then runs through the population, while r is greater than a sum of selection probabilities, S, for each individual, the procedure adds the selection probability, S, to the selection probability of the current individual of the population. The iteration ends with the first member found, where S is greater than r, and that member is returned as the selected member.

3. Tournament: In this strategy, n individuals are randomly selected from the population. The best among the randomly chosen n individuals wins the tournament selection [173].

4. Rank-based: Baker [13] proposed rank-based selection in order to eliminate a disadvantage with proportionate selection [173]. In this strategy of selection, the population is first sorted in decreasing order of fitness. The first individual in the sorted population is assigned a rank of N, corresponding to the population size. The rank of the second individual is $N-1$, the third is $N-2$, etc. The last individual, which is the one with the lowest fitness, is assigned a rank of 1. The selection probability is then computed according to a mathematical relation proposed in [173] for linear and exponential rankings. The selection probability is

proportional to the ranks, unlike the fitness in the case of proportional selection.

5. Elitism: In this selection strategy, the best individuals in any generation are preserved, without mutation, and survive to the next generation. With this strategy the best genetic materials are preserved from one generation to the next, ensuring that operators, such as crossover and mutation, do not destroy the best materials. Elitism may favor exploitation at the expense of exploration, thereby inhibiting diversity of genetic materials in the population.

6. Hall of Fame: The best members of each generation are added to an archive, and the members in the archive are used as parents in the subsequent generation during crossover. This strategy, first proposed by Rosin et al [160] [161], ensures that offspring are produced from the best individuals in the population.

3.2.4 Crossover Operator

Offspring are produced from parents through the process of reproduction, i.e. crossover and/or mutation [58]. Unlike selection operators, reproduction operators are highly affected by the choice of chromosome representation. In crossover, offspring may be produced from a single parent. This is called asexual crossover. In sexual crossover, two parents are used to produce offspring, while multi-recombination crossover employs more than two parents. When the selected chromosomes are binary, binary crossover is used, while real-coded crossovers are used where real-encoding of chromosomes is adopted.

In binary crossover, two parents are typically used. One-point binary sexual crossover randomly selects a crossover point, and bitstrings after that point are swapped between the two parents [95]. Two-point crossover selects two points in the parents. Every bitstring between the two points are swapped in the two parents. However, the idea of two-point crossover may be extended to more points [43] [59] [154]. Uniform crossover [1] [183], however, employs a fixed mixing ratio between the two parents. It starts by setting a probability measure. Assuming a measure of 0.5, bitstrings in the offspring are likely to be composed of half of each parent's bitstrings. Each bit position in the chromosomes of the parents are swapped based on the probability measure.

Floating-point representation of chromosomes requires specific crossover operators that exploit the peculiarities in representation scheme. However, the preceding binary crossover operators have been used for floating-point representation despite their limitations in capturing the nuances of the floating-point search space. While binary crossover operators specifically swap bitstrings in the parent chromosomes in order to generate offspring, floating-point, otherwise called real-coded, operators blend and transform components of genetic material in parents in order to produce offspring. Wright [204] developed one of the earlier floating-point operators called linear crossover. Three solutions are generated from two parents and the best two are selected as offspring [56]. The arithmetic crossover operator was proposed by Michalewicz [131]. The operator takes a weighted average of two or more parents in a kind of recombination strategy. One offspring is generated in the process. Elsehelman and Schaffer [60] created an adaptation of the weighted arithmetic operator, called blend crossover (BLX-α). To simulate the functionality of single-point binary crossover, Deb and Agrawal [46] developed the simulated binary crossover (SBX). SBX has the quality of ensuring that generated offspring are not biased towards any of the parents. Further details about the mathematical formulations of these floating-point operators can be found in [56].

The foregoing floating-point operators are mostly used in sexual, or two parent, crossover operations. Sometimes, there is a compelling need for intensive exploratory operations, a need that is served by multi-parent crossover. In multi-parent crossover two or more parents are involved in the process of generating offspring. Bringing together two or more parents increase the dissemblance between parents and offspring, therefore resulting in a more diversified population. Ono and Kobayashi [136] developed a unimodal normal distribution crossover (UNDX). In UNDX three parents are utilized to generate two or more offspring. An ellipsoidal probability distribution, where one of the axes is formed along the line that connects two of the parents, is used to generate the offspring. UNDX has however been generalized to work with two or more parents using the center of mass of the parents [56]. Another center of mass crossover operator, called simplex crossover (SPX), was proposed by Tsutsio and Goldberg [190], as well as Renders and Bersini [155]. SPX uses a uniform probability distribution in a restricted search space around the parents. Lastly, Deb et al [50] developed a generic parent-centric

crossover operator, called Parent-Centric Recombination (PCX) that does not employ center of mass in computing offspring from the parents. The procedural rules used by PCX are presented in [50].

3.2.5 Mutation Operator

The goal of mutation is to introduce new genetic material into an existing individual of a population. Mutation helps to facilitate diversity in a population. Together with crossover, an evolutionary algorithm can increase its exploratory capability in the search space. Like crossover, the chosen approach for mutation depends on the representation scheme used for the chromosomes. Generally, binary and floating-point mutations are the dominant types.

Binary mutation works on binary strings. Three main types of binary mutations are generally identified:

1. Uniform mutation [94]: Bit positions are randomly selected, and the selected positions are negated.

2. Inorder mutation: Two bit positions are randomly selected and the bitstrings between the two positions undergo uniform mutation.

3. Gaussian mutation: Hinterding [91] proposed that the bitstrings to be mutated are first converted to floating-point values. Gaussian noise is then added, thus mutating the underlying chromosome. The mutated floating-point is then converted to equivalent bitstrings.

Hinterding [91] and Michalewicz [131] indicated that better performance is obtained when floating-point mutation is employed in situations where decision variables take floating-point values. Floating-point mutations directly work on real-encoded chromosomes, as against conversion from binary to real. The real values that represent genetic material are generally mutated by adding a noise value, which is sampled from a statistical distribution. Floating-point mutations have been classified based on the choice of statistical distribution that is used to sample the noise value. Examples are Gaussian mutation [64] [66], uniform mutation [128], Cauchy mutation [208] [209] [210], etc.

3.2.6 Dynamic Non-Dominated Sort Genetic Algorithm II

DNSGA-II is one of three nature-inspired algorithms that are used in the experimental studies conducted in this research. Therefore, the purpose of this section is to describe some of the salient features of the algorithm.

DNSGA-II is an implementation of the GA that employs a non-dominated sorting procedure [48] in solving optimization problems in dynamic environments. Dynamic optimization problems are characterized in such a way that the objective function(s), constraint functions, and/or other parameters of the problems are changing with time [87] [134]. Because this dissertation focuses on MOOPs, optimization problems are hereby assumed to possess two or more objectives.

In the real-world, dynamic optimization problems usually present themselves as optimal control problems or online problems [147]. The algorithms used in optimal control problems usually involve evolution of control laws or rules, which are used to solve optimization problems offline. For online optimization, the problem is considered stationary for a time period, and a solution is computed during the stationary moment of the problem. Then the problem is advanced to the next time step in order to compute a solution for a new instance of the problem. The process of incremental change in time is continued until the stopping criterion for the algorithm has been reached [47]. Every time a new instance of the optimization problem is produced, an algorithm seeks to find a solution that approximates the true solution to the problem. Because of the incremental nature of the employed timing procedures, and the associated assumption of stationarity of the problem while the time is fixed, the best an algorithm can obtain is an approximated solution to the real problem. For MOOPs, a set of solutions are usually found. The solutions are not inferior to any other solutions in the set when all the objectives are considered. Therefore, they are also referred to as non-dominated solutions, or Pareto-optimal solutions [177]. Because of the conflicting nature of the objectives used in MOOPs, Pareto-optimal solutions represent the best tradeoff between objectives.

A non-dominated sorting genetic algorithm was first proposed by Srinivas and Deb [177], and the proposed algorithm was called the non-dominated sort genetic algorithm (NSGA). However, NSGA is characterized by the following pitfalls:

1. High computational complexity: time complexity of the algorithm is $\mathcal{O}(mN^3)$,

which is not good for algorithms with a large population size.

2. Lack of elitism: use of elitism can assist in speeding up the performance of a GA [217], and to preserve quality solutions that have been found. NSGA, therefore lacks the benefits of elitism.

3. Need for specifying a sharing parameter: to increase diversity, NSGA depends on the use of a specified sharing parameter.

DNSGA-II can maintain diversity without using any special parameter. The implementation of DNSGA-II employed in this dissertation uses elitism and the crowding distance operation, a parameterless approach to maintaining diversity [68], while still making use of the version of non-dominated sort that was used in NSGA [177]. Because convergence and diversity, which are two important measures of performance used in DMOO, are believed to be unaffected by run time complexity, $\mathcal{O}(mN^3)$ complexity is adequate for the purpose of this research. Future work on this research will however be carried out with the fast non-dominated-sort that is specifically associated with the non-dominated sort genetic algorithm version II (NSGA-II).

3.3 Particle Swarm Optimization

This sections discusses PSO. Section 3.3.1 presents a background to PSO. The basic PSO algorithm is discussed in Section 3.3.2. A general PSO algorithm is presented in Section 3.3.3. Network topologies are discussed in Section 3.3.4. PSO control parameters are presented in Section 3.3.5. Section 3.3.6 presents a multi-population PSO algorithm, namely DVEPSO.

3.3.1 Background

PSO is a population-based optimization and search algorithm that is based on the social behavior of birds in a flock [112]. It was originally developed by James Kennedy and Russel Eberhart. In PSO individuals in a swarm are called particles. For optimization problems, each particle represents a potential solution to the optimization problem being

solved. Particles in a swarm exhibit cooperative behaviours, such as sharing position data with the neighboring particles. Such a sharing mechanism helps to ensure that the best known data about the target position, in a search or optimization sense, is propagated to all the particles in the neighborhood. Put differently, particles in a swarm emulate the success of their neighbors and their own successes. The implication of this self and neighbor emulation is that the globally superior successful behaviors, such as the optimal solutions in an optimization problem, is ultimately communicated to all particles in the neighborhood.

3.3.2 Basic Particle Swarm Optimization Algorithm

A basic PSO algorithm comprises the following key steps:

1. Initialization stage: random generation of particles, i.e. for optimization problems, the individuals in a search space.

2. The algorithm is run for a number of iterations, or until a level of approximation to the target solution is obtained.

3. At each time step, t, a new position is computed for each particle in the swarm. The step size and the direction of movement of a particle is determined by the particle's velocity, $v_i(t)$. Equation 3.1 presents how a particle's new position, $x_i(t)$, is calculated from its previous position, $x_i(t-1)$, and its new velocity.

$$x_i(t) = x_i(t-1) + v_i(t) \tag{3.1}$$

4. The velocity is estimated from the knowledge of the best position of the particle at a time, and the best position of all the neighboring particles at the time.

$$v_i(t) = \omega v_i(t-1) + c_1 r_1 (x_i^{pbest}(t-1) - x_i(t-1)) + c_2 r_2 (x_i^{nbest}(t-1) - x_i(t-1)) \tag{3.2}$$

where ω is the inertia weight that controls the impact of the previous velocity on the new one [171]. r_1 and r_2 are randomly drawn from U(0,1). c_1 and c_2 are positive

acceleration constants that are used to scale the contributions of the cognitive (second term of Equation 3.2) and social (third term of Equation 3.2) components respectively. $x^{pbest}(t)$ is the personal best of a particle at time t. $x^{nbest}(t)$ is the neighborhood best of a particle at time t. If gbest PSO is employed, $x^{nbest}(t)$ defaults to the global best position from all particles in the swarm, otherwise the local best is used which is computed from the local neighborhood of each particle.

5. The velocity vector, $v_i(t)$, drives the algorithm towards the optimal solutions. It reflects the experiential knowledge of a particle and the socially acquired knowledge of the particle. The experiential knowledge is generally referred to as the cognitive component and the socially acquired knowledge is referred to as the social component. Particles' velocities are updated according to Equation 3.2.

6. In order to prevent particles from moving too fast within the search space, a problem that may degrade the exploitative capability of a particle, velocity clamping rules were proposed [172]. Such clamping rules limit particles' velocities.

3.3.3 A General Particle Swarm Optimization Algorithm

Algorithm 4 presents the general PSO algorithm.

In Algorithm 4, the initialization of the swarm is the first step. During initialization the following must be set for each particle:

1. Position - Each particle's position, x_i, is set as follows: $x_i = x_i^{min} + r(x_i^{max} - x_i^{min})$, where r is a random number in U(0,1), x_i^{max} is the maximum value of x_i in each dimension and x_i^{min} is the minimum value of x_i in each dimension.

2. Velocity - A very simple rule is to set the initial velocity of each particle to zero. This is in agreement with the natural idea of a particle starting at rest.

3. Personal best - The personal best of each particle is initialized to the initial position of the particle.

Algorithm 4 does not strictly enforce how particles are initialized. Implementations of the algorithm can adopt other proposals in the literature [20] [21] [142].

In the algorithm, the update of the swarm's best can take two major forms. The swarm's best can be computed from the neighborhood of each particle. This update mechanism gives rise to a Local Best PSO. The update of the swarm's best can also be obtained from the whole swarm, beyond the neighborhood of the particle. This update mechanism gives rise to a Global Best PSO.

The stopping condition of the algorithm should prevent premature convergence and over sampling of the search space. Over sampling of the search space will result in a disproportionate increase in computational complexity. The normally used stopping conditions include the following:

1. Terminate when an upper limit for number of iterations is exceeded.
2. Terminate when the quality of the found solution is satisfactory.
3. Terminate when no better solution is found after a specified number of iterations.
4. Terminate when the value of the normalized swarm radius is close to zero.
5. Terminate when the slope of the objective function is close to zero.

3.3.4 Network Topologies

As stated in the previous section, each particle uses its socially acquired knowledge to improve on its position. The socially acquired knowledge is propagated in different ways, depending on the type of neighborhood relations, or network topologies, that exist between the particles. Distance measures used by the neighborhood relations are topological, rather than spatial, i.e. distance between two particles in the search space is not measured by the actual distance. Different topologies have been studied [111]. Four of such popular topologies are hereby presented, and Figure 3.1 graphically presents the four topologies discussed below.

1. Star Topology: First introduced by Kennedy and Eberhart [112]. In a star topology every particle is directly connected to all other particles in the population. Therefore, the neighborhood best of each particle corresponds to the global best of the swarm.

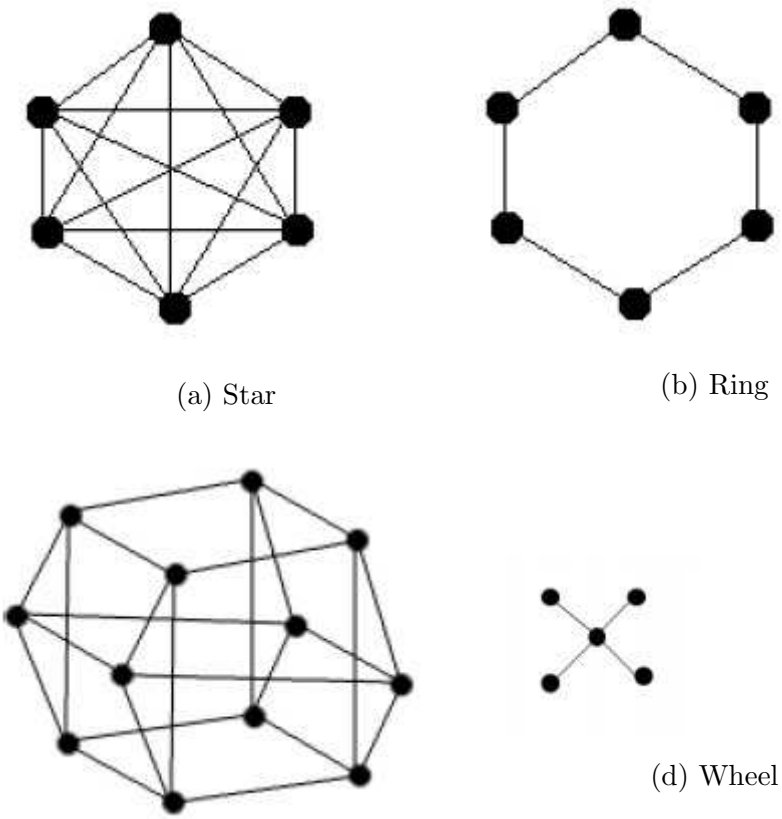
Algorithm 4 General PSO Algorithm

Set generation counter, $t = 0$
Initialize swarm of particles, C(t), where n is the size of the swarm
repeat
  for each particle $i = 1...n$ **do**
    if $f(x(i,t)) < f(x^{pbest}(i,t))$ **then**
      Set $x^{pbest}(i,t) = x(i,t)$ {set the personal best of particle}
    end if
    if $f(x^{pbest}(i,t) < f(x^{nbest}(i,t))$ **then**
      Set $x^{nbest}(i,t) = x^{pbest}(i,t)$ {set the swarm best}
    end if
    for each particle $i = 1...n$ **do**
      update the velocity using Equation 3.2
      update the position using Equation 3.1
    end for
    Set $t = t + 1$
  end for
until stopping condition is true

2. Ring Topology: Each particle in a ring topology communicates with its m immediate neighbors in the population. The neighborhood best of a particle is computed using the immediate neighborhoods that comprise of the particle itself and the other m immediate particles.

3. Von Neumann: A particle in this topology is connected on its left, right, top and bottom to other neighboring particles in a two dimensional lattice. It has been demonstrated to be superior to other topologies for a number of problems [113].

4. Wheel: A central particle connects all other particles and socially acquired knowledge is propagated through the central particle. Like the star topology, every particle receives the global best information. However, the global best information is passed through the central particle, slowing down propagation of information among the particles.



(a) Star

(b) Ring

(c) Von Neumann

(d) Wheel

Figure 3.1: Network Topologies

3.3.5 Control Parameters

Some of the control parameters that can influence Algorithm 4 include: the dimension of the problem, acceleration coefficients, inertia weight, clamping velocity, etc. Three of the control parameters are discussed in this section. However, for a more comprehensive discussion of the control parameters the reader is referred to [56].

1. Inertia weight: This is represented by the ω term in Equation 3.2. It was introduced to assist with controlling the exploration and exploitation abilities of the swarm [170]. The inertia weight avoids the need to clamp velocities [53]. The inertia weight moderates the momentum of particles in the search space and determines how much the past velocity influences the new velocity of a particle. A large value

of ω promotes exploration with increased diversity, while a small value promotes local exploitation. The smaller the inertia weight, the more the cognitive and social components contribute to the position update of particles. An optimal value for this parameter is problem-dependent [54]. However, an optimal choice for ω has been related with another important control parameter, namely the acceleration coefficients. Van den Bergh and Engelbrecht [191] proposed the relation expressed in Equation 3.3.

$$\omega > \frac{1}{2}(c_1 + c_2) \text{ and } \omega \leq 1 \tag{3.3}$$

where c_1 and c_2 are the acceleration coefficients associated with the cognitive and social components in the velocity update equation respectively.

The above relation guarantees convergence. Divergent or cyclic behavior may result if the relation does not hold. Tresla [187] also derived a similar relation.

2. Acceleration Coefficients: The stochastic behavior of the cognitive and the social components in the velocity update equation is controlled by the acceleration coefficients, (c_1 and c_2), and the associated random vectors, (r_1 and r_2). A particle's trust in its own knowledge is embodied by c_1, while c_2 reflects how much the particle trusts the knowledge of the surrounding particles. Other variations of acceleration coefficient combinations can be used: $c_1 = 0$ for a situation where a particle only relies on the socially acquired knowledge; $c_2 = 0$ where a particle only relies on its own knowledge and every particle behaves like an independent hill-climber. An independent hill-climber may wander unnecessarily for too long before finding the global optimal, resulting from the fact that it is not exploiting the socially available information about the optimal. When $c_1 = c_2$, a particle places equal amount of trust on the cognitive and socially acquired knowledge. An interesting combination, such as $c_1 = c_2 = 0$ results in particles that wander the search space under the sole influence of their velocity. Such particles do not exploit any knowledge learned from themselves and the neighboring particles. It is a kind of brute-force search, however boundary constraint rules can ensure that such brute-force proceeding particles do not wander out of the allowable limits.

Kennedy [110] suggested through an empirical study that particles should employ $c_1 + c_2 \leq 4$, otherwise velocities and positions of particles may overshoot. Acceleration coefficients are usually static, and optimal values for them have been suggested by empirical studies [191]. However, there are proposals on the notion of adaptive acceleration coefficients [29, 152, 182, 196, 205]. Convergence behaviors of Algorithm 4 have been demonstrated to be correlated with the choice of values for the inertia weight and the acceleration coefficients [28][187], as also indicated in Equation 3.3.

3. Velocity Clamping: Algorithm 4 is expected to demonstrate a right mix of exploration and exploitation. The velocity updates should proceed in such a way that a reasonable number of candidate solutions in the search space are consulted. The consultation should be deep, i.e. showing good exploitation capability, and wide, i.e. showing good exploration capability. Particles in a swarm should not move too fast, in order to perform not too bad on exploitation. Similarly, they should consult as much of the search space as possible and without going out of the bounds. Velocity clamping sets a limit, like an upper bound, for allowable velocities [55]. Clamping the velocity ensures that the right mix of exploration and exploitation is demonstrated by the algorithm, thus ensuring that particles do not move too fast in a swarm. Equation 3.4 presents a velocity clamping rule, where $v'_{ij}(t+1)$ is computed using Equation 3.2.

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{max,j} \\ V_{max,j} & \text{if } v'_{ij}(t+1) \geq V_{max,j} \end{cases} \tag{3.4}$$

where $V_{max,j}$ is the maximum velocity for the particle.

3.3.6 Dynamic Vector Evaluated Particle Swarm Optimization

DVEPSO [84] [86] uses PSO to solve DMOOPs by employing multiple swarms, or multiple populations, where each swarm optimizes one objective. Dynamic problems are the focus of this dissertation. Some of these problems are characterized by two or more objectives. PSO is a popular nature-inspired algorithm that has been used to solve some of

these problems, particularly the static ones. However, when the problems' characteristics change with time, algorithms that can track the changing characteristics are required, and DVEPSO serves this purpose. In this dissertation, and in some of the empirical studies conducted in the dissertation, an implementation of DVEPSO is variously compared against two other nature-inspired algorithms. The following chapters, where the empirical studies are discussed, will present additional detail about DVEPSO, especially its runtime performance relative to the two other algorithms for the set of problems to which the algorithms were applied.

3.4 Differential Evolution

This section discusses DE. Section 3.4.1 provides a background to DE. DE operators are discussed in Section 3.4.2. Strategies used by DE algorithms are presented in Section 3.4.3. A basic DE algorithm is discussed in Section 3.4.4. Control parameters are presented in Section 3.4.5.

3.4.1 Background

DE was proposed by Storn and Price in 1995 [180]. It is a stochastic and population-based search strategy originally designed to solve continuous valued problems. The search is guided by distance and direction information from the current population. However, it resembles other EAs in many ways. Individuals in a population are called parameter vectors. The difference in the parameter vectors is used to explore the search space. DE is simple and straightforward to implement when compared to other EAs. Studies have shown that DE performs better than PSO on certain problems [41, 150, 194]. Like other EAs that were covered in previous sections, crossover and mutation are employed to ensure variation in the population from one generation to another. DE mutation always precedes crossover. The major operators used by DE are discussed in the following section.

3.4.2 Operators

1. Mutation: An individual in a DE population is mutated by using a trial vector. The trial vector is computed after the mutation of a target vector with a weighted differential. The trial vector is used during crossover to produce an offspring. For each parent, x_i, a target vector, x_{i_1}, is selected, such that $i \neq i_1$. Then, two individuals x_{i_2} and x_{i_3}, such that $i \neq i_1 \neq i_2 \neq i_3$, are uniformly selected and used to compute a difference vector. The trial vector, $u_i(t)$, is thus computed from the target and the difference vector as shown in the equation below:

$$u_i(t) = x_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \tag{3.5}$$

where $\beta \in (0, \infty)$ is the scale factor which controls the amplification of the difference vector.

2. Crossover: The crossover operator in DE uses a discrete recombination of the trial vector, $u_i(t)$, and the parent vector, $x_i(t)$, to produce offspring, $x_i^{'}(t)$. The operator is implemented as follows:

$$x_{ij}^{'}(t) = \begin{cases} u_{ij}(t) & j \in J \\ x_{ij}(t) & otherwise \end{cases} \tag{3.6}$$

where $x_{ij}(t)$ refers to the j-th element of the vector $x_i(t)$ and J is the set of crossover points. J can be determined using various approaches. Two of the most frequently used approaches are binary crossover and exponential crossover [150, 179]. In binary crossover, crossover points are randomly selected from S, which is defined as $S = 1, 2, ..., n$ where n is the dimension of the problem space. Algorithm 5 presents binary crossover. In Algorithm 5 p_r is the probability of crossover, and the larger its value the more points in trial vectors will be used in creating offspring. In exponential crossover, from a randomly selected index, a set of crossover points is constructed. The constructed set is a sequence that comprises indices starting from the randomly selected index. The population indices are considered to be organized in a circular form, in case the crossover procedure requires to move past the last index of the population. Algorithm 6 presents exponential crossover.

Algorithm 5 Differential Evolution: Binary Crossover

$j^* \sim U(1, n)$, *where n is the problem dimension*
$J \leftarrow J \cup \{j^*\}$
$\{p_r$ is crossover probability$\}$
for each j = 1...n **do**
  if $U(0,1) < p_r$ *and* $j \neq j^*$ **then**
    Set $J \leftarrow J \cup \{j\}$
  end if
end for

3. Selection: This operator is used for two purposes. First, to select which individuals in the population will be used to construct the trial vector. The target vector can be randomly selected from the population. It can also be selected by some other scheme, some of which are examined in following section. Different vectors are also selected by various schemes from the population. Secondly, selection is used to determine which of the parents and offspring survive to the next generation. One simple strategy used by DE to determine candidates for the next generation is to choose between a parent and its offspring based on their respective fitness values.

3.4.3 Strategies - DE/x/y/z

DE algorithms are generally characterized by a scheme that specifies how the trial vector is chosen, the number of difference vectors employed, and the crossover method adopted. The scheme generally employs the notation: DE/x/y/z [178] [180]. In the notation, x denotes the method used in selecting the trial vector, y denotes the number of vectors used, and z refers to the crossover method that is employed. Some of the popular strategies are as follows:

1. DE/rand/1/bin: The target vector is randomly selected, one difference vector is used and binary crossover is adopted [56].

2. DE/rand/1/exp: This is the same as DE/rand/1/bin, except for the crossover that is exponential [56].

Algorithm 6 Differential Evolution: Exponential Crossover

1: $j^* \sim U(0, n-1)$, *where* n *is the problem dimension*
2: $J \leftarrow \emptyset$
{p_r is crossover probability}
3: **repeat**
4: $J \leftarrow J \cup \{j+1\}$
5: $j = (j+1)\ mod\ n$
6: **until** $U(0,1) \geq p_r$ *or* $|J| = n$

3. DE/best/1/z: The target vector $x_{best}(t)$ is the individual with the best position in the population. One difference vector is used. The trial vector is calculated as follows:

$$u_i(t) = x_{best}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)) \tag{3.7}$$

Any of the crossover methods can be used.

4. DE/x/n_v/z: More than one difference vector is used. The trial vector is calculated as follows:

$$u_i(t) = x_{i_1}(t) + \beta \sum_{k=1}^{n_v} (x_{i_2,k}(t) - x_{i_3,k}(t)) \tag{3.8}$$

Any suitable method can be used to select the target vector, $x_{i_1}(t)$. n_v is the number of difference vectors, and $x_{i_2,k}(t) - x_{i_3,k}(t)$ is the k-th difference vector. Any suitable crossover method can be used.

5. DE/rand-to-best/n_v/z: This strategy combines random selection and best schemes in calculating the target vector. The trial vector is computed as follows:

$$u_i(t) = \gamma^* x_{best}(t) + (1 - \gamma^* x_{i_1}(t) + \beta \sum_{k=1}^{n_v} (x_{i_2,k}(t) - x_{i_3,k}(t)) \tag{3.9}$$

$\gamma^* \in [0, 1]$ is a control parameter that determines how much an algorithm balances the forces of exploitation and exploration. A value closer to one means that an algorithm will do more exploitation. One or more difference vectors can be used. Any suitable crossover method can be employed.

Algorithm 7 Differential Evolution: Basic Algorithm

Set the generation counter, $t = 0$
Initialize the control parameters: β, p_r
{β: scaling factor}
{p_r: crossover probability}
Create and initialize the population, $P(t)$, of n_s individuals
while stopping conditions not true **do**
  for each individual, $x_i \in P(t)$ **do**
    Evaluate fitness, $f(x_i(t))$
    Create the trial vector, $u_i(t)$, according to a mutation operator of choice
    Create an offspring, $x_i^{'}(t)$, according to a crossover operator of choice
    if offspring, $x_i^{'}(t)$, is fitter than the parent, $x_i(t)$ **then**
      Add $x_i^{'}(t)$ to $P(t+1)$
    else
      Add $x_i(t)$ to $P(t+1)$
    end if
  end for
  Increment the generation counter: $t = t + 1$
end while

3.4.4 Basic Differential Evolution Algorithm

The basic DE algorithm starts by creating and initializing a random set of individuals that represent candidate solutions to the problem being solved. Each individual, x_i, is generated in such a way that the boundary constraint requirements are not violated, i.e. $x_{ij} \in (x_{min,j}, x_{max,j})$, where $x_{min,j}$ and $x_{max,j}$ respectively represent the minimum and maximum values of x_i in dimension j. Algorithm 7 presents the workings of a basic DE algorithm.

3.4.5 Control Parameters

Three major parameters influence the behavior of a DE algorithm:

1. Population size, n_s: The population size directly affects how well an algorithm

explores the search space. The more the population, the more the points that are accessible by an algorithm, all other facts being constant. A higher population size implies more options in selecting difference vectors and target vectors. The increased options improves the search capability of an algorithm. Computational costs, however, grow with higher a population size.

2. Scaling factor, β: The scaling factor, $\beta \in (0, 1)$, helps to amplify the effects of the difference vector. A smaller value means smaller mutation step sizes. Smaller mutation step sizes slows down convergence. However, smaller step sizes facilitate better exploitation of the search space. For better exploration and faster convergence, the scaling factor should approach its upper limit of 1. It has been shown that large values of β result in premature convergence [103, 108]. A scaling factor of 0.5 has been shown to provide good performance [3] [122] [180].

3. Crossover Probability, p_r: This is also called the recombination probability and determines the proportion of the parent's components that can change. The higher this value, the more diversity subsequent generations exhibit. Faster convergence is associated with higher value of crossover probability [3] [122].

3.5 Summary

This chapter discussed nature-inspired algorithms that are used in this study. The three key algorithms discussed are: genetic algorithms, particle swarm optimization and differential evolution. For each of the algorithms, key terms, concepts and building blocks were discussed. The next chapter deals with decision-making, important concepts and processes of decision-making, and how human beings and their preferences can influence the processes in the context of optimization problems.

Chapter 4

Decision-Making Essentials

Making a decision is a cognitive activity that typically occurs in solving problems. It is an age-long phenomenon among humans. Achieving one's objectives in a world of scarce resources requires judicious use of choices available to the decision-maker. When Christopher Columbus, with his hypothesis about the distance between Canary Island and the Far East, set out to sail westward in order to find a new trade route between Europe and the Far East, he had to contend with the choice of sailing westward or staying at home. Even though he eventually embarked on the sail, he also had to live with the risks associated with the expedition. Some of the risks were: he may die while sailing through the Atlantic, the westward sail may not yield a fruitful path to the Far East, etc.

As an academic study, decision making, especially in the context of decision science, was widely and systematically studied in the 20^{th} century. Many theories have been developed to guide rational decision making, the kind of decisions that are examined in this chapter. When a rational decision-maker makes decisions, he does so bearing the desired objectives in mind. The desired objectives, also called preferences, embody the decision-maker's beliefs about the states of the world in which decisions are made.

This chapter presents the essential elements of rational decision making. The chapter starts by presenting the grand theories of decision making in Section 4.1. Section 4.2 presents a formalisation of decision problems and some of the essential elements of decision making, such as objectives, states of the world and acts/actions of a decision-maker.

Section 4.3 presents contexts in which decisions can be made. Preferences in decision making are discussed in Section 4.4. Decision making under multiple criteria, or two or more objectives, is presented in Section 4.5. A summary of the chapter is provided in Section 4.6.

4.1 Decision Theories

Contributions to decision theories have been inspired by different disciplines: philosophy, psychology, computer science, statistics, mathematics, economics, etc. This section takes a critical look at some of the key ideas underlying decision theories. Section 4.1.1 presents the descriptive and normative perspectives to decision theories. The notions of rational and right decisions are discussed in Section 4.1.2.

4.1.1 Descriptive and Normative

Theories of decisions have been divided into descriptive and normative theories. Descriptive theory accounts for generalities in the choices that people make. It plays explanatory and predictive roles in decision making [25]. It is an empirical discipline that emerged out of experimental psychology [146]. The actual decision making activities of humans have been observed to sometimes deviate from rationality models proposed by normative theories [109]. In order to study and examine how humans actually make decisions, descriptive theory lead to models and frameworks. Kahneman and Teversky [109] in their seminal work proposed a prospect theory of decision making which descriptively accounts for human decisions that may appear irrational under the expected utility theory. They found that decision-maker prefer certain gains as opposed to just probable outcomes. According to the finding [109], humans show aversion for certain losses regardless of the expected utility, i.e. gains, of their actions. Besides the certainty effect, the introspection effect was another major cause of the behaviors manifested in their findings. Prospect theory and many other models, such as Subjective Expected Utility [24], have proved the need for descriptive theory of decisions to explain human decision making that deviate from the postulates of expected utility models.

Normative theory of decisions is, however, more developed and prevalent than the

descriptive, perhaps because of its generalizability. The focus of normative theory is to address the behaviour of a rational decision-maker. It addresses what a decision-maker ought to do, given his preferences and beliefs about the world. Expected Utility is a dominant model employed in normative theory. Normative theory prescribes that a rational decision-maker would choose between alternative acts that maximize its expected utility. Interests in normative theory is justified, firstly, for its philosophical appeal, and secondly, it is pragmatic to deal with normative issues, because people make rational decisions most of the time.

4.1.2 Rational and Right Decisions

Rationality is a dominant idea in normative decision theory. A rational decision-maker is expected to behave in ways that are consistent with his desired objectives and beliefs about the state of the world. When decisions are made under risks, where the probabilities of the desired objectives are known, rationality, in a normative sense, presupposes maximization of expected value of the decisions [146]. The kind of rationality implied here is also called instrumental rationality [146]. However, there are instances where rational acts do not correspond to right decisions. A rational decision-maker, striving to maximize expected value, may take actions whose outcomes are inferior to other possible outcomes [146].

4.2 Decision Problems: Formalisation

A real-world decision problem can be represented in many ways. A decision matrix and tree [17] are popular representations. Mathematical representations are also widely employed, especially in undertakings that strive for a formalization of rational decision making. The representations and the counterpart formalizations aim at abstracting the problems. The process of abstraction typically reduces the elements of the problem into sets of acts, states and outcomes [11]. Equation 4.1 presents a functional formalisation of decision problems [17]:

$$D : A \times B \rightarrow O \tag{4.1}$$

In Equation 4.1, A is the set of acts, B the set of states of the world and O is the set of outcomes or objectives. For a decision-maker, whose agency is assumed, A is totally within his control. B and O are elements of the world of the decision-maker. For instance, the states of the world are given to him by factors outside his control and they form his belief about the world. He must accept them. Similarly, the set of possible outcomes are given. He may affect the values of the outcomes, but the set are externally imposed on him.

In the decision problem faced by Christopher Columbus, in his expedition to discover the Far East, the problem may be abstracted as follows:
A = $\{Sail\,westward\,(a_1),\ Stay\,at\,home\,(a_2)\}$,
B = $\{Christopher's Hypothesis\,of\,the\,world\,is\,true\,(s_1)$,
$There\,is\,no\,land\,westward\,(s_2), There\,are\,some\,land\,westward\,(s_3)\}$,
O = $\{He\,got\,rewarded\,(o_1), He\,got\,fame\,(o_2),\ He\,died\,on\,the\,expedition\,(o_3)$,
$Statusquo\,(o_4)\}$

A representation of the problem using a decision matrix is presented in Table 4.1. The decision tree in Figure 4.1 presents another representation of the same decision problem.

	Christopher's Hypothesis is True (s_1)	There is no land westward (s_2)	There is some land westward (s_3)
Sail Westward	He got rewarded	He died on the expedition	He got fame
Stay at home	Status quo	Status quo	Status quo

Table 4.1: A Decision Matrix: Christopher Columbus Expedition

4.3 Decision-Making Contexts

Theories of individual decisions, games theories and social choice theories are the major areas studied in decision theories. The decision problems that are examined in each of these areas are formulated in ways that account for the degree of certainty that surrounds

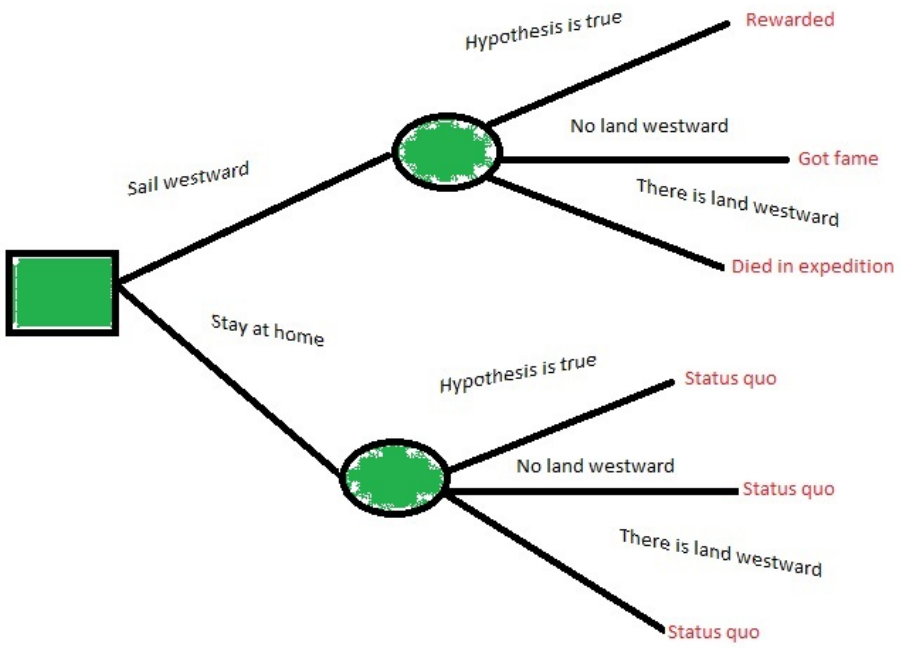


Figure 4.1: A Decision Tree: Christopher Columbus Expedition

the possible outcomes that are obtainable by decision-makers. This section examines some of the ideas associated with decision making contexts: ignorance in Section 4.3.1, risks in Section 4.3.2 and uncertainty in Section 4.3.3.

4.3.1 Ignorance

When Christopher Columbus set out to sail westward, he hoped that his hypothesis would be true. In addition, he did not know about the lands he finally found. Even if the lands were known to exist, because no person had ever completed such a journey before him, he could not ascertain the exact likelihood of finding the lands during his expedition. Faced with problems where the probability of the outcomes are not known, a rational decision-maker would be making decisions under ignorance. The notion of ignorance invoked here does not necessarily mean that the decision-maker is not aware of the possible outcomes. Also, the possible outcomes may turn out to be right. The

ignorance derives especially from the fact that the possible outcomes are not associated with any known measure of probability. The dominance principle [146] is a major decision rule used for decision problems associated with ignorance. Given two acts, a_1 and a_2, if every outcome of a_1 is at least as good as every corresponding outcome in a_2, then act a_1 is said to dominate a_2. According to the dominance principle, a rational decision-maker should select from non-dominated acts, where a non-dominated act is defined as one that is not dominated by any other act. There are weak and strong dominance [146]. In some decision problems, two or more acts may be non-dominated, therefore the dominance principle would be inadequate in solving such problems, because a single solution would not be available for such problems. Some other decision rules, such as Maximin and Leximin, Maximax and the Optimism-Pessimism rule, Minimax Regrest, etc., are also used [146].

4.3.2 Risks

Some decision problems have outcomes which are associated with probability measures. Decisions made under such problems are associated with risk. Put differently, making decisions under risk simply means that the consequences of actions associated with the decisions can be determined statistically. Decision under risks typically use the maximization of expected value as a decision rule.

The principle of maximization of expected value, which is the dominant decision rule used under risk, has been justified on two grounds. First is the justification that is rooted in the theory of large numbers and second, it is rooted in axiomatic analysis. Large number justification states that you would be better off in the long run if you maximize expected value. Axiomatic justification strives to derive the expected value principle from a set of fundamental axioms that are believed to approximate behaviors of rational decision-makers. More information can be found in [146].

Decision problems that involve ignorance can be converted to decision problems under risks by employing appropriate transformations that preserve the integrity of the underlying problems. For instance, a decision under ignorance may assume that all outcomes are equally likely, and assign probabilities to the outcomes accordingly.

4.3.3 Uncertainty

Uncertainty is a term that is widely used to mean ignorance or risk.

4.4 Preferences

Preference manifests when alternatives are ordered based on their values, or their utilities. Optimal choice results from a preferred ordering of the alternatives. In the Functional Formalization (Equation 4.1), a rational decision-maker facing a set of feasible alternative actions, A, and a set of preferred objectives, O, would be making optimal choices, given his beliefs about the world. The implication of the foregoing is that preference is at the core of rational decision making. Also implicit is the fact about the ordering of actions, as some actions would be deemed to be more valuable than others. The following sections present different notions of preferences. Section 4.4.1 presents instrumental preference. Intrinsic preference is discussed in Section 4.4.2. Section 4.4.3 highlights revealed preference.

4.4.1 Instrumental Preference

Preferences embody values. However, things in the world are valued either as a means to an end or as an end themselves. Instrumental value is attached to a thing because of another thing. It is a form of extrinsic value, and it carries with it the notion of a thing that works because of the ends it brings about. This is the notion of value that carries a functional connotation. Instrumental preferences are synonymous with instrumental value. In the context of the functional formulation in Equation 4.1, instrumental preferences are state dependent. They do not tell absolute truths about the world, and they tend to change whenever the beliefs of a rational decision-maker change. The sense of instrumentality assumed here is also a kind of preference over actions. John Dewey [30] and John Fagg Foster [186] are some of the 20th century advocates of instrumental value and preference.

4.4.2 Intrinsic Preference

There is another sense of value that carries with it the idea of goodness for its sake. For instance, the ten commandments are valued by Christians, not because they are means to some other ends, but because the laws contained in the commandments are assumed to have divine authority. The commandments are thus valued, because they are inherently right in the eyes of the believers of the laws. Intrinsic preference are state independent in the context of Equation 4.1. To a decision-maker, their values do not derive from actions, nor from the beliefs of the decision-maker about the states of the world. The sense of rightness they carry to a decision-maker is time and space independent. They are close to the idea of eternal truths, which do not depend on the decision contexts of a decision-maker.

4.4.3 Revealed Preference

Thomas Hobbes' writings carried the assumption that when faced with two alternatives, a and b, that choosing a over b is the same as liking a over b. Jeremy Bentham hypothesized about a utility measuring machine. However, it fell on Paul Samuelson to propose a more systematic way to measure preferences. Similar to some other 20th century economists who worked in the area of decision theory, Samuelson tried to separate decision making from psychology, or more in particular from cognitive psychology. Preferences, according to the revealed preference theory, is reflected in the behaviors of the decision-maker. With revealed preference, what goes in people's brains is not considered. There is also no need for the idea of a utility machine as proposed by Bentham. This is a kind of empiricism at work. By looking at what people do and generating data about choice behaviors, decision making can be explained and what people will do in similar circumstances can be predicted. Revealed preference, however, rests on two assumptions in order to be able to predict decision making. Firstly, that there is stability in the decision making contexts. This presupposes that the beliefs, or the states of the world, is unchanging and the preferences of the decision-maker is invariable. Where there is stability, inference can be made from choice data that will consistently match what a rational decision-maker would do in similar circumstances. Secondly, revealed prefer-

ence's predictive power breaks down when the decision-maker randomly makes decisions. Therefore, consistency is a key assumption in revealed preference theory.

4.5 Multi-Criteria Decision Making

Decisions, or courses of actions, are sometimes made in the context of multiple criteria, or objectives. Real-world decision problems are usually of this nature, where two or more criteria are the norm. These criteria are usually conflicting. Multi-criteria decision making (MCDM) [214] is used in business, finance, marketing, engineering, and a host of other fields of human endeavors. In our daily lives, we implicity weigh multiple criteria and then use intuition to arrive at the desired course of actions after due considerations are accorded to the consequences of the actions. Where the stakes are high, such as business and some mission critical systems in engineering, a systematic procedure is preferred in choosing the desired course of actions. Such a procedure firstly seeks a structuring of the decision problem, and then formalizes the solution steps. Working principles of MCDM is presented in Section 4.5.1. Section 4.5.2 presents the typologies of MCDM. Types of MCDM problems are discussed in Section 4.5.3. Approaches to solving MCDM problems are discussed in Sections 4.5.4, 4.5.5 and 4.5.6.

4.5.1 Working Principles

Different approaches exist for solving MCDM problems. Approaches vary depending on the type of decision problem. However, the approaches have a basic working principle, which generally include:

- Selection of criteria - criteria should not exhibit dependency, and they should be measurable in a definite scale and they should be related to the actions.
- Selection of courses of actions, or simply actions - actions should be comparable, real and feasible.
- Selection of weighing methods to represent importance of the actions and criteria.
- Method of aggregation - to arrive at the overall importance of the actions.

- Decision making using the aggregated results.

4.5.2 Typologies

MCDM problems may be classified using different criteria. When information about the actions, or the alternatives, are used and the number of such alternatives are finite, the class of decision problems is called evaluation problems [124]. The alternatives are unambiguously known at the start of the solution process. Each alternative is associated with a performance measure, i.e. a measure of importance or priority. The performance measure reflects the degree of relevance of the alternative in the context of the criteria used in the decision problem. Research has been extensively conducted on methods that are well suited for solving evaluation problems. Full aggregation, outranking and goal/aspiration setting approaches are major categories of solution methods widely used in solving evaluation problems. Detailed information about these methods can be found in [188].

However, there is another class of MCDM problems. In these problems, the number of alternative actions are infinite, or many and countably finite. These are called design problems. Design problems have alternatives that are not known at the beginning of the solution procedure. Solving this class of problems usually requires the formulation of a mathematical model. The objective functions of the model constitute the criteria associated with the decision problem. The domain of the model, or the mathematical functions, is defined by the variables representing the alternatives, or the course of actions underlying the decision problem. Design problems have typically been solved by methods prevalent in operations research, namely numerical analysis approaches. Nature-inspired and iterative methods have also been widely used in solving design problems [125]. Methods that solve design problems usually reformulate the problems as optimization problems, after incorporating the decision-makers' preferences into the mathematical model that embodies the problem. Decision-makers' preferences are incorporated into the mathematical formulation of design problems at the beginning of the solution process, in the course of the process or at the end, leading to the notions of a priori, interactive and posteriori preference incorporations [125] respectively.

Subsequent sections in this chapter are largely devoted to evaluation problems. Pre-

vious chapters in this dissertation have presented ideas, concepts and leading solution approaches that are related to design problems. In particular they related to ideas about nature-inspired approaches that have been widely used in solving multi-criteria decision problems with infinite number of alternatives, or course of actions. It is important to note that the alternatives in design problems are referred to as solutions by methods that are used to solve these problems.

4.5.3 Problem Types

MCDM problems are formulated to serve different ends. A decision-maker may be interested in making a choice between the alternative courses of actions. Choice problems are geared at achieving this end. A choice may be about selecting an alternative that best satisfies the criteria in line with defined preferences of the decision-maker. A choice problem [100] [163] may also be used to select a set of alternatives where a number of alternatives prove non-inferior to any other alternatives.

Sorting problems are not widely used as choice problems. With sorting problems, alternatives are sorted into groups in accordance with defined criteria. Alternatives in the same group are setup to facilitate better descriptions and improved predictions. Sorting can be used, in addition to a choice problem, as a preliminary step to short-list alternatives. For repetitive and automatized ends, sorting problems are a good choice [100].

Where there is no complete order relation that can be used on the alternatives, partial ordering schemes can be used. Ranking problems use partial ordering schemes, where pairwise comparison is used for the alternatives. Partial comparisons can generate measures of local importance, or priorities, between pairs of alternatives. Aggregation techniques can then be used to synthesize global measures of importance, which factor in all criteria considered in the decision problem.

Descriptive problems, another type of decision problems presented by Roy [163], define the characteristics of decision problems, by describing actions and their consequences. Such descriptions usually help to foster a deeper understanding of the problems, and can perhaps be used as a preliminary step to employing other richer formulations of the problems.

The problem types presented in this section are not exhaustive, but they demonstrate some of the leading types of problems that are widely studied in the MCDM community.

4.5.4 Solution Approaches: Full Aggregation

The weights, importance or priorities, associated with alternatives in a decision problem, are combined for all criteria. Such a combination uses relations/functions that compute the summation, product, or some other means of aggregating the weighted criteria. Analytical Hierarchy Process (AHP), first developed by Saaty [2], is one of the methods that uses the aggregation approach to solve multi-criteria decision problems.

A decision problem is first decomposed into an hierarchy (refer to Figure 4.2). At the top of the hierarchy is the goal. Next on the hierarchy is the list of criteria upon which the decision would be based. The courses of actions, or alternatives, occupy the third layer in the hierarchy. Criteria are weighted, where the weights associated with criteria are measures of importance attached to each criteria by a decision-maker.

The alternatives are then prioritized, for each criterion, in a process of pairwise comparison. This process starts with a criterion. The alternative actions are selected in a pairwise fashion. A measure of importance is assigned to the two alternatives in the pair, based on their relative importance to the selected criterion. This process of pairing and computation of per-criterion importance is done for all actions defining the decision problem. The process is extended to all other criteria, resulting in a comparative matrix, where a pair of actions are compared, per criterion. The number of criteria correspond to the number of comparative matrices.

The obtained comparative matrices all contain local priorities, or importance, associated with a pair of actions, or alternatives. The local priorities are then aggregated in order to arrive at global priorities for each action. Equations 4.2, 4.3 and 4.4 present a method of computing the priorities of the alternatives:

$$A^* = (a^*_{ij}) \tag{4.2}$$

where A^* is the comparative matrix for a criterion.

$$r^*_i = \sum_j a^*_{ij} \tag{4.3}$$

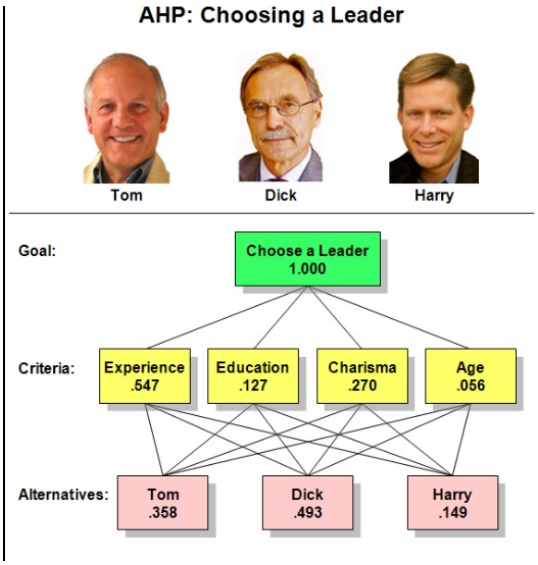


Figure 4.2: AHP. Wiki Source - [201]

where r_i^* is the summation of elements of row i in the comparative matrix A^*.

$$p_i^* = \frac{r_i^*}{\sum_j r_j^*} \tag{4.4}$$

where p_i^* is the priority of alternative i for a given criterion.

The last phase of the AHP is the computation of priorities, for each alternative, for all criteria. One method of performing this aggregation is as follows:

$$P_i^* = \sum_j w_j^* . p_{ij}^* \tag{4.5}$$

where P_i^* is the priority of alternative i, after aggregating all the per-criterion priorities for the alternative, and w_j^* is the weight, or priority, associated with criterion j.

4.5.5 Solution Approaches: Outranking

The Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) is widely used to solve MCDM problems. It uses outranking of priorities and applies the following steps:

- computation of unicriterion preference degrees for each ordered pair of alternatives
- computation of multi-criteria preference degrees
- computation of multi-criteria flows.

First, the alternatives constituting the MCDM problem are assigned values for every criterion - refer to Figure 4.3. In the figure, $\mathbf{a} = (a_1, ..., a_n)$ is the set of n actions, while $\mathbf{f} = (f_1, ..., f_q)$ is the set of q criteria.

	$f_1(\cdot)$	$f_2(\cdot)$	$\cdots$	$f_j(\cdot)$	$\cdots$	$f_q(\cdot)$
a_1	$f_1(a_1)$	$f_2(a_1)$	$\cdots$	$f_j(a_1)$	$\cdots$	$f_q(a_1)$
a_2	$f_1(a_2)$	$f_2(a_2)$	$\cdots$	$f_j(a_2)$	$\cdots$	$f_q(a_2)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
a_i	$f_1(a_i)$	$f_2(a_i)$	$\cdots$	$f_j(a_i)$	$\cdots$	$f_q(a_i)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
a_n	$f_1(a_n)$	$f_2(a_n)$	$\cdots$	$f_j(a_n)$	$\cdots$	$f_q(a_n)$

Figure 4.3: PROMETHEE. Wiki Source: [202]

A preliminary step uses the data in Figure 4.3 as inputs. This step makes pairwise comparisons between the alternatives. The comparison is applied on each criterion using Equation 4.6:

$$d_k(a_i, a_j) = f_k(a_i) - f_k(a_j) \tag{4.6}$$

where k is the index number of the criterion in the set of criteria. Preference degrees are then computed, using the differences computed in Equation 4.6 and a preference function. Different preference functions are discussed in [100] [202]. A linear preference function is presented in Equation 4.7. In the equation, p^{**} and q^{**} are the preference and indifference threshold respectively, defined by the decision-maker.

$$P_k(a_i, a_j) = \begin{cases} 0 & if\ d_k(a_i, a_j) \leq q^{**} \\ \frac{d_k(a_i,a_j)-q^{**}}{p^{**}-q^{**}} & if\ q^{**} \leq d_k(a_i, a_j) \leq p^{**} \\ 1 & if\ d_k(a_i, a_j) \geq p^{**} \end{cases} \tag{4.7}$$

The computed unicriterion preference degrees are then converted to multi-criteria preference degrees using Equation 4.8, which culminates in the second major step of PROMETHEE.

$$\pi^*(a_i, a_j) = \sum_{k=1}^{q} P_k(a_i, a_j).w_k \tag{4.8}$$

The third step of PROMETHEE computes the multi-criteria preference flows as follows:

Positive preference flows, which is a measure of how much an alternative is preferred over all other alternatives, is computed according to Equation 4.9, where n is the number of alternatives.

$$\phi^+(a_i) = \frac{\sum_{j=1}^{n} \pi^*(a_i, a_j)}{n-1} \tag{4.9}$$

Negative preference flows, which is a measure of how much other alternatives are preferred to the subject alternative, is computed as follows:

$$\phi^-(a_i) = \frac{\sum_{j=1}^{n} \pi(a_j, a_i)}{n-1} \tag{4.10}$$

The net preference flow is then computed using Equation 4.11.

$$\phi(a_i) = \phi^+(a_i) - \phi^-(a_i) \tag{4.11}$$

where $\phi(a_i) \in [-1, 1]$ and $\sum_i \phi(a_i) = 0$

Alternatives are then ranked using the computed net preference flows.

4.5.6 Solution Approaches: Goal Setting

MCDM problems can be solved by methods that set aspiration levels or goals. Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [100] is one of such methods. In TOPSIS, chosen actions or alternatives should have the shortest distance to the ideal alternative and the longest distance to the anti-ideal alternative. Figure 4.4 shows how two alternatives, $A1$ and $A2$, are related to the ideal and anti-ideal alternatives.

The idea underlying the TOPSIS method requires a decision-maker to supply the ideal and the anti-ideal alternatives at the beginning of the solution process. Alternatives

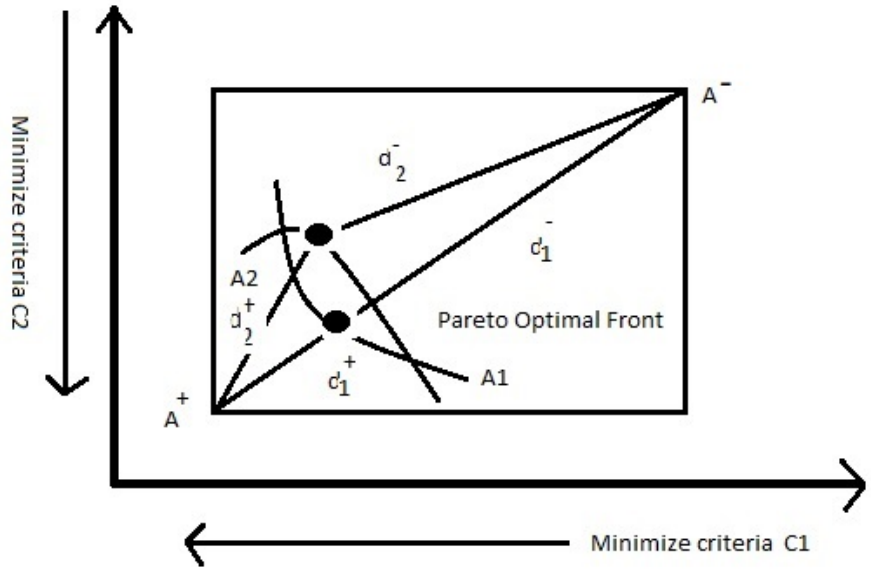


Figure 4.4: TOPSIS - Ideal and Anti-ideal Alternatives

constituting the decision problem are then compared on how much they satisfy the distance requirement to the ideal and anti-ideal alternatives. The solution process in TOPSIS is carried out as follows:

First Step: Create a performance matrix $X = (x_{ij})$, where x_{ij} is the performance of alternative i in criterion j. The dimension of X is n x q.

Second Step: The performances in the matrix X, as per the First Step, are normalised. Equation 4.12 presents one possible normalization technique:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{q} x_{ij}^2}} \tag{4.12}$$

Third Step: The normalized matrix from the Second Step is weighted. Equation 4.13 presents the computation of a weighted normalized matrix.

$$v_{ij} = w_j.r_{ij} \tag{4.13}$$

where w_j is the weight associated with criterion j.

Fourth Step: The ideal and the anti-ideal alternatives are computed. One method of computing these virtual alternatives is to select the best and the worst performance on

each criterion in the matrix $V = (v_{ij})$ in the Third Step above. The ideal alternative is calculated as follows:

$$\text{Ideal alternative: } A^+ = (v_1^+, ..., v_q^+) \tag{4.14}$$

where v_j^+ is the ideal value of alternatives for the criterion j.
Equation 4.15 presents the computation of the anti-ideal alternative.

$$\text{Anti-ideal alternative: } A^- = (v_1^-, ..., v_q^-) \tag{4.15}$$

where v_j^- is the anti-ideal value of alternatives for the criterion j.

Fifth Step: The distance of each alternative to the ideal and anti-ideal alternatives are computed. Equation 4.16 presents the distance computation for the ideal alternative.

$$d_a^+ = \sum_j^q (v_j^+ - v_{aj})^2,\ a = 1, ..., n \tag{4.16}$$

And Equation 4.17 presents the distance computation for the anti-ideal alternative.

$$d_a^- = \sum_j^q (v_j^- - v_{aj})^2,\ a = 1, ..., n \tag{4.17}$$

Figure 4.4 visualises the ideal and the anti-ideal alternatives and distances of two alternatives, relative to the virtual alternatives.

Euclidean distance is used in Equations 4.16 and 4.17. However, other measures of distance can also be used.

Sixth Step: The similarity of each alternative to the worst condition is then computed. This is also referred to as the relative closeness coefficient. The computation of the closeness coefficient is presented in Equation 4.18.

$$C_a = \frac{d_a^-}{d_a^+ + d_a^-}\ a = 1, ..., n \tag{4.18}$$

where C_a lies between 0 and 1. Preferred alternatives are closer to the ideal and further from the anti-ideal. Therefore C_a tends to be closer to 1.

4.6 Summary

This chapter presented key concepts underlying decision theory, including decision making. Some theoretical perspectives to decision making were discussed. Formalisation of decision problems and the contexts in which humans make decisions were highlighted. The notion of preferences and their formalisations were also presented. The discipline of MCDM was discussed, as well as the different approaches used in solving MCDM problems. A representative method of each approach, i.e. full aggregation, outranking and goal setting, was presented. The next chapter presents the nature-inspired algorithm proposed in this dissertation.

Chapter 5

Differential Evolution Algorithm I

DMOO is a problem-solving procedure that maximizes/minimizes objectives, or goals, usually under some imposed constraints. The procedure is, however, often mathematical and/or algorithmic. DMOOPs are characterized by two or more objectives, where the objectives and/or the constraints change with time [87] [134]. At least two of the objectives of DMOOPs are usually in conflict, therefore algorithms search for optimal trade-off solutions [156].

There are real-world DMOOPs, however the experimental study which is presented in this chapter focuses on artificial DMOOPs [61] [88]. Artificial problems typically embody the essential features of real-world problems and at the same time offer researchers a way to analyse the performance of algorithms on problems that are well understood. The study presented in this chapter proposes a new DMOA, called 2DEVENS.

Nature-inspired algorithms, such as DMOAs, are popularly used in computing optimal trade-off solutions to DMOOPs. DMOAs employ iterative search procedures, which mimic evolutionary processes in nature. DMOAs' search space is sub-divided into a solution space or decision space, and an objective space [48].

DNSGA-II is a DMOA originally proposed in [47] and discussed in Section 3.2.6. DNSGA-II is a nature-inspired algorithm that employs metaphors of, and simulates processes in, natural evolution. Major processes of natural evolution, which were employed in Darwin's theory [164], are simulated by DNSGA-II using operators such as selection, crossover and mutation. The success of DNSGA-II in solving DMOOPs has been widely

reported in the literature. For instance, in [47] DNSGA-II successfully tracked the POFs of hydro-power scheduling optimization problems.

Group behaviours - such as swarming, flocking and herding - among animals have been a source of inspiration for DMOAs. PSO is an optimization procedure that simulates group behaviour in a swarm of birds [58]. Individuals in a swarm are referred to as particles. Each particle adapts its behavior by emulating his best success and the success of the best particle in the swarm, which means PSO particles are capable of self- and group-learning [31]. The DVEPSO [84] algorithm is a DMOA that uses a cooperative PSO strategy and with proven records of solving DMOOPs [84] [86] and is discussed in Section 3.3.6. Helbig and Engelbrecht [84] [85] demonstrated the ability of DVEPSO to successfully find the POFs of a number of benchmark functions, such as FDA1, FDA2, HE1, HE2, etc., which variously present different types of difficulties to DMOAs.

DE algorithms share many similarities with other nature-inspired algorithms, such as genetic algorithms. However, distance and step sizes are the defining measures used by DE to drive the search for optimal trade-off solutions [148] [180].

Unlike approaches employed in [104] [127] [195], the proposed algorithm combines the non-dominated sort and vector evaluation schemes in driving the proposed DE optimization procedure. Also, the proposed algorithm uses an archive to keep track of the changing Pareto-optimal solutions. Changes in the dynamic environment are detected by computing sentry vectors [84], and the magnitude of the changes in the objective values of the sentry vectors between two consecutive environments is used as an estimate of a change in the simulated environment.

In the study presented in this chapter, 2DEVENS is compared with DNSGA-II and DVEPSO for different measures of performance [90] and for different experimental configurations, on different DMOOPs.

The rest of the chapter is organized as follows: Section 5.1 discusses key concepts, mathematics and algorithms required to facilitate understanding of the rest of the chapter. Section 5.2 discusses the experimental configurations, benchmark functions, performance measures and statistical analyses employed in the experimental study. Results of the experimental study are discussed in Section 5.3, and Section 5.4 draws conclusions from the results. A summary of this chapter is presented in Section 5.5.

5.1 Background

This section presents key concepts, mathematics and algorithms required to facilitate understanding of the other sections of this study. Section 5.1.1 presents background information on DMOO. Sections 5.1.2 and 5.1.3 discuss DMOAs and the DE algorithm respectively. Vector evaluation and non-dominated sorting are discussed in Sections 5.1.4 and 5.1.5.

5.1.1 Dynamic Multi-Objective Optimization

DMOOPs have conflicting objectives, therefore no single optimal solution exists. Optimal trade-off solutions are computed using the Pareto dominance relation [140]. A solution dominates another solution if and only if both solutions are at least equal in quality with regards to all objectives and the dominating solution is better than the other solution for at least one of the objectives. For any given DMOOP, the set of Pareto dominating solutions is called the POS. The set that corresponds to the objective values of the POS is called the POF [48]. When DMOAs solve DMOOPs, they try to find the POS and POF for the problems. However, because the objectives and/or constraints of DMOOPs change with time, the POS and/or POF may also change with time. Therefore, DMOAs should be capable of tracking the changing POS and/or POF. Meanwhile, DMOAs are expected to yield very diversified solution sets, which are also very good approximations of the true solutions of the DMOOPs, i.e. accurate solutions.

5.1.2 Dynamic Multi-Objective Optimization Algorithm

Algorithm 8 presents the dynamic multi-objective evolutionary algorithm, which serves as the entry point to the process of optimizing DMOOPs that are studied in this chapter. The core of this algorithm is a procedure called Optimizer, which is the main algorithm proposed in this chapter.

A high-level description of Algorithm 8 is as follows:

- Data structures and counters, such as archive, iteration counter and time, are set.

- A random population of individuals, which are potential solutions to the associated problem, is created.
- The non-dominated individuals in the current population are obtained, and they are assigned to the archive.
- The following set of algorithmic instructions are processed until the required exit condition, i.e. maximum iteration count, is satisfied:
  - The current time, in computational terms, is obtained using the severity and frequency of change
  - The Optimizer is invoked at this time, t
  - Sentry particles are evaluated for a change in the simulation environment
  - If there is a change in environment, required updates, e.g. updating the archive, are performed
  - Then, the iteration counter is incremented by one, and this block of instructions are re-processed until the exit condition, which is previously defined, is satisfied.

5.1.3 Differential Evolution Algorithm

This section presents the proposed DE algorithm (refer to Algorithm 9). In Algorithm 9, getTrialVector(β, v, P_{gen}, F, t) is a procedure that computes the trial vector. This procedure computes the target vector, given the input parameters passed to it. In the proposed algorithm, the population of solutions or vectors, P_{gen}, is divided into groups, sub-populations or hypercubes, with the number of groups being equal to the number of objective functions to be optimized. The process of grouping occurs when the population of solutions, or vectors, is initialized (refer to Algorithm 8). During the initialization, the solution space is divided into sub-populations. Solutions are then drawn from the sub-populations, which constitute the groupings of the solutions. Next generation vectors are selected using the procedure getNextGenerationVectors.

Algorithm 8 Dynamic Multi-Objective Optimization

Input: freq, severity, maxiteration, dMOOP

Output: $\{POS_t\}$, $\{POF_t\}$

Set population size, N
Set archive max size, SizeArchive
Initialize the iteration counter, iteration $\leftarrow 0$
Initialize time, $t \leftarrow 0$
Initialize$(P_t, freq, severity, dMOOP, t)$
{Initialize population of solutions, P_t}
AssignNonDominatedToArchive$(P_t, dMOOP, t)$
loop:
if *iteration* $\leq$ *maxiteration* **then**
    $t \leftarrow 1/severity * floor(iteration/freq)$
    {Frequency of environment change: $freq$}
    {Severity of environment change: $severity$}
    Optimizer$(P_t, dMOOP, t)$
    Pick sentry solutions
    if ENV changes$(P_t, dMOOP, t)$ **then**
        ProcessChange$(P_t, freq, severity, dMOOP, t)$
    end if
    iteration $\leftarrow$ *iteration* $+ 1$
    goto *loop*
end if

5.1.4 Vector Evaluation Procedure

The vector evaluation strategy divides a population of solutions into groups or sub-populations, and each sub-population is then optimized for a single objective. The strategy is employed in the proposed algorithm as follows: The procedure getTrialVector (refer to Algorithm 9) computes the target vector, which together with a difference vector are used to compute the trial vector [144]. Given a parent vector, the best vector in the adjacent group or sub-population of the parent is chosen as the target vector using the vector evaluation scheme, which was proposed in [167] and applied in [141] [143] [144].

Algorithm 9 Optimizer

Input: P, F, t

Output: POS_t, POF_t

{β: scaling factor set per algorithm}
{p_r: recombination prob set per algorithm}
{$maxgen(\geq 1)$: no of function evaluations set per algorithm}
{P: current population of vectors}
{F: multi-objective function to be optimized}
{t: current time}
{v: a vector drawn from a population of vectors}

1: *Set the generation counter, gen = 1*
2: $P_{gen} \leftarrow P$
3: $V \leftarrow \emptyset$
4: loop:
5: **if** $gen \leq maxgen$ **then**
6: nextVector:
7: **if** $moreUnprocessed(v \in P_{gen})$ **then**
8: $v' \leftarrow getTrialVector(\beta,v,P_{gen},F,t)$
9: $v'' \leftarrow getChildVector(p_r,v', v,F,t)$
10: $V \leftarrow V \cup \{v, v''\}$
11: $markAsProcessed(v \in P_{gen})$
12: **goto** *nextVector*
13: **end if**
14: $P_{gen} \leftarrow getNextGenerationVectors(V)$
15: $gen \leftarrow gen + 1$
16: $V \leftarrow \emptyset$
17: **goto** *loop*
18: **end if**
19: $AssignNonDominatedToArchive(P_{maxgen}, F, t)$

The target vector and the difference vector are combined to produce the new trial vector, which is combined in a crossover procedure with the parent vector to construct the child vector [144]. The two vectors that constitute the difference vector are randomly selected

from the parent vector's group.

5.1.5 Non-Dominated Sorting

The procedure getChildVector in the proposed algorithm (refer to Algorithm 9) combines the trial vector with the parent vector in a binomial crossover [58] [212] procedure to produce the child vector. The child vectors produced from all the parent vectors are pooled together with the parent vectors, and a set of next generation vectors is selected from the pool using the non-dominated sorting scheme [74]. The procedure getNextGenerationVectors in Algorithm 9 implements the non-dominated sorting scheme.

5.2 Experiment

This section discusses the experimental setup used for this study. Section 5.2.1 discusses the benchmark functions used in the study. Section 5.2.2 discusses the performance measures used in a comparative analysis of the algorithms considered in the study. Algorithmic setups used by the proposed algorithm are discussed in Section 5.2.3, while the statistical analysis approach used in analysing the results of this study is discussed in Section 5.2.4.

5.2.1 Benchmark Functions

Four DMOOPs, with various τ_t-n_t combinations, were used in this study. The experimental configurations used for these problems are:

Table 5.1: First Experimental Study: Configuration

S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
1	FDA1	4	10	16	20	30
2	FDA1	5	10	20	20	30
3	FDA1	2	10	8	20	30
4	FDA1	4	1	16	20	30
5	FDA1	5	1	20	20	30
6	FDA1	2	1	8	20	30
S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
7	FDA5	4	10	16	20	30
8	FDA5	5	10	20	20	30
9	FDA5	2	10	8	20	30
10	FDA5	4	1	16	20	30
11	FDA5	5	1	20	20	30
12	FDA5	2	1	8	20	30
S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
13	dMOP2	4	10	16	20	30
14	dMOP2	5	10	20	20	30
15	dMOP2	2	10	8	20	30
16	dMOP2	4	1	16	20	30
17	dMOP2	5	1	20	20	30
18	dMOP2	2	1	8	20	30
S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
19	dMOP3	4	10	16	20	30
20	dMOP3	5	10	20	20	30
21	dMOP3	2	10	8	20	30
22	dMOP3	4	1	16	20	30
23	dMOP3	5	1	20	20	30
24	dMOP3	2	1	8	20	30

The following symbols were used in Table 5.1:

τ_t: frequency of change.

n_t: severity of change.

$c(f(x))$: number of function evaluations per iteration.

$\sigma(runs)$: number of runs per configuration.

$FDA1$: type I DMOOP (POS is dynamic, POF is static), $POF = 1-\sqrt{f_1}$ and is convex, POS is $x_i = G(t)$ [61] [84].

$FDA5$: type II DMOOP (POS and POF are dynamic); for 3 objectives, POF $= f_1^2 + f_2^2 + f_3^2 = (1 + G(t))^2$ and is non-convex; POS is $x_i = G(t)$ [61] [84].

$dMOP2$: POF changes from convex to concave; type II DMOOP; POF $= 1 - f_1^{H(t)}$; POS is $x_i = G(t)$ [73] [84].

$DMOP3$: type I DMOOP (POS is dynamic, POF is static), $POF = 1 - \sqrt{f_1}$ and is

convex, POS is $x_i = G(t)$ [61] [84].

5.2.2 Performance Measures

Five performance measures were used in this study. Each of the measures were computed immediately before a change in the environment occurred. This was done for thirty runs. An average of the values of the thirty runs was then computed for each measure in each environment. The performance measures are as follows:

- Accuracy Measure (acc): measures how accurate a DMOA is able to approximate the true POF of a DMOOP [84] [87] [89]. The lower the value of acc, the better the performance of the algorithm.
- Stability Measure (stab): quantifies the effect of environment changes on the accuracy measure [84] [87] [89]. The lower the value of this measure, the better.
- Hypervolume Ratio (hvr): measures the proportion of the objective space that is covered by a non-dominated set without suffering from the bias of convex regions seen with the HyperVolume measure [90]. hvr was proposed in [192]. The higher the value of this measure, the better the algorithm performed.
- Reactivity Measure (react): measures how long it takes a DMOA to recover after a change in the environment occurred. The length of time it takes to reach a specified accuracy threshold is employed in computing this measure [84]. It was originally proposed in [176]. The lower the value of this measure, the better.
- Number of Non-Dominated Solutions (NS): measures the number of non-dominated solutions in the archive; it measures the size of the approximated POF [84].

5.2.3 Algorithmic Setup

1. The proposed algorithm is characterized as DE/best/1/bin.
2. Trial vector: to generate a trial vector from a parent vector during the mutation phase of the algorithm, the best vector in the adjacent sub-population or

hypercube as the parent vector is chosen as the target vector. The number of hypercubes or sub-populations used by the algorithm is the same as the number of objective functions in the underlying DMOOP. Each hypercube optimizes one objective function.

3. One difference vector: two randomly chosen vectors from the parent vector's hypercube are used to form a difference vector.

4. Crossover: Binomial crossover [58] is used because of its viability as a crossover method in DE algorithms.

5. Scaling factor, $\beta \in (0.4, 1)$.

6. Recombination probability, $p_r = 0.8$.

7. DE convergence is insensitive to the control parameters [58]. Therefore the proposed DE algorithm randomly chooses the scaling factor and fixes the recombination probability as defined above.

8. Neighbourhood topology: ring (the same for DVEPSO).

9. Population size: 50 (the same for DNSGA-II and DVEPSO).

10. Maximum archive size: 100 (the same for DNSGA-II and DVEPSO).

11. Control parameters for DVEPSO: Cognition learning factor is 0.5, the social learning factor is 0.5 and the inertia weight is 0.8. These sample parameter values guarantee convergence, because they satisfy the convergence criterion in [58].

12. DVEPSO in this study uses the lbest PSO strategy in a ring topology in order to compute the velocities of the particles in the optimization procedure [58].

13. DVEPSO uses a velocity clamping rule that randomly forces particles' velocities to fall between the range of (-1,1). The range is chosen to ensure a reasonable balance between the exploration and exploitation capabilities of the algorithm.

14. Control parameters for DNSGA-II: for crossover, the distribution index is 2 and crossover probability is 0.9 (refer to [79], [80]). Polynomial mutation [79] is used by the algorithm. Mutation parameters employ commonly used values: distribution index of 20 and mutation probability of 1/6 [80], where 6 is the number of decision variables.

5.2.4 Statistical Analysis

A statistical analysis of the algorithms' performance was done in accordance with the $wins-losses_B$ algorithm proposed in [87], which is presented in Algorithm 1. The major steps of the algorithm are:

- Kruskal-Wallis tests are performed on the performance measure values.
- If there is a statistical significant difference, then:
  - For each DMOA-pair, the following is performed:
    - Conduct Mann-Whitney U test on performance measure values
    - If there is a statistical significant difference, wins and losses are assigned.

The average performance measure value of each time step just before a change in the environment occurred are used to award wins and losses. This is done when the Mann-Whitney U test indicates that there is a statistical significant difference. At each time step just before a change in the simulated environment happened, the average performance measure values of the DMOA-pair are compared. The DMOA with the best performance measure value is awarded a win and the other is awarded a loss. Also, to avoid skewness in the results that are obtained, the wins and losses are normalized as proposed in [87].

The algorithm was implemented in R [149] and the Kruskal-Wallis and Mann Whitney U statistical functions in R were used as stipulated in [87].

5.3 Results and Discussions

This section presents the results of the study and how the proposed algorithm, 2DEVENS, performed in relation to DVEPSO and DNSGA-II. The winning algorithm in each table presented in this section has a rank number indicated in bold. An experimental configuration in this section, and in any other sections of this chapter, is the same as a pair of (severity of change, frequency of change) values used in the study.

Table 5.2 presents the overall wins-losses of the three algorithms considered in this study. DVEPSO obtained more losses than wins, therefore it performed poorly compared to DNSGA-II and 2DEVENS. DNSGA-II ranked first, followed closely by 2DEVENS, and DVEPSO ranked last.

Table 5.3 presents the wins-losses of the algorithms in relation to the five performance measures employed in the study. In Table 5.3, 2DEVENS ranked first for both the accuracy and stability measures. DVEPSO ranked second for the two measures, while DNSGA-II ranked last. DNSGA-II recorded the highest number of losses for the two measures. DNSGA-II ranked first for the hvr and NS measures. 2DEVENS ranked second for the two measures, while DVEPSO ranked last. DVEPSO, however, ranked first for the reactivity measure and 2DEVENS ranked last. 2DEVENS and DVEPSO each recorded their highest number of wins with the accuracy measure, and their individual highest number of losses was recorded for the hvr measure. The highest number of wins recorded by DNSGA-II was for hvr, and its highest number of losses was recorded for the accuracy measure.

In Table 5.4, where the performance measure is acc, 2DEVENS won the first three experimental configurations, while DVEPSO won the remaining three configurations. DNSGA-II tied with 2DEVENS for the first experimental configuration in Table 5.4. In all the configurations where 2DEVENS ranked first, DVEPSO ranked last. 2DEVENS recorded its highest number of wins for the experimental configuration ($n_t = 10, \tau_t = 5$) and recorded its highest number of losses for the configuration ($n_t = 10, \tau_t = 4$). DVEPSO recorded its highest number of wins for the experimental configuration ($n_t = 1, \tau_t = 5$) and the algorithm recorded its highest number of losses for the configuration ($n_t = 10, \tau_t = 5$). DNSGA-II recorded its highest number of wins for the experimental configuration ($n_t = 10, \tau_t = 5$) and its highest number of losses for the configuration

$(n_t = 1, \tau_t = 5)$. DVEPSO and DNSGA-II obtained more losses than wins for three configurations. DVEPSO performed poorly when $(n_t = 10)$ and DNSGA-II when $(n_t = 1)$. 2DEVENS ranked first or second for all configurations, and it obtained more wins than losses for five configurations. 2DEVENS obtained equal number of wins and losses for $(n_t = 1, \tau_t = 5)$.

In Table 5.5, where the performance measure is stab, 2DEVENS won the following configurations: $(n_t = 10, \tau_t = 2)\,(n_t = 1, \tau_t = 4)\,(n_t = 1, \tau_t = 2)$. 2DEVENS recorded its highest number of wins for the configuration $(n_t = 10, \tau_t = 2)$ and its highest number of losses for the configuration $(n_t = 10, \tau_t = 5)$. Its worst ranking was the second position. DVEPSO ranked first for the following configurations: $(n_t = 10, \tau_t = 4)\,(n_t = 10, \tau_t = 5)\,(n_t = 10, \tau_t = 2)$. It tied with 2DEVENS for the configuration $(n_t = 10, \tau_t = 2)$. Its highest number of losses was recorded for $(n_t = 10, \tau_t = 2)\,(n_t = 1, \tau_t = 2)$. DNSGA-II ranked first for $(n_t = 1, \tau_t = 5)$. DNSGA-II highest number of wins was for $(n_t = 10, \tau_t = 4)\,(n_t = 10, \tau_t = 5)\,(n_t = 10, \tau_t = 2)$. Its highest number of losses was also recorded for the configurations where it recorded the highest number of wins. 2DEVENS obtained more losses than wins for $(n_t = 10, \tau_t = 5)$ and was ranked first or second for all configurations. DVEPSO obtained more losses than wins for $(n_t = 1)$. DNSGA-II only obtained more wins than losses for $(n_t = 1, \tau_t = 5)$.

In Table 5.6, performances of the three algorithms are presented for the hvr. 2DEVENS ranked second for all the experimental configurations. The highest number of wins recorded by 2DEVENS was for the configuration $(n_t = 1, \tau_t = 5)$, while the highest number of losses was recorded for the configuration $(n_t = 1, \tau_t = 4)$. DVEPSO ranked last for all the configurations. DVEPSO recorded zero number of wins for the configuration $(n_t = 10, \tau_t = 4)$, which was also its worst number of wins for all the configurations presented. The highest number of wins recorded by DVEPSO was for the configuration $(n_t = 1, \tau_t = 2)$, and its highest number of losses was recorded for the configuration $(n_t = 10, \tau_t = 4)$. DNSGA-II ranked first for all the experimental configurations. Its highest number of wins was recorded for $(n_t = 10, \tau_t = 4)\,(n_t = 10, \tau_t = 2)$ and its highest number of losses was recorded for $(n_t = 1, \tau_t = 2)$. 2DEVENS ranked second for all configurations, obtained only a few more wins than losses for three configurations, equal number of wins and losses for two configurations and more losses than wins for one

configuration. DVEPSO obtained the worst rank for all configurations and was awarded more losses than wins for all configurations. DNSGA-II ranked first for all configurations and obtained more wins than losses for all configurations.

In Table 5.7, the algorithms' performances are presented for the reactivity measure under different experimental configurations. 2DEVENS ranked second for the following configurations: $(n_t = 10, \tau_t = 4)\,(n_t = 1, \tau_t = 4)$, while it ranked third for the remaining configurations. 2DEVENS's highest number of wins was for $(n_t = 1, \tau_t = 4)$, and its highest number of losses was for $(n_t = 1, \tau_t = 2)$. DVEPSO ranked first for all the configurations. DVEPSO recorded its highest number of wins for the configuration $(n_t = 1, \tau_t = 4)$, and its highest number of losses was recorded for the configuration $(n_t = 1, \tau_t = 2)$. DNSGA-II ranked third for the configuration $(n_t = 1, \tau_t = 4)$, and it ranked second for the remaining configurations. It tied for the second position with 2DEVENS for $(n_t = 10, \tau_t = 4)$.

In Table 5.8 the performances of the three algorithms are presented for NS. 2DEVENS ranked first for $(n_t = 1, \tau_t = 4)\,(n_t = 1, \tau_t = 5)$, and it tied with DNSGA-II for $(n_t = 1, \tau_t = 5)$. In the remaining configurations, 2DEVENS ranked second, where it was outranked by DNSGA-II. 2DEVENS recorded its highest number of wins for the configurations $(n_t = 1, \tau_t = 4)\,(n_t = 1, \tau_t = 5)$, and its highest number of losses was recorded for $(n_t = 10, \tau_t = 2)$. DVEPSO ranked third for all the configurations. Its highest number of wins was recorded for the configuration $(n_t = 10, \tau_t = 5)$, and its highest number of losses was recorded for $(n_t = 1, \tau_t = 2)$. DNSGA-II ranked first for all the configurations. Its highest number of wins was recorded for the configurations $(n_t = 10, \tau_t = 5)\,(n_t = 10, \tau_t = 2)$, and its highest number of losses was recorded for $(n_t = 1, \tau_t = 4)\,(n_t = 1, \tau_t = 5)$. DVEPSO obtained a huge number of losses for all configurations.

The POFs for the four DMOOPs employed in this study are presented in Figures 5.2, 5.4, 5.6 and 5.7. Each of the POFs applies to the proposed algorithm, and a randomly selected experimental configuration.

The POFs presented in the figures apply to the first single run. A run contains four environments, which is why each of the found POFs presented by the figures in this section contains four plotted lines. Figure 5.2 presents the found POF for the FDA1

DMOOP. The true POF of FDA1 is presented in Figure 5.1. Figures 5.4, 5.6 and 5.7 present the found POFs for FDA5, DMOP2 and DMOP3 respectively. The found POF for FDA5 shows a tendency to approximate a non-convex shape. For DMOP2, the found POF shows a subtle convexity towards the origin. Similarly, for DMOP3, the found POF shows a bit of convexity towards the origin.

The true POF of FDA5 is presented in Figure 5.3, while that of DMOP2 is presented in Figure 5.5. DMOP3 and FDA1 share the same true POF, which is presented in 5.1.

Table 5.2: First Experimental Study: Overall Wins and Loses

RESULTS	2DEVENS	DVEPSO	DNSGA-II
Wins	399	295	476
Losses	377	493	300
Diff	22	-198	176
Rank	2	3	**1**

Table 5.3: First Experimental Study: Wins and Losses per Measure

PM	RESULTS	2DEVENS	DVEPSO	DNSGA-II
acc	Wins	103	97	77
acc	Losses	81	89	107
acc	Diff	22	8	-30
acc	Rank	**1**	2	3
stab	Wins	67	64	46
stab	Losses	50	56	71
stab	Diff	17	8	-25
stab	Rank	**1**	2	3
hvr	Wins	99	22	167
hvr	Losses	93	170	25
hvr	Diff	6	-148	142
hvr	Rank	2	3	**1**
react	Wins	30	76	34
react	Losses	61	22	57
react	Diff	-31	54	-23
react	Rank	3	**1**	2
NS	Wins	100	36	152
NS	Losses	92	156	40
NS	Diff	8	-120	112
NS	Rank	2	3	**1**

Table 5.4: First Experimental Study: Wins and Losses per Configuration for Accuracy Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
acc	10	4	Wins	17	14	17
acc	10	4	Losses	15	18	15
acc	10	4	Diff	2	-4	2
acc	10	4	Rank	**1**	3	**1**
acc	10	5	Wins	22	8	18
acc	10	5	Losses	10	24	14
acc	10	5	Diff	12	-16	4
acc	10	5	Rank	**1**	3	2
acc	10	2	Wins	18	14	16
acc	10	2	Losses	14	18	16
acc	10	2	Diff	4	-4	0
acc	10	2	Rank	**1**	3	2
acc	1	4	Wins	16	21	9
acc	1	4	Losses	14	11	21
acc	1	4	Diff	2	10	-12
acc	1	4	Rank	2	**1**	3
acc	1	5	Wins	14	22	6
acc	1	5	Losses	14	6	22
acc	1	5	Diff	0	16	-16
acc	1	5	Rank	2	**1**	3
acc	1	2	Wins	16	18	11
acc	1	2	Losses	14	12	19
acc	1	2	Diff	2	6	-8
acc	1	2	Rank	2	**1**	3

Table 5.5: First Experimental Study: Wins and Losses with various Frequency and Severity of Change for Stability Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
stab	10	4	Wins	13	17	9
stab	10	4	Losses	13	9	17
stab	10	4	Diff	0	8	-8
stab	10	4	Rank	2	**1**	3
stab	10	5	Wins	11	19	9
stab	10	5	Losses	15	7	17
stab	10	5	Diff	-4	12	-8
stab	10	5	Rank	2	**1**	3
stab	10	2	Wins	15	15	9
stab	10	2	Losses	11	11	17
stab	10	2	Diff	4	4	-8
stab	10	2	Rank	**1**	**1**	3
stab	1	4	Wins	10	4	7
stab	1	4	Losses	4	10	7
stab	1	4	Diff	6	-6	0
stab	1	4	Rank	**1**	3	2
stab	1	5	Wins	5	3	7
stab	1	5	Losses	3	8	4
stab	1	5	Diff	2	-5	3
stab	1	5	Rank	2	3	**1**
stab	1	2	Wins	13	6	5
stab	1	2	Losses	4	11	9
stab	1	2	Diff	9	-5	-4
stab	1	2	Rank	**1**	2	3

Table 5.6: First Experimental Study: Wins and Losses with various Frequency and Severity of Change for hvr Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
hvr	10	4	Wins	17	0	31
hvr	10	4	Losses	15	32	1
hvr	10	4	Diff	2	-32	30
hvr	10	4	Rank	2	3	**1**
hvr	10	5	Wins	17	1	30
hvr	10	5	Losses	15	31	2
hvr	10	5	Diff	2	-30	28
hvr	10	5	Rank	2	3	**1**
hvr	10	2	Wins	16	1	31
hvr	10	2	Losses	16	31	1
hvr	10	2	Diff	0	-30	30
hvr	10	2	Rank	2	3	**1**
hvr	1	4	Wins	15	9	24
hvr	1	4	Losses	17	23	8
hvr	1	4	Diff	-2	-14	16
hvr	1	4	Rank	2	3	**1**
hvr	1	5	Wins	18	4	26
hvr	1	5	Losses	14	28	6
hvr	1	5	Diff	4	-24	20
hvr	1	5	Rank	2	3	**1**
hvr	1	2	Wins	16	7	25
hvr	1	2	Losses	16	25	7
hvr	1	2	Diff	0	-18	18
hvr	1	2	Rank	2	3	**1**

Table 5.7: First Experimental Study: Wins and Losses with various Frequency and Severity of Change for react Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
react	10	4	Wins	3	11	3
react	10	4	Losses	7	2	8
react	10	4	Diff	-4	9	-5
react	10	4	Rank	2	**1**	2
react	10	5	Wins	3	9	4
react	10	5	Losses	7	3	6
react	10	5	Diff	-4	6	-2
react	10	5	Rank	3	**1**	2
react	10	2	Wins	3	8	5
react	10	2	Losses	7	3	6
react	10	2	Diff	-4	5	-1
react	10	2	Rank	3	**1**	2
react	1	4	Wins	10	19	4
react	1	4	Losses	12	4	17
react	1	4	Diff	-2	15	-13
react	1	4	Rank	2	**1**	3
react	1	5	Wins	6	14	8
react	1	5	Losses	13	4	11
react	1	5	Diff	-7	10	-3
react	1	5	Rank	3	**1**	2
react	1	2	Wins	5	15	10
react	1	2	Losses	15	6	9
react	1	2	Diff	-10	9	1
react	1	2	Rank	3	**1**	2

Table 5.8: First Experimental Study: Wins and Losses with various Frequency and Severity of Change for NS Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
NS	10	4	Wins	16	4	28
NS	10	4	Losses	16	28	4
NS	10	4	Diff	0	-24	24
NS	10	4	Rank	2	3	**1**
NS	10	5	Wins	10	9	29
NS	10	5	Losses	22	23	3
NS	10	5	Diff	-12	-14	26
NS	10	5	Rank	2	3	**1**
NS	10	2	Wins	15	4	29
NS	10	2	Losses	17	28	3
NS	10	2	Diff	-2	-24	26
NS	10	2	Rank	2	3	**1**
NS	1	4	Wins	20	8	20
NS	1	4	Losses	12	24	12
NS	1	4	Diff	8	-16	8
NS	1	4	Rank	**1**	3	**1**
NS	1	5	Wins	20	8	20
NS	1	5	Losses	12	24	12
NS	1	5	Diff	8	-16	8
NS	1	5	Rank	**1**	3	**1**
NS	1	2	Wins	19	3	26
NS	1	2	Losses	13	29	6
NS	1	2	Diff	6	-26	20
NS	1	2	Rank	2	3	**1**

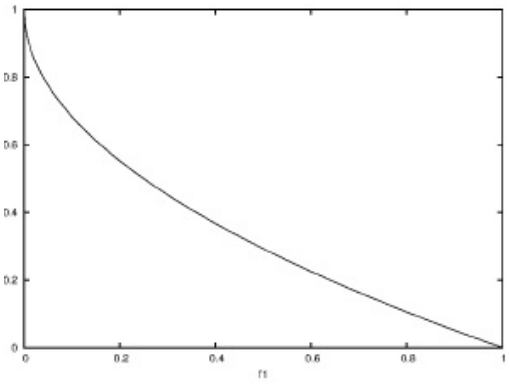


Figure 5.1: True POF of FDA1

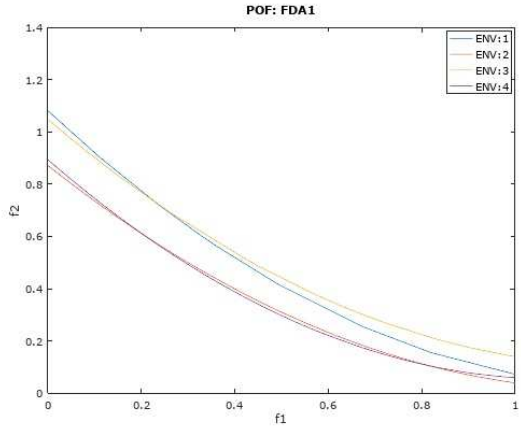


Figure 5.2: DMOA = 2DEVENS, DMOOP = FDA1, $n_t = 1\, \tau_t = 4$

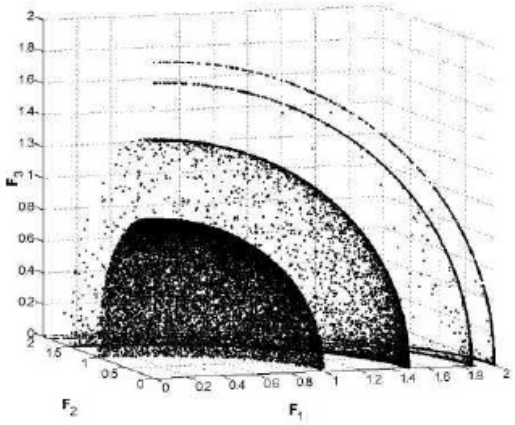


Figure 5.3: True POF of FDA5

5.4 Conclusion

This study proposed a DE algorithm, which utilizes the vector evaluation and non-dominated sorting schemes used in DVEPSO and DNSGA-II algorithms respectively, in order to search for optimal trade-off solutions of DMOOPs. The proposed algorithm shows good performance based on the results in Section 5.3. While the overall winner is DNSGA-II, the proposed algorithm ranks second. The proposed algorithm is the overall winner for the two important performance measures of accuracy and stability, therefore making it a favourite in situations where accuracy and stability measures are critical

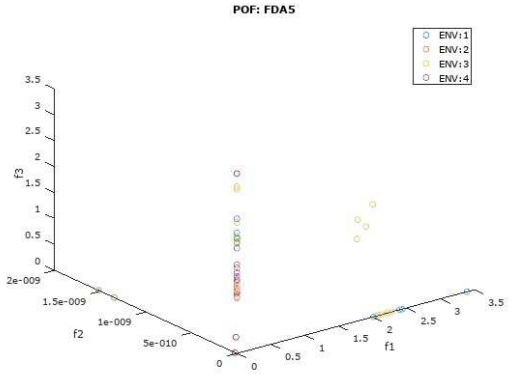


Figure 5.4: DMOA = 2DEVENS, DMOOP = FDA5, $n_t = 1\, \tau_t = 4$

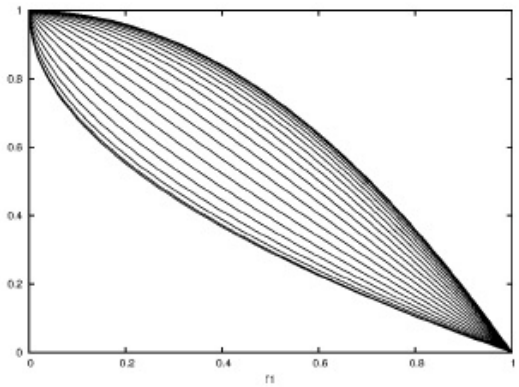


Figure 5.5: True POF of DMOP2

when choosing an algorithm for solving DMOOP.

5.5 Summary

This chapter presented the first experimental study conducted in this research. A new differential evolution algorithm, 2DEVENS, was presented. 2DEVENS was compared with DVEPSO and DNSGA-II on a selected set of DMOOPs. The algorithms were compared on a number of performance measures. Different experimental configurations were employed to measure the performance characteristics of the algorithms. Results of the study showed that the proposed algorithm, 2DEVENS, performed very well relative

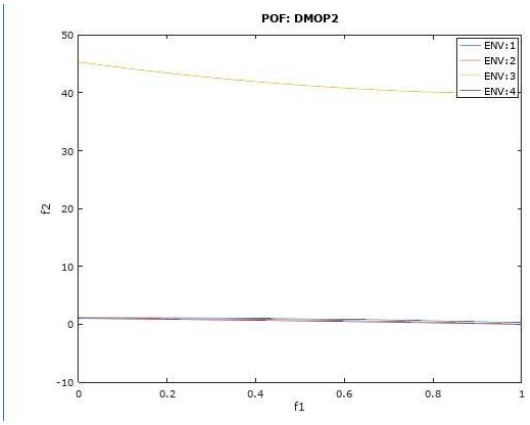


Figure 5.6: DMOA = 2DEVENS, DMOOP = DMOP2, $n_t = 1\, \tau_t = 4$

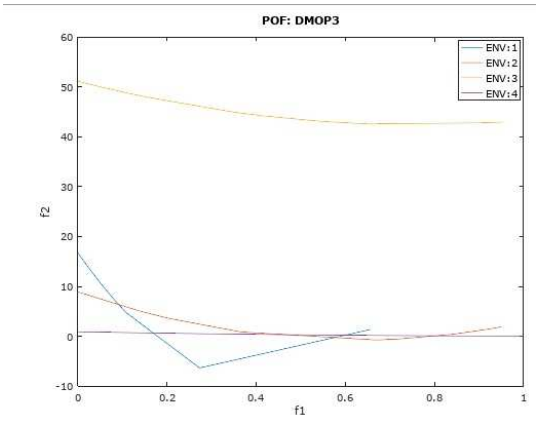


Figure 5.7: DMOA = 2DEVENS, DMOOP = DMOP3, $n_t = 1\, \tau_t = 4$

to the two standard algorithms, DVEPSO and DNSGA-II. The next chapter presents another study where the performance of 2DEVENS, DVEPSO and DNSGA-II are compared on another set of DMOOPs, under different experimental configurations.

Chapter 6

Differential Evolution Algorithm II

Chapter 5 presented an algorithm proposed to solve DMOOPs. A comparative study of the performance of the proposed algorithm and two other well established algorithms, namely DNSGA-II and DVEPSO, was conducted. All three algorithms were compared on a number of benchmark functions, which are artificial optimization problems [61] [88].

This chapter presents another study, where the three algorithms are, again, compared, using the same performance measures as the previous study that was presented in Chapter 5. However, performance measures used in this study apply to a different set of experimental configurations. Also, a new set of benchmark functions, such as the H1 and HE3, is presented in this study, in addition to some of the benchmark functions presented in Chapter 5. Section 5.1 discusses key concepts, mathematics and algorithms required to facilitate understanding of the rest of this chapter.

The remainder of the chapter is organized as follows: Section 6.1 discusses the experimental configurations, benchmark functions, performance measures and the statistical analyses employed in the study. Results of the experiment are presented and discussed in Section 6.2, and Section 6.3 draws conclusions from the results. Section 6.4 presents a summary of the chapter.

6.1 Experiment

This section presents the experimental setup employed in this study. The benchmark functions used in the study are presented in Section 6.1.1. Performance measures employed in the study are highlighted in Section 6.1.2. Section 6.1.3 discusses the algorithmic setup used in the study. Lastly, Section 6.1.4 provides an overview of the statistical analysis employed in the study.

6.1.1 Benchmark Functions

In this study, seven DMOOPs, with various τ_t-n_t combinations, were employed. The experimental configurations used for these problems are presented in Table 6.1.

In Table 6.1, the following symbols and definitions apply:

S/N: Table's row number, τ_t: frequency of change, n_t: severity of change, $c(f(x))$: number of function evaluations per iteration and $\sigma(runs)$: number of runs per configuration.

FDA5: type II DMOOP (POS and POF are dynamic); For 3 objectives, POF = $f_1^2 + f_2^2 + f_3^2 = (1 + G(t))^2$ and is non-convex; POS is $x_i = G(t)$ [61] [84].

FDA5$_{iso}$: The DMOOP is a FDA5 function (Refer to FDA5 above). In addition, the DMOOP creates a flat region in the search space, which presents a difficulty to algorithms in searching for the global optimum [84]. A transformation from FDA5 to FDA5$_{iso}$ is presented in [84].

FDA5$_{dec}$: The DMOOP is a FDA5 function (Refer to FDA5 above). In addition, this DMOOP favors convergence to a local optimum [84]. A transformation from FDA5 to FDA5$_{dec}$ is presented in [84].

dMOP2: The POF changes from convex to concave; type II DMOOP; POF = $1 - f_1^{H(t)}$; POS is $x_i = G(t)$ [73] [84].

Table 6.1: Second Experimental Study: Configuration

S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
1	FDA5	10	10	40	20	30
2	FDA5	5	10	20	20	30
3	FDA5	2	10	8	20	30
4	FDA5	10	1	40	20	30
5	FDA5	5	1	20	20	30
6	FDA5	2	1	8	20	30
7	$FDA5_{iso}$	10	10	40	20	30
8	$FDA5_{iso}$	5	10	20	20	30
9	$FDA5_{iso}$	2	10	8	20	30
10	$FDA5_{iso}$	10	1	40	20	30
11	$FDA5_{iso}$	5	1	20	20	30
12	$FDA5_{iso}$	2	1	8	20	30
13	$FDA5_{dec}$	10	10	40	20	30
14	$FDA5_{dec}$	5	10	20	20	30
15	$FDA5_{dec}$	2	10	8	20	30
16	$FDA5_{dec}$	10	1	40	20	30
17	$FDA5_{dec}$	5	1	20	20	30
18	$FDA5_{dec}$	2	1	8	20	30
19	dMOP2	10	10	40	20	30
20	dMOP2	5	10	20	20	30
21	dMOP2	2	10	8	20	30
22	dMOP2	10	1	40	20	30
23	dMOP2	5	1	20	20	30
24	dMOP2	2	1	8	20	30
25	dMOP3	10	10	40	20	30
26	dMOP3	5	10	20	20	30
27	dMOP3	2	10	8	20	30
28	dMOP3	10	1	40	20	30
29	dMOP3	5	1	20	20	30
30	dMOP3	2	1	8	20	30
31	He1	10	10	40	20	30
32	He1	5	10	20	20	30
33	He1	2	10	8	20	30
34	He1	10	1	40	20	30
35	He1	5	1	20	20	30
36	He1	2	1	8	20	30
37	He3	10	10	40	20	30
38	He3	5	10	20	20	30
39	He3	2	10	8	20	30
40	He3	10	1	40	20	30
41	He3	5	1	20	20	30
42	He3	2	1	8	20	30

dMOP3: type I DMOOP (POS is dynamic, POF is static), $POF = 1 - \sqrt{f_1}$ and is convex, POS is $x_i = G(t)$ [61] [84].

HE1: type III DMOOP (POS is static, POF is dynamic), $POF = 1 - \sqrt{f_1} - f_1 sin(10\pi t f_1)$, and the POF is characterized by dis-connected sub-regions [84].

HE3: type III DMOOP (POS is static, POF is dynamic), POS is defined by non-linear functions. POF: $(2 - \sqrt{x_1}) \left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$.

6.1.2 Performance Measures

Performance measures presented and discussed in Section 5.2.2 also apply to this study.

6.1.3 Algorithmic Setup

This study used an algorithmic setup similar to the setup discussed in Section 5.2.3.

6.1.4 Statistical Measures

The statistical analysis in Section 5.2.4 was also used in this study.

6.2 Results and Discussions

The results of the study are presented in this section. The performance of 2DEVENS was compared against two other nature-inspired algorithms, namely DVEPSO and DNSGA-II. For each table presented in this section, the winning algorithm has a rank number that is indicated in bold. Experimental configuration in this section, and in any other sections of this study, is the same as a pair of (severity of change, frequency of change) values used in the study.

In Table 6.2, the overall wins-losses is presented for each of the three algorithms. DVEPSO recorded more losses than wins, therefore it performed poorly compared to DNSGA-II and 2DEVENS. DNSGA-II ranked first, followed closely by 2DEVENS, and DVEPSO ranked last.

Table 6.3 presents wins-losses statistics for each algorithm with regards to the performance measures used in the study. 2DEVENS ranked first for the accuracy measure, acc, and DVEPSO ranked last. DVEPSO recorded the highest number of losses for acc.

For the stability measure, stab, DVEPSO ranked first, while 2DEVENS ranked second. 2DEVENS also ranked second for hvr and NS, while DNSGA-II ranked first for these two measures. 2DEVENS ranked second for the reactivity measure, react, and DNSGA-II ranked last for the measure. For all the measures, 2DEVENS never ranked last. 2DEVENS recorded its highest number of wins for NS and its highest number of losses for hvr. For DVEPSO, its highest number of wins was recorded under react while its highest number of losses was recorded under NS. DNSGA-II obtained its highest number of wins for hvr and its highest number of losses for acc.

Table 6.4 presents performance of the algorithms for acc and different experimental configurations. 2DEVENS ranked first for all the configurations, except $(n_t = 1, \tau_t = 10)$. For the configuration where 2DEVENS ranked second, it was only slightly inferior to the winner. DVEPSO ranked last for all the configurations. DNSGA-II ranked second for all the configurations, except for $(n_t = 1, \tau_t = 10)$. 2DEVENS obtained its highest number of wins for the configuration $(n_t = 10, \tau_t = 10)$, and the number of wins was the highest among all the wins recorded by all the algorithms for all the configurations. The highest number of losses recorded by 2DEVENS was 18, and that number was recorded for three configurations. 2DEVENS had a higher number of wins than losses for all the configurations. DVEPSO recorded the highest number of losses for the configuration $(n_t = 10, \tau_t = 10)$. The highest number of wins recorded by DVEPSO was for $(n_t = 1, \tau_t = 5)$. DVEPSO obtained higher number of losses than wins for all the configurations, indicating a poor performance in comparison to the other two algorithms. DNSGA-II recorded its highest number of wins for $(n_t = 1, \tau_t = 5)$ and the highest number of losses for $(n_t = 10, \tau_t = 10)$.

In Table 6.5, the algorithms' performances in relation to stab, for different experimental configurations, are presented. 2DEVENS won for the experimental configurations $(n_t = 1, \tau_t = 5)$ and $(n_t = 1, \tau_t = 10)$. The algorithm recorded its highest number of wins for $(n_t = 10, \tau_t = 5)$ and its highest number of losses for $(n_t = 10, \tau_t = 10)$. DVEPSO obtained the highest numbers of wins, four out of five, from all the configurations. DVEPSO recorded its highest number of wins for $(n_t = 10, \tau_t = 10)$ and its highest number of losses for $(n_t = 1, \tau_t = 10)$. DVEPSO ranked second for $(n_t = 1, \tau_t = 5)$, third for $(n_t = 1, \tau_t = 10)$ and first for the other configurations. DNSGA-II never ranked

first. DNSGA-II ranked second for $(n_t = 10, \tau_t = 10)$ and $(n_t = 1, \tau_t = 10)$. Its highest number of wins was recorded for $(n_t = 10, \tau_t = 10)$ and its highest number of losses was recorded for $(n_t = 10, \tau_t = 5)$.

Performances of the algorithms are presented for hvr in Table 6.6. 2DEVENS ranked second for all the experimental configurations. Its highest number of wins was recorded for the configuration $(n_t = 10, \tau_t = 10)$, and its highest number of losses was recorded for $(n_t = 1, \tau_t = 5)$. The number of wins obtained by 2DEVENS was at least equal the number of losses, except for $(n_t = 1, \tau_t = 5)$. DVEPSO ranked last for all the configurations. Its highest number of wins was recorded for $(n_t = 1, \tau_t = 5)$. Its highest number of losses was awarded for $(n_t = 10, \tau_t = 2)$. It recorded more losses than wins for all the configurations. DNSGA-II was ranked first for all the configurations. Its highest number of wins was for $(n_t = 10, \tau_t = 2)$. Its highest number of losses was recorded for $(n_t = 10, \tau_t = 10)$. It obtained more wins than losses for all the configurations, indicating a good performance.

In Table 6.7, the performances of the algorithms are presented for react. 2DEVENS ranked second for $(n_t = 10, \tau_t = 5)$, $(n_t = 1, \tau_t = 2)$, $(n_t = 1, \tau_t = 5)$ and $(n_t = 1, \tau_t = 10)$. It ranked last for the other configurations. Its highest number of wins was recorded for $(n_t = 1, \tau_t = 2)$ and $(n_t = 10, \tau_t = 5)$. Its highest number of losses was for $(n_t = 10, \tau_t = 2)$. It obtained more losses than wins for all the configurations. DVEPSO was ranked first for all configurations. Its highest number of wins was for $(n_t = 10, \tau_t = 10)$, $(n_t = 10, \tau_t = 2)$ and $(n_t = 1, \tau_t = 2)$. Its highest number of losses was recorded for $(n_t = 1, \tau_t = 5)$. It obtained more wins than losses for all configurations. DNSGA-II had its highest number of wins for $(n_t = 1, \tau_t = 5)$. Its highest number of losses was recorded for $(n_t = 1, \tau_t = 2)$. It obtained more losses than wins for all the configurations.

Table 6.8 presents the performances of the algorithms for the NS measure. 2DEVENS ranked second for all the experimental configurations. Its highest number of wins was awarded for $(n_t = 10, \tau_t = 10)$. Its highest number of losses was recorded for $(n_t = 1, \tau_t = 5)$. It obtained more wins than losses for all the configurations. DVEPSO ranked last for all the configurations. Its highest number of wins was recorded for $(n_t = 1, \tau_t = 5)$. Its highest number of losses was for $(n_t = 10, \tau_t = 2)$. It obtained more losses than

wins for all configurations, and it obtained zero number of wins for $(n_t = 10, \tau_t = 2)$. DNSGA-II was ranked first for all configurations. Its highest number of wins was for $(n_t = 10, \tau_t = 2)$. Its highest number of losses was for $(n_t = 1, \tau_t = 10)$. It obtained more wins than losses for all the configurations.

The found POFs for the five DMOOPs used in this study are presented in Figures 6.1, 6.2, 6.3, 6.5 and 6.7. Figures 6.1, 6.2 and 6.3 show non-convexity towards the origin, which is consistent with the true POF of FDA5 presented in Figure 5.3. Figure 6.5 shows discontinuities, while Figure 6.7 displays concavity towards the origin. Each of the found POFs applies to the proposed algorithm, and a randomly selected experimental configuration. The found POFs presented in this section apply to the first single run. A run contains four environments, which is why each of the POFs contains four plotted lines. The true POFs of HE1 and HE3 are presented in Figures 6.4 and 6.6. Figure 6.8 presents the found POFs of HE3 zoomed into values of f1 between 0 and 1. The figure shows that the POFs of the four environments for f1-values between 0 and 1, which were also shown in Figure 6.7, were actually not converging to the same set of values, as one may erroneously conclude from Figure 6.7. Also observed from Figure 6.8 is the fact that the environments whose POFs obtained smaller f2-values for f1-values that are greater than 1 are now showing bigger f2-values for f1-values between 0 and 1.

Table 6.2: Second Experimental Study: Overall Wins and Loses

RESULTS	2DEVENS	DVEPSO	DNSGA-II
Wins	727	415	769
Losses	536	879	496
Diff	191	-464	273
Rank	2	3	**1**

6.3 Conclusion

This study again demonstrated the ability of 2DEVENS in tracking the POFs of DMOOPs. The results of the study show that 2DEVENS was superior on the accuracy measure, which means it will be the choice algorithm when the need to accurately track the true POFs is prioritized. On other measures, such as stability, reactivity, etc., it compared very well with the other two algorithms, i.e. DVEPSO and DNSGA-II. For the stab

Table 6.3: Second Experimental Study: Wins and Losses per Measure

PM	RESULTS	2DEVENS	DVEPSO	DNSGA-II
acc	Wins	199	92	154
acc	Losses	95	204	146
acc	Diff	104	-112	8
acc	Rank	**1**	3	2
stab	Wins	88	106	58
stab	Losses	79	67	106
stab	Diff	9	39	-48
stab	Rank	2	**1**	3
hvr	Wins	182	41	253
hvr	Losses	134	275	67
hvr	Diff	48	-234	186
hvr	Rank	2	3	**1**
react	Wins	57	142	54
react	Losses	108	39	106
react	Diff	-51	103	-52
react	Rank	2	**1**	3
NS	Wins	201	34	250
NS	Losses	120	294	71
NS	Diff	81	-260	179
NS	Rank	2	3	**1**

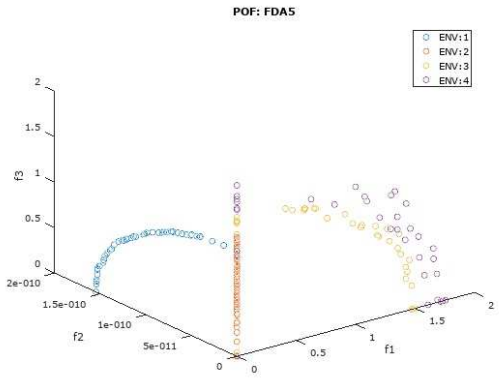


Figure 6.1: DMOA = 2DEVENS, DMOOP = FDA5, $n_t = 10\, \tau_t = 10$

measure, under the experimental configurations $(n_t = 1, \tau_t = 5)$ and $(n_t = 1, \tau_t = 10)$, it ranked first, which means that it may be preferred in some circumstances to the other algorithms, when the stab measure and specific configurations where it ranked first are prioritized.

Table 6.4: Second Experimental Study: Wins and Losses per Configuration for Accuracy Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
acc	10	10	Wins	45	13	26
acc	10	10	Losses	11	43	30
acc	10	10	Diff	34	-30	-4
acc	10	10	Rank	**1**	3	2
acc	10	5	Wins	38	14	32
acc	10	5	Losses	18	42	24
acc	10	5	Diff	20	-28	8
acc	10	5	Rank	**1**	3	2
acc	10	2	Wins	38	15	31
acc	10	2	Losses	18	41	25
acc	10	2	Diff	20	-26	6
acc	10	2	Rank	**1**	3	2
acc	1	2	Wins	32	15	20
acc	1	2	Losses	14	27	26
acc	1	2	Diff	18	-12	-6
acc	1	2	Rank	**1**	3	2
acc	1	5	Wins	24	20	22
acc	1	5	Losses	18	25	23
acc	1	5	Diff	6	-5	-1
acc	1	5	Rank	**1**	3	2
acc	1	10	Wins	22	15	23
acc	1	10	Losses	16	26	18
acc	1	10	Diff	6	-11	5
acc	1	10	Rank	2	3	**1**

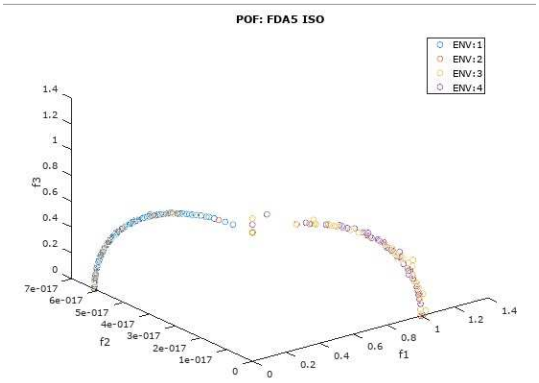


Figure 6.2: DMOA = 2DEVENS, DMOOP = FDA5$_{iso}$, $n_t = 10\,\tau_t = 10$

6.4 Summary

This chapter presented the second experimental study conducted in this research. 2DEVENS's performance was compared with DVEPSO and DNSGA-II with respect to a

Table 6.5: Second Experimental Study: Wins and Losses with various Frequency and Severity of Change for Stability Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
stab	10	10	Wins	13	27	17
stab	10	10	Losses	25	11	21
stab	10	10	Diff	-12	16	-4
stab	10	10	Rank	3	**1**	2
stab	10	5	Wins	22	24	5
stab	10	5	Losses	13	11	27
stab	10	5	Diff	9	13	-22
stab	10	5	Rank	2	**1**	3
stab	10	2	Wins	15	24	9
stab	10	2	Losses	18	9	21
stab	10	2	Diff	-3	15	-12
stab	10	2	Rank	2	**1**	3
stab	1	2	Wins	10	13	7
stab	1	2	Losses	11	8	11
stab	1	2	Diff	-1	5	-4
stab	1	2	Rank	2	**1**	3
stab	1	5	Wins	12	11	10
stab	1	5	Losses	8	12	13
stab	1	5	Diff	4	-1	-3
stab	1	5	Rank	**1**	2	3
stab	1	10	Wins	16	7	10
stab	1	10	Losses	4	16	13
stab	1	10	Diff	12	-9	-3
stab	1	10	Rank	**1**	3	2

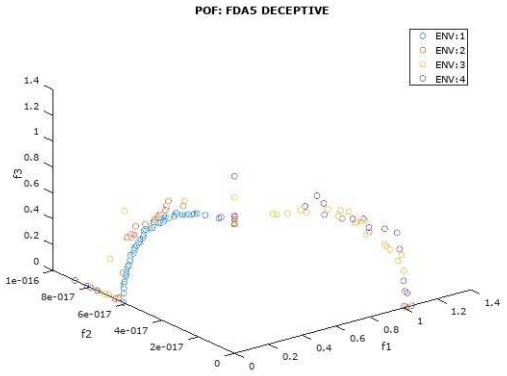


Figure 6.3: DMOA = 2DEVENS, DMOOP = FDA5$_{dec}$, $n_t = 10\,\tau_t = 10$

selected set of DMOOPs. The algorithms were compared on a number of performance measures. Different experimental configurations were used to measure the performance characteristics of the algorithms. Results of the study shows that the proposed algo-

Table 6.6: Second Experimental Study: Wins and Losses with various Frequency and Severity of Change for hvr Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
hvr	10	10	Wins	38	4	42
hvr	10	10	Losses	18	52	14
hvr	10	10	Diff	20	-48	28
hvr	10	10	Rank	2	3	**1**
hvr	10	5	Wins	34	3	47
hvr	10	5	Losses	22	53	9
hvr	10	5	Diff	12	-50	38
hvr	10	5	Rank	2	3	**1**
hvr	10	2	Wins	34	2	48
hvr	10	2	Losses	22	54	8
hvr	10	2	Diff	12	-52	40
hvr	10	2	Rank	2	3	**1**
hvr	1	2	Wins	26	8	42
hvr	1	2	Losses	26	40	10
hvr	1	2	Diff	0	-32	32
hvr	1	2	Rank	2	3	**1**
hvr	1	5	Wins	26	15	43
hvr	1	5	Losses	30	41	13
hvr	1	5	Diff	-4	-26	30
hvr	1	5	Rank	2	3	**1**
hvr	1	10	Wins	24	9	31
hvr	1	10	Losses	16	35	13
hvr	1	10	Diff	8	-26	18
hvr	1	10	Rank	2	3	**1**

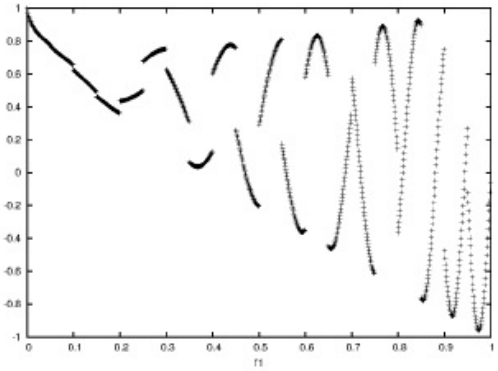


Figure 6.4: True POF of HE1

rithm, 2DEVENS, performed very well relative to the other standard DMOOPs. The next chapter presents another study, where 2DEVENS is used in solving problems that employ decision-maker preferences in the process of finding optimal solutions to opti-

Table 6.7: Second Experimental Study: Wins and Losses with various Frequency and Severity of Change for react Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
react	10	10	Wins	4	26	9
react	10	10	Losses	21	4	14
react	10	10	Diff	-17	22	-5
react	10	10	Rank	3	**1**	2
react	10	5	Wins	10	24	6
react	10	5	Losses	14	8	18
react	10	5	Diff	-4	16	-12
react	10	5	Rank	2	**1**	3
react	10	2	Wins	4	26	11
react	10	2	Losses	22	6	13
react	10	2	Diff	-18	20	-2
react	10	2	Rank	3	**1**	2
react	1	2	Wins	14	26	5
react	1	2	Losses	15	4	26
react	1	2	Diff	-1	22	-21
react	1	2	Rank	2	**1**	3
react	1	5	Wins	14	20	13
react	1	5	Losses	19	10	18
react	1	5	Diff	-5	10	-5
react	1	5	Rank	2	**1**	3
react	1	10	Wins	11	20	10
react	1	10	Losses	17	7	17
react	1	10	Diff	-6	13	-7
react	1	10	Rank	2	**1**	3

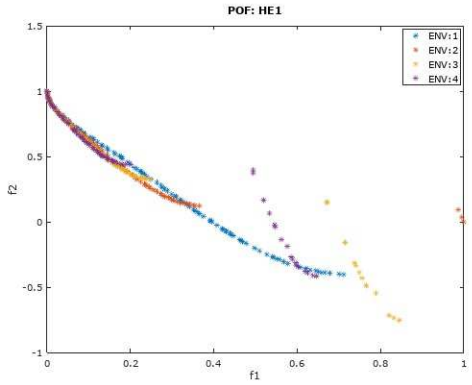


Figure 6.5: DMOA = 2DEVENS, DMOOP = HE1, $n_t = 10\,\tau_t = 10$

mization problems.

Table 6.8: Second Experimental Study: Wins and Losses with various Frequency and Severity of Change for NS Measure

PM	n_t	τ_t	RESULTS	2DEVENS	DVEPSO	DNSGA-II
NS	10	10	Wins	38	4	39
NS	10	10	Losses	15	52	14
NS	10	10	Diff	23	-48	25
NS	10	10	Rank	2	3	**1**
NS	10	5	Wins	34	5	43
NS	10	5	Losses	20	51	11
NS	10	5	Diff	14	-46	32
NS	10	5	Rank	2	3	**1**
NS	10	2	Wins	35	0	49
NS	10	2	Losses	21	56	7
NS	10	2	Diff	14	-56	42
NS	10	2	Rank	2	3	**1**
NS	1	2	Wins	33	5	46
NS	1	2	Losses	23	51	10
NS	1	2	Diff	10	-46	36
NS	1	2	Rank	2	3	**1**
NS	1	5	Wins	31	11	42
NS	1	5	Losses	25	45	14
NS	1	5	Diff	6	-34	28
NS	1	5	Rank	2	3	**1**
NS	1	10	Wins	30	9	31
NS	1	10	Losses	16	39	15
NS	1	10	Diff	14	-30	16
NS	1	10	Rank	2	3	**1**

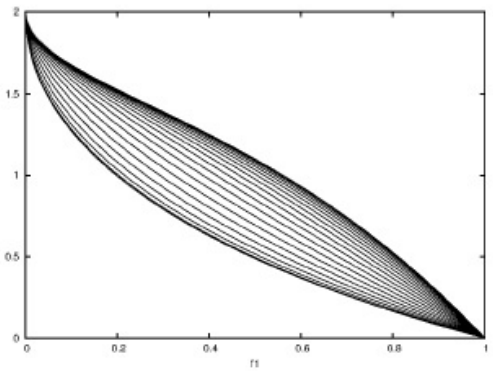


Figure 6.6: True POF of HE3

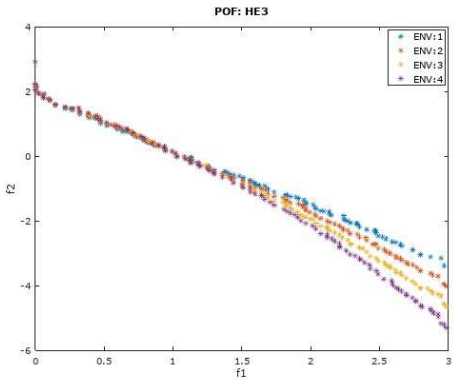


Figure 6.7: DMOA = 2DEVENS, DMOOP = HE3, $n_t = 10\, \tau_t = 10$

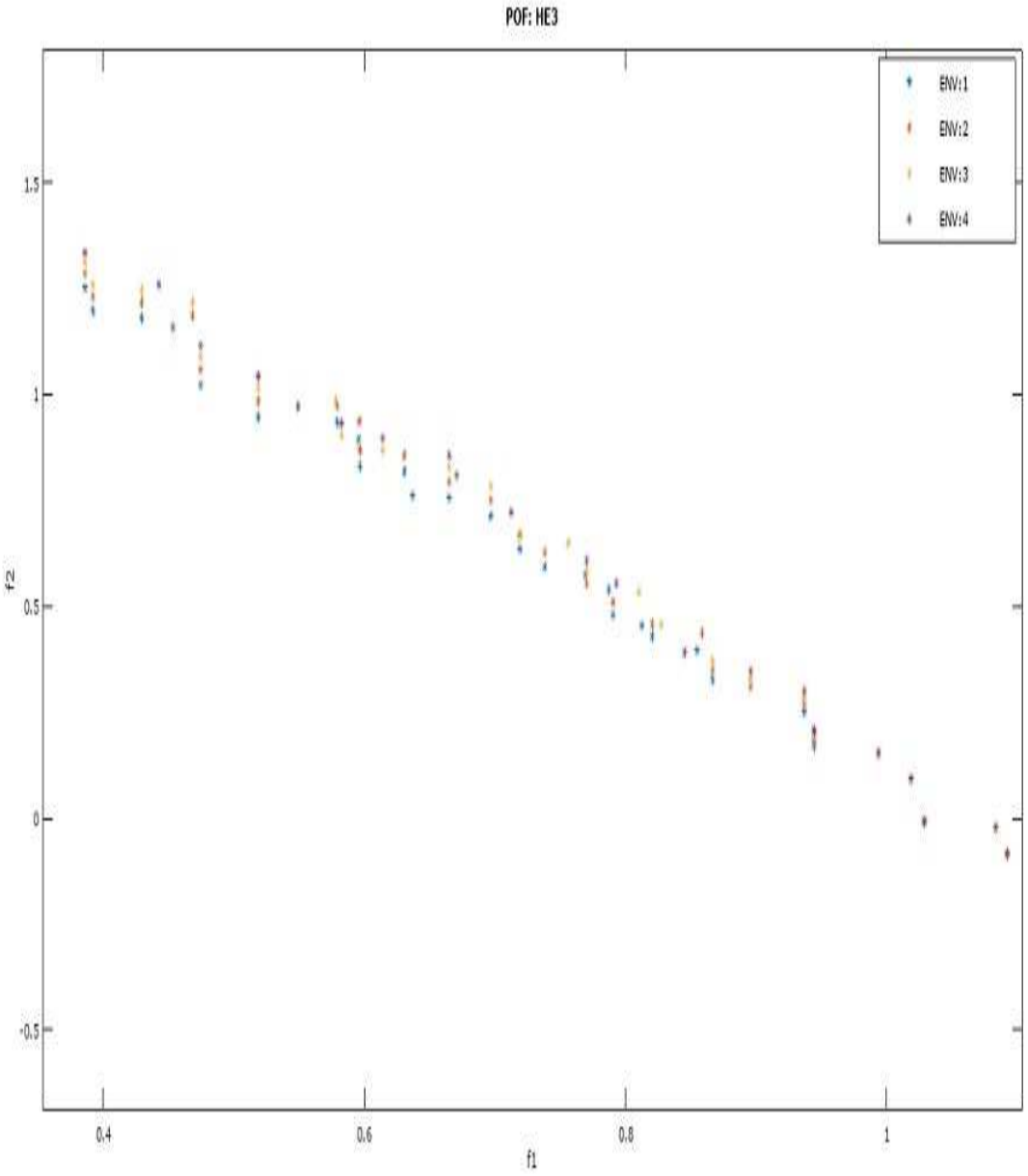


Figure 6.8: DMOA = 2DEVENS, DMOOP = HE3, $n_t = 10\, \tau_t = 10$

Chapter 7

Decision-maker's Preference Driven Optimization Problem

Many DMOOPs occur in the real-world [9] [26] [35] [47] [61] [78] [137] [96] [116] [133] [139] [169], while their artificial counterparts are well researched in [61] [73] [84] [88] [97] [106]. DMOOPs have many goals or objectives, and elements of the problems, such as objectives and/or constraints, change over time [10] [87] [105] [134]. However, these goals are usually in conflict in relation to one another, thereby making the process of finding a single optimal solution a very difficult task [33]. Trade-off solutions are therefore the norm. The Pareto-dominance relation is a popular operator used to compare the trade-off solutions [140].

The set of optimal trade-off solutions of a DMOOP in the decision variable space, also called solution space, is called the POS, while in the objective space it is called the POF [48]. Sometimes, the set of trade-off solutions may be large in number and a subset may be required by a decision-maker in the hope of finding a subset that better reflects the preferences of the decision-maker [98]. Some research has been conducted on methods of incorporating a decision-maker's preferences when solving MOOPs in static environments [32] [36] [37] [47] [62] [76] [101] [162] [181] [185]. Apriori, interactive and posteriori are the dominant classification of the methods. It is noteworthy to state that none of these preference incorporation methods have been applied to DMOOPs. Therefore, this chapter presents an experimental study, which aims at a preference incorporation

method that is adapted for DMOOPs. The method is partly a priori and interactive. A procedure, named bootstrap, simulates the a priori incorporation of decision-maker preferences, while the interactive incorporation of preferences is employed whenever there is a change in the dynamic environment in such a way that the decision-maker preference set may be significantly affected.

Introducing decision-maker preferences, however leads to the reformulation of DMOOPs as constrained problems, which are then solved by the proposed algorithms using a variation of penalty functions in [39] [102] [129] [130] [213]. The constraints imposed on DMOOPs as a result of a decision-maker's preferences are defined in the objective space, and such constraints partition the objective space into feasible and infeasible regions.

In this study, a solution will henceforth be referred to as a decision.

This study proposes a bounding box approach (refer to Equation (7.2)) to incorporate a decision-maker's preferences into a DMOO process. The proposed bounding box, unlike the proposal in [8], is used in the context of DMOOPs, thus making it the first of its kind. Secondly, the study proposes new approaches that can drive the DMOO search, which are constrained by a decision-maker's preferences. Thirdly, the study motivates a comparative analysis of the performance of three proposed algorithms (ALG:1, ALG:2, ALG:3) incorporating these approaches. The three proposed algorithms, however, differ by how they treat solutions that violate decision-maker's preferences. ALG:1, also referred to as the Proportionate-Penalty, penalizes solutions by the degree of their violations of decision-maker's preferences. ALG:2, also referred to as the Death-Penalty, logically kills solutions that violate decision-maker's preferences. ALG:3, also referred to as Restrict-Search-to-Feasible-Region, keeps the search for the optimal solutions in the region where only non-violating solutions are found. Solutions used by ALG:3 in the process of searching for the optimal solutions do not violate decision-maker's preferences. The algorithms are implemented using an hybrid form of differential evolution in [42], which combines non-dominated sorting [44] with vector-evaluation schemes in selecting target vectors and the vectors that survive to the next generation during the optimization process. Performance of the proposed algorithms are measured using selected measures in [84] [87] [89] and the new measures proposed in this study. The three proposed algorithms are different from one another in terms of how they penalize solutions that violate

a decision-maker's preferences.

The rest of this chapter is organized as follows: Section 7.1 provides background to the experimental study conducted in this chapter. The experiments that underlie this chapter are described in Section 7.2. Results and their discussion are presented in Section 7.3. Conclusions of the experiments are presented in Section 7.4. Section 7.5 summarizes the chapter.

7.1 Background

This section discusses the key mathematics and algorithmic setup, which underly some of the proposals in this study. Section 7.1.1 discusses the mathematical formulation of the DMOOPs that are used in this study. It also discusses the mathematics of the proposed bounding box approach and the limiting behaviors of the penalty functions used in the study. Section 7.1.2 discusses the mathematics used by one of the proposed measures in computing the deviation of the violating decisions, while Section 7.1.3 presents an algorithm that computes the spread of non-violating decisions that are found in the bounding box. Section 7.1.4 discusses the core evolutionary algorithm on which the proposed DMOAs are based.

7.1.1 The Bounding Box Mathematics

Let a composite function F be defined as follows:

$$F : \Omega_x \times \Omega_t \longrightarrow O \tag{7.1}$$

where
$\Omega_x(decision\ space) = \Re^n$, $n \geq 2$, $\Omega_t \subseteq R$ is the time space, $t \in \Omega_t$ is a real-valued time instance and $t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor$, n_t is the severity of change, τ is the iteration counter, τ_t is the frequency of change.

$$O(objective\ space) = \begin{cases} \Re^2 & \text{(e.g. FDA1 and dMOP2)} \\ \Re^3 & \text{(e.g. FDA5)} \end{cases}$$

And a decomposition of F hereby follows:

$$F(x,t) = \begin{cases} (f1, f2) & \text{(e.g. FDA1 and dMOP2)} \\ (f1, f2, f3) & \text{(e.g. FDA5)} \end{cases}$$

Each f_i is defined as:

$$f_i : \begin{cases} \Omega_x \times \Omega_t \longrightarrow R & \text{(e.g. FDA1, dMOP2, FDA5)}, \forall i = 1, 2, 3 \end{cases}$$

Let a decision-maker's preference set be defined as follows:

$$Box(z; p) = \{z \in O | d(z, p) \leq r, p \in O, r \in \Re\} \tag{7.2}$$

where

d is the Euclidean distance measure, p is the center of the box formed by the points in this set, $Box(z; p)$, r is the radius of the box, O is as defined in Equation (7.1), and p and r are interactively chosen by the decision-maker.

And let a penalty function and its limiting behaviours be defined as follows:

$$penalty(z_i \in O, \lambda) = \begin{cases} 0 & if \quad d(z_i, p) \leq r \\ \lambda(d(z_i, p) - r) & if \quad d(z_i, p) > r \end{cases} \tag{7.3}$$

$$\lim_{\lambda \to c} penalty(z_i \in O, \lambda) = \begin{cases} 0 & if \quad d(z_i, p) \leq r \\ c(d(z_i, p) - r) & if \quad d(z_i, p) > r \end{cases} \tag{7.4}$$

$$\lim_{\lambda \to realmax} penalty(z_i \in O, \lambda) = \begin{cases} 0 & if \quad d(z_i, p) \leq r \\ realmax & if \quad d(z_i, p) > r \end{cases} \tag{7.5}$$

$$\lim_{\lambda \to realmax} z_i + penalty(z_i \in O, \lambda) = \begin{cases} z_i & if \quad d(z_i, p) \leq r \\ I_1 * realmax & if \quad d(z_i, p) > r \end{cases} \tag{7.6}$$

$$\lim_{\lambda \to c} z_i + penalty(z_i \in O, \lambda) = \begin{cases} z_i & if \quad d(z_i, p) \leq r \\ z_i + I_1 * c(d(z_i, p) - r) & if \quad d(z_i, p) > r \end{cases} \tag{7.7}$$

where

$\lambda(>= 0)$ is a penalty control parameter whose value is determined by each algorithm,

and p, r and d are as defined in Equation (7.2).

Then a penalized outcome $z_i^* \in O$ is defined as:
$z_i^* = z_i + I_1 * penalty(z_i, \lambda)$
where
z_i is the un-penalized outcome in the objective space, $z_i = F(x_i, t), x_i \in \Omega_x$,
F is as defined in Equation (7.1), and I_1 is an all-ones vector in the objective space (i.e. (1,1)$\in \Re^2$).

7.1.2 Deviation of Violating Decisions

Solution space vectors whose objective values are outside the preference set are called violating decisions, because such solutions violate decision-maker preferences. Depending on the control parameters used in the implementation of the penalty function of the proposed algorithms, occasionally the violating decisions may find their way into the archive, especially in situations where all the non-dominated solutions violate decision-maker preferences and the non-violating decisions are not found. It is a rare scenario, however, because the non-violating decisions, if they are found in the archive, are very likely to dominate, in Pareto-dominance sense, the violating and penalized decisions. However, when such violating decisions find their way into the archive, a measure of the proximity of these violating decisions to the preferred bounding box is required. The smaller the total proximity, the better the violating decisions are. This section presents the mathematics underlying the computation of the total proximity/deviation of the violating decisions.

Let p, r and the distance measure $d(z, p)$ be as defined in Section 7.1.1.
Let a set of violating decisions, Z, be defined as follows:

$$Z = \{z_i \in O | d(z, p) > r\} \tag{7.8}$$

Let the cardinality, N, of Z be defined as follows:

$$N = |Z| \tag{7.9}$$

Let the deviation of $z_i \in Z$ be defined as follows:

$$d_i = d(z_i, p) - r \, (d_i > 0) \tag{7.10}$$

The total deviation of all elements in Z is therefore:

$$dVD = \sqrt{\frac{\sum(1 + d_i)^2}{N}} \tag{7.11}$$

7.1.3 Spread of Non-violating decisions

This is one of the four measures proposed in this study. This measure estimates how well spread out the preferred decisions are within the bounding box located in the objective space. The greater the value of this measure, the better the performance of the algorithm. The computational procedure used in computing this measure is presented in Algorithm 10.

7.1.4 Core Evolutionary Algorithms

The core evolutionary algorithm underlying the proposals of this study is presented in Algorithm 11.

7.2 Experiment

This section discusses the experimental setup used in the study. Section 7.2.1 discusses the various algorithms incorporating the decision-maker's preference approach. The decision-maker preferences are discussed in Section 7.2.2. Section 7.2.3 discusses benchmark functions used in this study. Performance measures are presented in Section 7.2.4, while Section 7.2.5 discusses the statistical analysis approach used in the study.

7.2.1 Algorithmic Setup

1. **Apriori Preference Incorporation**: A single run of the DMOA simulation is executed. A series of POFs at every environment change is presented to the decision-maker. The decision-maker chooses one of these POFs, and then selects one of

Algorithm 10 Spread Estimation

```
Procedure SpreadEstimator(outcomes)
Get count of outcomes, N ← count(outcomes)
if N ≤ 1 then
    return 0
end if
if N == 2 then
    return norm_2(outcomes(2) − outcomes(1))
end if
dtot ← 0
Get a node, firstNode ← outcomes(1)
Set current node, currentNode ← firstNode
loop:
if unProcessedNodes(outcomes) > 1 then
    MarkNodeAsProcessed(currentNode)
    nearestNode=getNearestNode(currentNode,outcomes)
    dist = norm_2(currentNode − nearestNode)
    dtot ← dtot + dist
    currentNode ← nearestNode
    goto loop
end if
dist = norm_2(currentNode − firstNode)
dtot ← dtot + dist
return dtot
```

the points on the chosen POF, which will become his x_p and p, the preferred decision vector and the outcome respectively. This preference, together with the radius of the bounding box to be specified by the decision-maker during the bootstrap procedure, will be used to drive the various DMOA simulations that optimize the DMOOP under the constraints of decision-maker preferences. Algorithm 12 presents the computational details of the bootstrap procedure.

2. **Interactive Preference Incorporation**: A significant change in the environment

Algorithm 11 Dynamic Multi-Objective Optimization

Procedure DMOA $(freq, severity, maxiteration, dMOOP)$
Set population size, N
Set archive max size, SizeArchive
Initialize the iteration counter, $iteration \leftarrow 0$
Initialize time, $t \leftarrow 0$
Initialize$(P_t, freq, severity, dMOOP, t)$
{Initialize population of solutions, P_t}
AssignNonDominatedToArchive$(P_t, dMOOP, t)$
loop:
if $iteration \leq maxiteration$ **then**
  $t \leftarrow 1/severity * floor(iteration/freq)$
  Optimizer$(P_t, dMOOP, t)$
  Pick sentry solutions
  if ENV changes$(P_t, dMOOP, t)$ **then**
    ProcessChange$(P_t, freq, severity, dMOOP, t)$
  end if
  $iteration \leftarrow iteration + 1$
  goto *loop*
end if

Algorithm 12 Apriori Preference Incorporation

Procedure BootStrap (freq, severity, iteration, F)
Call $DMOA(freq, severity, iteration, F)$
{DMOA returns $\{POS_t^k, k = 1, ..., n\}$ }
{k : k^{th} environment change }
$i \leftarrow DMChooseIn(1...n)$
$x_p \leftarrow DMChooseIn(POS_t^i)$
$p \leftarrow F(x_p, t)$
Decision-maker chooses box radius, $r \leftarrow random()$
return $(x_p : p : r)$

Algorithm 13 Interactive Incorporation of Preferences

1: Procedure RepositionBoundingBox(F, x_p, p, r, t)
{F: multi-objective function to be evaluated}
{x_p: Decision-maker's preferred decision vector}
{p: Decision-maker's preferred outcome as defined in Eq (7.2), page 113}
{r: Decision-maker's preferred box radius}
{ARCHIVE: POS_t}
2: $POF_t \leftarrow F(POS_t, t)$
3: **if** $p \in POF_t$ **then**
4: *return* $(x_p : p : r)$
5: **end if**
6: **if** $F(x_p, t) \in POF_t$ **then**
7: *Reposition center of box,* $p \leftarrow F(x_p, t)$
8: *return* $(x_p : p : r)$
9: **end if**
{Decision-maker picks a new position for x_p and p}
10: $x_p \leftarrow DMChooseIn(POS_t)$
11: *Reposition center of box,* $p \leftarrow F(x_p, t)$
12: *return* $(x_p : p : r)$

may occur, and the resulting POF may shift in such a way that the earlier decision-maker preference, p, is no longer part of the new POF. In this scenario, the decision-maker will be expected to interactively redefine the position of the bounding box, so as to have its preference lie on the new POF. A few scenarios may result when this shifting POF occurs. The initially preferred outcome, p, may no longer lie on the new POF, but the functional value of the corresponding decision variable x_p may still lie on the new POF. The second possibility is that both p and the functional value of x_p do not lie on the new POF. In these cases, a new bounding box position needs to be defined. To simulate the interactive redefinition of the bounding box position, Algorithm 13 presents a proposed computational procedure.

3. **Proportionate-Penalty**: This approach, and the next two that follow immediately in this section, use a penalty function (refer to Equation (7.3)) to penal-

Algorithm 14 Proportionate Penalty

1: Procedure FuncEvaluate (F, x, t)
{F: multi-objective function to be evaluated}
{x: decision vector}
{p: as defined in Eq (7.2), page 113}
{r: as defined in Eq (7.2), page 113}
{λ: as defined in Eq (7.2), page 113}
{λ: a random number between 100 and 1000}
{I_1: as defined in Eq (7.2), page 113}
2: *Compute objective value of x,* $z \leftarrow F(x,t)$
3: *Compute violation of z,* $d \leftarrow norm_2(z-p) - r$
4: **if** $d \leq 0$ **then**
5: *return* z {x:z is a non-violating decision}
6: **end if**
{x:z is a preference violating decision}
{proceed to penalize for violation}
7: *compute penalty,* $penalty \leftarrow \lambda * d$
8: *vectorize penalty,* $penalty \leftarrow I_1 * penalty$
9: *penalize objective value,* $z \leftarrow z + penalty$
10: *return* z

ize violating decisions which do not satisfy the decision-maker's preferences. The penalty proposed is however, proportional to the violation. Violating decisions are penalized during function evaluation. Algorithm 14 presents function evaluations employed by this approach.

4. **Death-Penalty**: Maximum/death penalty is imposed on violating decisions during the function evaluation. Some penalty, which is death, is administered on a decision irrespective of the magnitude of violation of that decision. With maximum penalty, it becomes very unlikely that violating decisions will find their way into the archive, because they will be dominated by non-violating decisions. This is how violating decisions are computationally eliminated during the search process, and the optimization process is driven towards a region of the search space

Algorithm 15 Death Penalty

1: Procedure FuncEvaluate (F, x, t)
   {F: multi-objective function to be evaluated}
   {x: decision vector}
   {p: as defined in Eq (7.2), page 113}
   {r: as defined in Eq (7.2), page 113}
   {I_1: as defined in Eq (7.2), page 113}
   {realmax: maximum real value on a machine}
2: *Compute objective value of x,* $z \leftarrow F(x,t)$
3: *Compute violation of z,* $d \leftarrow norm_2(z-p) - r$
4: **if** $d \leq 0$ **then**
5: *return* z {x:z is a non-violating decision}
6: **end if**
   {x:z is a preference violating decision}
   {proceed to penalize for violation}
7: *compute penalty,* $penalty \leftarrow realmax$
8: *vectorize penalty,* $penalty \leftarrow I_1 * penalty$
9: *impose death/max penalty,* $z \leftarrow penalty$
10: *return* z

dominated by non-violating decisions. Algorithm 15 presents how this approach performs function evaluation.

5. **Restrict-Search-to-Feasible-Region** Feasibility is preserved by starting the search within the preferred bounding box, and using the death penalty to prevent preference violating decisions from entering the archive. This approach restricts the search to the feasible region, unlike [138], and it improves the explorative capability of this algorithm. Preferred decisions kick off the search during initialization of the population of decisions. A pool of preferred decisions is aggregated with the decision-maker preference and the current decisions in the archive. Then, a loop is performed, where nearly-identical clones of the pool of preferred decisions are created using polynomial mutation [79]. These new clones constitute a new population from where search will start. A selection of the non-dominated decisions

Algorithm 16 Restrict Search to Feasible Region

```
Procedure Initialize (P_t, freq, severity, F, t)
  {x_p: Decision-maker's preferred decision vector)}
  {ARCHIVE: POS}
  {F: multi-objective function to be evaluated}
  {N: population size, fixed for this study}
Pooled preferences, pool ← [x_p; ARCHIVE]
Initialize counter, i ← 1
loop:
if i ≤ N then
   iNumberAttempts ← 1
   MoreAttempts:
   if iNumberAttempts > 100 then
      goto exitMoreAttempts
   end if
   solution ← randomlyChooseIn(pool)
   solution ← polynomial_mutate(solution)
   if isPreferredDecision(solution, F, t) then
      goto exitMoreAttempts
   end if
   iNumberAttempts ← iNumberAttempts + 1
   goto MoreAttempts
   exitMoreAttempts:
   addSolutionToPopulation(P_t, solution)
   i ← i + 1
   goto loop
end if
AssignNonDominatedToArchive(P_t, F, t)
```

in the new population are added to archive. Algorithm 16 presents the procedural details.

6. **Differential Evolution Algorithm: Control Parameters**

(a) The Algorithm (refer to Algorithm 9) is characterized as DE/best/1/bin.

(b) Trial vector: To generate a trial vector from a parent vector during the mutation phase of the algorithm, the best vector in the adjacent hypercube or sub-population of the parent vector is chosen as the target vector. The number of hypercubes employed by the algorithm is the same as the number of objective functions in the underlying DMOOP.

(c) One difference vector: two randomly chosen vectors from the parent vector's hypercube are used to form a difference vector.

(d) Crossover: Binary crossover [58] is used by the algorithm, because of its viability as a crossover method in DE algorithms.

(e) Scaling factor, $\beta \in (0.4, 1)$.

(f) Recombination probability, $p_r = 0.8$.

(g) DE convergence is insensitive to the control parameters [57]. Therefore, the algorithm randomly chooses the scaling factor and fixes the recombination probability as defined above.

7.2.2 Decision Maker's Preferences

The decision-maker preferences are correspondingly associated, serially, with each of the eighteen experimental configurations presented in Section 7.2.3. For instance, the first experimental preference in Table 7.1 is associated with the first experimental preference in Table 7.2, while both are associated with the first experimental configuration in Table 7.3.

7.2.3 Benchmark Functions

Three DMOOPs, with various τ_t-n_t combinations, were used in this study. The experimental configurations used for these DMOOPs are as follows:

The following symbols were used in Table 7.3:
τ_t: frequency of change.

Table 7.1: Third Experimental Study: Experimental Preferences - Decision Space

S/N	x_1	x_2	x_3	x_4	x_5	x_6
1	0.476	5.3e-01	0.5877	0.59104	0.5324	0.4989
2	0.477	3.3e-01	0.1630	0.42817	0.3250	0.2654
3	0.476	1.9e-01	0.0912	0.17928	0.1616	0.2331
4	0.816	9.5e-01	0.8768	0.86623	0.4892	0.7924
5	0.761	-1.0e-01	0.1513	-0.17267	0.0387	0.1229
6	0.167	1.8e-01	0.0032	-0.00311	0.1462	-0.0284
7	0.964	3.5e-06	0.5378	0.51880	0.3188	0.5433
8	0.000	2.8e-05	0.3660	0.28965	0.4592	0.3918
9	1.000	3.0e-04	0.3992	0.39925	0.5292	0.4621
10	0.357	1.0e+00	0.5083	0.74667	0.8383	0.7472
11	0.000	9.1e-04	0.7330	0.49810	0.8207	0.6616
12	0.000	8.0e-02	0.6419	0.87802	0.8485	0.8020
13	0.146	3.1e-01	0.2970	0.30878	0.3052	0.3065
14	0.734	1.6e-01	0.1657	0.13508	0.1101	0.1729
15	0.317	3.1e-01	0.3230	0.32197	0.3290	0.2881
16	0.061	4.6e-03	0.0027	0.00225	0.0032	0.0079
17	1.000	4.4e-02	0.0391	0.10593	0.0204	0.0658
18	0.000	2.8e-03	0.0015	0.00042	0.0016	0.0045

n_t: severity of change.

$c(f(x))$: count of function evaluations per iteration.

$\sigma(runs)$: number of runs per configuration.

$FDA1$: type I DMOOP (POS is dynamic, POF is static), POF $= 1 - \sqrt{f_1}$ and convex, POS is $x_i = G(t)$ [61] [84].

$FDA5$: type II DMOOP (POS and POF are dynamic); For 3 objectives, POF $= f_1^2 + f_2^2 + f_3^2 = (1 + G(t))^2$ and non-convex; POS is $x_i = G(t)$ [61] [84].

$dMOP2$: POF changes from convex to concave; type II DMOOP; POF $= 1 - f_1^{H(t)}$; POS is $x_i = G(t)$ [73] [84].

Table 7.2: Third Experimental Study: Experimental Preferences - Objective Space

S/N	f_1	f_2	f_3
1	4.8e-01	3.2e-01	N/A
2	4.8e-01	6.2e-01	N/A
3	4.8e-01	9.3e-01	N/A
4	8.2e-01	2.4e+00	N/A
5	7.6e-01	1.7e-01	N/A
6	1.7e-01	6.3e-01	N/A
7	9.3e-01	4.5e-71	1.3843
8	1.8e+00	2.9e-59	0
9	1.0e-16	4.5e-62	1.6781
10	2.8e-08	2.8e-08	1.6354
11	2.9e+00	4.1e-03	0
12	3.5e+00	4.4e-01	0
13	1.5e-01	4.6e+00	N/A
14	7.3e-01	9.5e+00	N/A
15	3.2e-01	4.3e+00	N/A
16	6.1e-02	9.7e-01	N/A
17	1.0e+00	2.1e-01	N/A
18	0.0e+00	1.0e+00	N/A

7.2.4 Performance Measures

Each of the measures were computed immediately before a change in environment occurred. This was done for thirty runs. An average of the values of the thirty runs was then computed for each measure in each environment. The performance measures are as follows:

- Accuracy Measure (acc): measures how accurate a DMOA is able to approximate the true POF of a DMOOP [84] [87] [89]. The lower the value of acc, the better the algorithm's performance.
- Stability Measure (stab): quantifies the effect of environment changes on the accuracy measure [23] [84] [87] [89]. The lower the value of this measure, the better.

Table 7.3: Third Experimental Study: Configuration

S/N	DMOOP	τ_t	n_t	Iterations	c(f(x))	$\sigma(runs)$
1	FDA1	4	10	16	20	30
2	FDA1	5	10	20	20	30
3	FDA1	2	10	8	20	30
4	FDA1	4	1	16	20	30
5	FDA1	5	1	20	20	30
6	FDA1	2	1	8	20	30
S/N	**DMOOP**	τ_t	n_t	**Iterations**	**c(f(x))**	$\sigma(runs)$
7	FDA5	4	10	16	20	30
8	FDA5	5	10	20	20	30
9	FDA5	2	10	8	20	30
10	FDA5	4	1	16	20	30
11	FDA5	5	1	20	20	30
12	FDA5	2	1	8	20	30
S/N	**DMOOP**	τ_t	n_t	**Iterations**	**c(f(x))**	$\sigma(runs)$
13	dMOP2	4	10	16	20	30
14	dMOP2	5	10	20	20	30
15	dMOP2	2	10	8	20	30
16	dMOP2	4	1	16	20	30
17	dMOP2	5	1	20	20	30
18	dMOP2	2	1	8	20	30

- Hypervolume Ratio (hvr): measures the proportion of the objective space that is covered by a non-dominated set without suffering from the bias of convex a region as seen in the HyperVolume measure [90]. hvr was proposed in [192]. The higher the value of this measure, the better.

- Reactivity Measure (react): measures how long it takes a DMOA to recover after a change in the environment occurred. The length of time it takes to reach a specified accuracy threshold is employed in computing this measure [84]. It was originally proposed in [176]. The lower the value of this measure, the better.

The next group of measures are new measures proposed in this dissertation:

- Number of Non-Violating Decisions (nNVD): measures the number of decisions that fall within the decision-maker's preference set. The higher the value of this measure, the better.
- Spread of Non-Violating Decisions (sNVD): measures the spread of decisions within the preferred set. The higher the value of this measure, the better the performance of the algorithm.
- Number of Violating Decisions (nVD): measures the number of violating decisions in the archive. These are decisions that do not lie within the preferred set. The lower the value of this measure, the better the performance of the algorithm.
- Total Deviation of Violating Decisions (dVD): measures the total deviation from the preferred set for all violating decisions in the archive. The lower the value of this measure, the better.

The four new performance measures proposed in this study specifically measure the performance of a DMOA with regards to decision-maker preference constraints, and thus facilitate comparative analysis of DMOAs in the context of a decision-maker's preferences.

7.2.5 Statistical Analysis

A statistical analysis of the performance measures studied in the study was done in accordance with the $wins - losses_B$ algorithm proposed in [87] (refer to Section 5.2.4). The algorithm was implemented in R [149] and the Kruskal-Wallis and Mann Whitney U statistical functions in R were used as stipulated in [87].

7.3 Results and Discussion

This section presents a summary of the results of the proposed algorithms for various performance measures and experimental configurations. Detailed results are however presented in Appendix D.

The summarized results are presented in Tables 7.4, 7.5 and 7.6.

For all the tables in this section, any column with a bold entry signifies the winning algorithm for the particular measure of performance, or the experimental configuration, in the corresponding row.

This section also presents, in Figures 7.1 and 7.2, the objective space for a selected DMOOP, which is constrained by a bounding box representing a decision-maker's preferences for two selected experimental configurations. The bounding box in these specific instances is a sphere.

The two figures present results for a randomly chosen run and environment among many environments (changes) that are typical of a single run of a DMOO.

In this section, and any other sections in this chapter, ALG:1, ALG:2 and ALG:3 are as defined in Section 7.2.1. These are the three proposed DMOAs in this study. Section 7.2.1 provides further details on these algorithms.

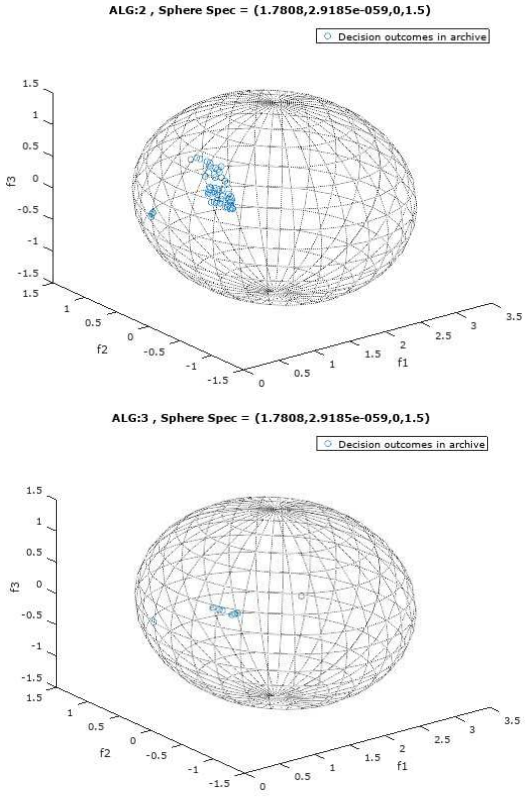
Table 7.4: Third Experimental Study: Overall Wins and Losses

RESULTS	ALG:1	ALG:2	ALG:3
Wins	346	403	274
Losses	322	292	409
Diff	24	111	-135
Rank	2	**1**	3

In Table 7.6, where results for six experimental configurations are presented, ALG:2 won four times, while ALG:1 won the other two experimental configurations ($n_t = 10, \tau_t = 2$ and $n_t = 10, \tau_t = 4$). For the six experimental configurations, ALG:3 never performed better than ALG:2. ALG:3 performed better than ALG:1 for two experimental configurations ($n_t = 1, \tau_t = 4$ and $n_t = 1, \tau_t = 2$), but in one ($n_t = 1, \tau_t = 2$) of those two experimental configurations, where it performed better than ALG:1, it suffered more losses than ALG:1.

Table 7.5 presents the performances of the proposed algorithms with respect to measures in Section 7.2.4. ALG:2 won in five of the eight measures. For two (react and dVD) of the five, the number of wins were tied with ALG:3. Results for the first four measures in Section 7.2.4 indicated that ALG:2 won for three (acc, hvr, react) out of

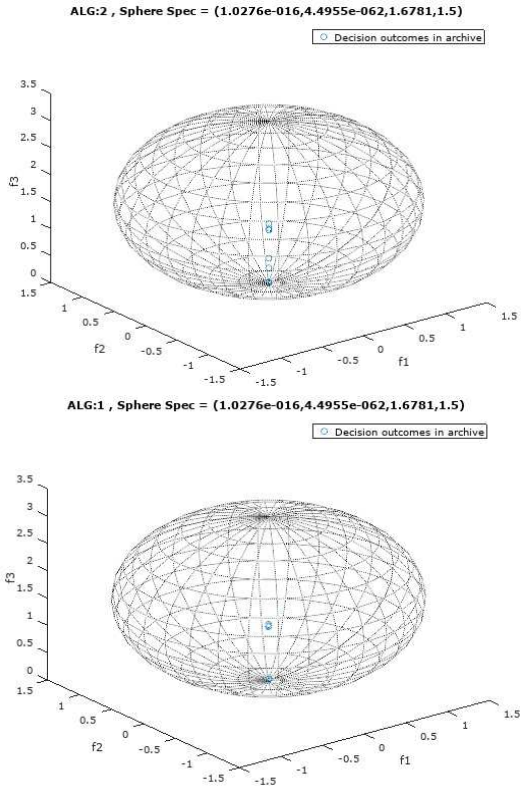
Figure 7.1: DMOA = 2DEVENS, DMOOP = FDA5, $n_t = 10\,\tau_t = 5$



the four measures. It won with the least number of losses for the accuracy measure, acc, making it the most accurate of the proposed algorithms in this study. For those four measures, ALG:3 won once (stab), but had the same number of losses as ALG:1 for the win. ALG:3 also had the highest number of worst rankings for those four measures.

ALG:2 had the highest number of wins for the new measures proposed in this study (refer to Section 7.2.4), namely two wins out of four measures, making ALG:2 the best of the proposed algorithms in terms of all the performance measures presented in Section 7.2.4. ALG:1 recorded the highest number of wins for the nNVD measure, while ALG:2 ranked first for the sNVD measure. Thus, ALG:1 and ALG:2 performed better than ALG:3 in finding non-violating decisions of a decision-maker within the search space. Even though ALG:3 ranked best for nVD and dVD, the magnitude of wins recorded by ALG:3 for those two measures were negligible. Despite ALG:3 ranking first for nVD and dVD, ALG:1 never lost to any other algorithm for those measures. ALG:2 tied with

Figure 7.2: DMOA = 2DEVENS, DMOOP = FDA5, $n_t = 10\,\tau_t = 2$



ALG:3 for the dVD measure.

Table 7.4 presents the overall results. ALG:1 ranked first with 403 wins, ALG:2 recorded 346 wins, while ALG:3 ranked last. In addition, ALG:3 recorded the highest number of overall losses, plus the most negative overall DIFF. ALG:2 recorded the least number of overall losses, and the most positive overall DIFF. These overall results are consistent with the earlier results, which showed that ALG:2 led on most of the measures and experimental configurations, while ALG:3 consistently lagged behind the other two proposed algorithms.

Figures 7.1 and 7.2 present the objective space where the preferred objective vectors, or preferred outcomes in the space, are contained in a decision-maker's preference set. The preference set or the bounding box, in these instances is a sphere whose defining properties are specified by a decision-maker in a bootstrap procedure that is described in Algorithm 12. For the sphere specifications in Figures 7.1 and 7.2, the first three

numbers represent the center of the sphere, while the last number represents the radius of the sphere.

Figures 7.1 and 7.2 are simply snapshots, thus are incapable of showing the dynamics of the preference set. They are, however, presented in this section to provide a one-time view into the state of the objective space during the optimization process.

In all the snapshots presented in Figures 7.1 and 7.2, all the decisions in the archive were preferred by the decision-maker, because all the objective vectors laid within the spheres representing the decision-maker's preferences. This is a testament to the fact that the proposed algorithms are effective in finding optimal trade-off solutions/decisions that reflect a decision-maker's preferences within the search space.

ALG:2 in Figure 7.1 had the highest number of preferred vectors/outcomes within its spheres, which is consistent with the earlier results in this section about its overall superiority over the other algorithms proposed in this study. As a matter of fact, it was ranked best for the experimental configuration represented by Figure 7.1. ALG:3 was ranked worst for the experimental configuration represented by Figure 7.1.

In the configuration represented by Figure 7.2, ALG:1 ranked best, but only marginally better than ALG:2 in that experiment. Both algorithms effectively found the decision-maker's preferred decisions, as none of the algorithms produced violating decisions.

7.4 Conclusion

The experimental study presented in this chapter proposed an approach for incorporating a decision-maker's preferences in a DMOA. The results showed that a decision-maker's preferences can effectively be specified using the approach proposed in this study, which is partly apriori and interactive. The results indicated that the proposed bounding box specification is an effective mathematical abstraction for a decision-maker's preferences. The study also proposed three algorithmic approaches for solving DMOOPs in the context of a decision-maker's preferences. The three proposed DMOAs showed good results, with a varying degree of performance. Death-Penalty, ALG:2, proved to be the overall winner, while Restrict-Search-To-Feasible-Region, ALG:3, lagged behind the other proposed algorithms. Also proving to be effective proposals of this study are

the four performance measures that specifically estimate the performances of DMOAs in the context of a decision-maker's preferences.

7.5 Summary

This chapter presented one of the experimental studies conducted in this research. The study investigated decision-making in the context of DMOOs. The study proposed a formalization of decision-makers' preferences. The study also proposed algorithmic approaches on how the preferences may be used to impart an optimization process. A comparative study of the approaches were conducted and the results were presented. The next chapter presents a summary of the various conclusions emanating from all the experimental studies carried out in this research.

Table 7.5: Third Experimental Study: Overall Wins and Losses for various Performance Measures

PM	RESULTS	ALG:1	ALG:2	ALG:3
acc	Wins	69	83	63
acc	Losses	75	60	80
acc	Diff	-6	23	-17
acc	Rank	2	**1**	3
stab	Wins	23	23	32
stab	Losses	22	34	22
stab	Diff	1	-11	10
stab	Rank	2	2	**1**
hvr	Wins	82	94	40
hvr	Losses	62	50	104
hvr	Diff	20	44	-64
hvr	Rank	2	**1**	3
react	Wins	14	45	45
react	Losses	57	25	22
react	Diff	-43	20	23
react	Rank	3	**1**	**1**
nNVD	Wins	91	65	39
nNVD	Losses	38	67	90
nNVD	Diff	53	-2	-51
nNVD	Rank	**1**	2	3
sNVD	Wins	67	87	47
sNVD	Losses	68	48	85
sNVD	Diff	-1	39	-38
sNVD	Rank	2	**1**	3
nVD	Wins	0	2	4
nVD	Losses	0	4	2
nVD	Diff	0	-2	2
nVD	Rank	3	2	**1**
dVD	Wins	0	4	4
dVD	Losses	0	4	4
dVD	Diff	0	0	0
dVD	Rank	3	**1**	**1**

Table 7.6: Third Experimental Study: Overall Wins and Losses with various Frequency and Severity

n_t	τ_t	RESULTS	ALG:1	ALG:2	ALG:3
10	4	Wins	64	59	53
10	4	Losses	54	58	64
10	4	Diff	10	1	-11
10	4	Rank	**1**	2	3
10	5	Wins	61	67	32
10	5	Losses	45	40	75
10	5	Diff	16	27	-43
10	5	Rank	2	**1**	3
10	2	Wins	73	60	35
10	2	Losses	40	54	74
10	2	Diff	33	6	-39
10	2	Rank	**1**	2	3
1	4	Wins	41	67	56
1	4	Losses	66	44	54
1	4	Diff	-25	23	2
1	4	Rank	3	**1**	2
1	5	Wins	56	73	43
1	5	Losses	53	47	72
1	5	Diff	3	26	-29
1	5	Rank	2	**1**	3
1	2	Wins	51	77	55
1	2	Losses	64	49	70
1	2	Diff	-13	28	-15
1	2	Rank	3	**1**	2

Chapter 8

Conclusions

This chapter presents a summary of all the studies conducted in this dissertation. Section 8.1 presents summaries of various conclusions reached from the studies. Possible future work derived from the studies are presented in Section 8.2.

8.1 Summary of Conclusions

The main objective of this dissertation is to propose new algorithmic procedures that can facilitate the incorporation of decision-makers (decision-makers)' preferences in the dynamic multi-objective optimization solution process. The algorithmic procedures use nature-inspired and evolutionary techniques. A differential evolution algorithm, dynamic differential evolution vector-evaluated non-dominated sorting (2DEVENS), was developed. Three experimental studies were conducted. Two of the studies compared 2DEVENS with two benchmark algorithms. The last of the three studies examined the incorporation of decision maker preferences into the 2DEVENS algorithm. Different methods of preference incorporation were examined, resulting in different adaptations of the 2DEVENS algorithm. The major findings and conclusions of the three studies are as follows:

First Study: This was presented in Chapter 5. In that study, 2DEVENS was compared with two other nature-inspired algorithms, namely the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm and the dynamic non-dominated sort

genetic algorithm version II (DNSGA-II). The results showed that 2DEVENS was able to track the Pareto-optimal Fronts (POFs) of the problems considered in the study. The results also showed that 2DEVENS outperformed the other algorithms on some of the performance measures and experimental configurations. It specifically ranked overall best for the accuracy and stability measures. The conclusion reached by this study was that 2DEVENS would be a choice algorithm for problems where accuracy and stability are important.

Second Study: This was presented in Chapter 6. Again, 2DEVENS was able to track the POFs of the problems considered in the study. Some of the experimental configurations adopted in the study, such as $(n_t = 1, \tau_t = 10)$ and $(n_t = 10, \tau_t = 10)$, were different from that of the First Study. Also, new benchmark functions, such as HE1 and HE3, were added, which were not used in the First Study. New characteristics, such as discontinuities in the objective space, were introduced by the new benchmark functions. 2DEVENS, again, showed a very good performance. Overall, it was the second best, following DNSGA-II. However, 2DEVENS remained the overall best performing algorithm for the accuracy measure, still making it the best for the accuracy measure in two successive experiments.

Third Study: This was presented in Chapter 7. In this study, 2DEVENS was adapted by incorporating decision maker preferences. Three different approaches were used in incorporating the preferences and the approaches were compared. The approaches are: Proportionate-Penalty, which is otherwise known as proportionate-penalty algorithm (ALG:1), Death-Penalty, which is otherwise known as death-penalty algorithm (ALG:2) and Restrict-Search-To-Feasible-Region, which is otherwise known as restrict-search-to-feasible-region algorithm (ALG:3). The workings of these approaches were presented in Chapter 7. The three approaches showed abilities to track solutions of the optimization problems that were considered in the study in light of the preferences specified by the decision makers. The study demonstrated the ability of 2DEVENS to work with the various adaptations in order to solve optimization problems that reflect decision maker preferences. The study also demonstrated the applicability of the new performance measures to evaluate algorithms seeking solutions to optimization problems where preferences are specified by decision makers. These new measures are: nNVD, which measures the

number of non-violating decisions, sNVD, which measures the spread of non-violating decisions, nVD, which measures the number of violating decisions and dVD, which measures the deviation of the violating decisions.

8.2 Future Work

First and Second Studies: In future, additional comparative studies will be conducted between 2DEVENS and other nature-inspired algorithms. The additional studies will be conducted for more benchmark functions, experimental configurations and algorithms' control parameters. Also, future studies will explore improvements on 2DEVENS, so as to experiment with the prospects of it outperforming the other algorithms in situations, e.g. benchmark functions, performance measures and experimental configurations, where it has been outperformed.

Third Study: Future work will consider the possibilities of experimenting with some of the geometric properties of the bounding box, and measuring the performance implications on the proposed algorithms. Also, the proposed algorithms will be applied on additional problems in [84]. Lastly, an investigation will be conducted on how decision making under risks may be incorporated into a decision-maker's preferences and how the proposed algorithms will perform in the presence of the risks. Ideas of decision making prevalent in the multi-criteria decision making research community, such as full aggregation of preferences, outranking, etc., would also be explored in future work.

Bibliography

[1] David H Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[2] J Aczél and T L Saaty. Procedures for synthesizing ratio judgements. *Journal of Mathematical Psychology*, 27(1):93–102, 1983.

[3] M. M. Ali and A. Törn. Population set-based global optimization algorithms: Some modifications and numerical studies. *Computers and Operations Research*, 31(10):1703–1725, 2004.

[4] Anonymous. Wind Turbine Paper. *AIMU Technical Services Committee*, (January):1 –24, 2012.

[5] Zikrija Avdagić, Samim Konjicija, and Samir Omanović. *Evolutionary Approach to Solving Non-stationary Dynamic Multi-Objective Problems*, pages 267–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[6] Demet Ayvaz, Haluk Rahmi Topcuoglu, and Fikret Gurgen. Performance evaluation of evolutionary heuristics in dynamic environments. *Applied Intelligence*, 37(1):130–144, July 2012.

[7] Carlos R B Azevedo and Aluizio F R Araujo. Generalized immigration schemes for dynamic evolutionary multiobjective optimization. *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, pages 2033–2040, 2011.

[8] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. Articulating Decision Maker ' s Preference Information within Multiobjective Artificial Immune Systems. 2012.

[9] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. *Recent Advances in Evolutionary Multi-objective Optimization*, volume 20. 2017.

[10] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. *Recent Advances in Evolutionary Multi-objective Optimization*, volume 20. 2017.

[11] Michael Bacharach and Susan Hurley. *Essays in the Foundations of Decision Theory*. Blackwell, 1991.

[12] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.

[13] James E Baker. Adaptive Selection Methods for Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[14] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[15] M Bellmore and G L Nemhauser. The Traveling Salesman Problem: A Survey. *Operations Research*, 16(3):538–558, 1968.

[16] Eva Besada-Portas, Luis de la Torre, Alejandro Moreno, and Jos L. Risco-Martn. On the performance comparison of multi-objective evolutionary uav path planners. *Information Sciences*, 238(Supplement C):111 – 125, 2013.

[17] Ken Binmore. *Rational Decisions*. Princeton University Press, stu - student edition edition, 2009.

[18] Tim Blackwell and Jürgen Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

[19] H J Bremermann. Optimization through evolution and recombination. In M C Yovits, G T Jacobi, and G D Golstine, editors, *Proceedings of the Conference on*

Self-Organizing Systems – 1962, pages 93–106, Washington, DC, 1962. Spartan Books.

[20] R. Brits, A. P. Engelbrecht, and F. Van Den Bergh. Solving systems of unconstrained equations using particle swarm optimization. *IEEE International Conference on Systems, Man and Cybernetics*, 3:2–7, 2002.

[21] Riaan Brits, Andries P Engelbrecht, and F Van den Bergh. A niching particle swarm optimizer. *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, 2(September):692–696, 2002.

[22] L.T. Bui, H.A. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. *2005 IEEE Congress on Evolutionary Computation*, 3:2349–2356, 2005.

[23] Mario Cámara, Julio Ortega, and Francisco de Toro. A Single Front Genetic Algorithm for Parallel Multi-objective Optimization in Dynamic Environments. *Neurocomput.*, 72(16-18):3570–3579, oct 2009.

[24] Colin Camerer and Martin Weber. Recent developments in modeling preferences: Uncertainty and ambiguity. *Journal of Risk and Uncertainty*, 5(4):325–370, 1992.

[25] Jake Chandler. Descriptive decision theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2017 edition, 2017.

[26] Cheng Liang Chen and Wen Cheng Lee. Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Computers and Chemical Engineering*, 28(6-7):1131–1144, 2004.

[27] Shi Cheng, Yuhui Shi, and Quande Qin. *On the Performance Metrics of Multiobjective Optimization*, pages 504–512. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[28] 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, 2002.

[29] Maurice Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3:1951–1957, 1999.

[30] Molly Cochran. *The Cambridge Companion to Dewey*. Cambridge Companions to Philosophy. Cambridge University Press, 2010.

[31] C a C Coello, G T Pulido, and M S Lechuga. Handling multiple objectives with particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):256–279, 2004.

[32] Carlos A Coello Coello, Garry B Lamont, and David A Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer US, 2007.

[33] Carlos a. Coello Coello and Margarita Reyes-Sierra. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[34] Yann Collette and Patrick Siarry. *Multiobjective Optimization*. Springer Berlin Heidelberg, 2004.

[35] Demetrakis Constantinou. Ant colony optimisation algorithms for solving multi-objective power-aware metrics for mobile ad hoc networks. (August), 2011.

[36] Laura Cruz-Reyes, Eduardo Fernandez, Claudia Gomez, and Patricia Sanchez. *Preference Incorporation into Evolutionary Multiobjective Optimization Using a Multi-Criteria Evaluation Method*. Springer International Publishing, 2014.

[37] Laura Cruz-Reyes, Eduardo Fernandez, Patricia Sanchez, Carlos A Coello Coello, and Claudia Gomez. Incorporation of implicit decision-maker preferences in multi-objective evolutionary optimization using a multi-criteria classification method. *Applied Soft Computing Journal*, 50:48–57, 2017.

[38] X. Cui, C. T. Hardin, R. K. Ragade, T. E. Potok, and A. S. Elmaghraby. Tracking non-stationary optimal solution by particle swarm optimizer. *Proceedings - Sixth Int. Conf. on Softw. Eng., Artificial Intelligence, Netw. and Parallel/Distributed Computing and First ACIS Int. Workshop on Self-Assembling Wireless Netw., SNPD/SAWN 2005*, 2005:133–138, 2005.

[39] Rc D, Carlos Artemio Coello Coello, Ruben Ruiz-femenia, Jose a Caballero, Carlos Artemio Coello Coello, Javad Sadri, Student Member, Ching Y Suen, J Kennedy, R C Eberhart, Y Shi, R C Eberhart, Wei Hong Lim, Nor Ashidi, and Mat Isa. *Constraint-handling Techniques Used with Evolutionary Algorithms*, volume 5. 1997.

[40] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, London, 1859.

[41] Swagatam Das, Ajith Abraham, Uday K. Chakraborty, and Amit Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, 13(3):526–553, 2009.

[42] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[43] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Ann Arbor, MI, USA, 1975.

[44] K Deb, S Agrawal, A Pratap, and T Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Proceedings of 6th International Conference on Parallel Problem Solving from Nature, PPSN VI*, 2000.

[45] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evol. Comput.*, 7(3):205–230, September 1999.

[46] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated Binary Crossover for Continuous Search Space The crossover operator is believed to be the main search operator in the working of a genetic. *Complex Systems*, 9(2):115–148, 1994.

[47] Kalyanmoy Deb, N Udaya Bhaskara Rao, and S Karthik. Dynamic Multi-objective Optimization and Decision-making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling. In *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, pages 803–817, Berlin, Heidelberg, 2007. Springer-Verlag.

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

[49] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable multi-objective optimization test problems*, volume 1, pages 825–830. IEEE Computer Society, 2002.

[50] Kalyanrnoy Deb, Dhiraj Joshi, and Ashish Anand. Real-coded evolutionary algorithms with parent-centric recombination. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, 1:61–66, 2002.

[51] Richard E. Rosenthal. Concepts, theory, and techniques: Principles of multiobjective optimization. *Decision Sciences*, 16:133 – 152, 06 2007.

[52] R.C. Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1:94–100.

[53] R.C. Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1:81–86, 2001.

[54] R.C. Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1:81–86, 2001.

[55] Russ Eberhart, Pat Simpson, and Roy Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, Inc., San Diego, CA, USA, 1996.

[56] Andries P Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, Chichester, England, dec 2002.

[57] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[58] Andries P Engelbrecht. Computational intelligence: an introduction, 2007.

[59] Larry J Eshelman, Richard A Caruana, and J David Schaffer. Biases in the Crossover Landscape. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[60] Larry J Eshelman and J David Schaffer. Real-Coded Genetic Algorithms and Interval-Schemata. In L DARRELL WHITLEY, editor, *Foundations of Genetic Algorithms*, volume 2 of *Foundations of Genetic Algorithms*, pages 187–202. Elsevier, 1993.

[61] M Farina, K Deb, and P Amato. Dynamic multiobjective optimization problem: Test cases, approximation, and applications, volume = 8, year = 2003. *Genetic Evol. Comput. Conf*, (5):311–326.

[62] Thiago Nascimento Ferreira, Silvia Regina Vergilio, and Jerffeson Teixeira de Souza. Incorporating User Preferences in Search-Based Software Engineering: A Systematic Mapping Study. *Information and Software Technology*, 0:1–15, 2017.

[63] Reynolds Number Flyers. an Introduction To Flapping Wing. 2013.

[64] David B Fogel. *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*. Ginn Press, 1991.

[65] David B Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.

[66] David Bruce Fogel. *Evolving Artificial Intelligence*. PhD thesis, La Jolla, CA, USA, 1992.

[67] Lawrence J. Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[68] F.-A. Fortin and M. Parizeau. Revisiting the NSGA-II crowding-distance computation. *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13*, page 623, 2013.

[69] A S Fraser. Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Science*, 10:484–491, 1957.

[70] A S Fraser. Simulation of Genetic Systems by Automatic Digital Computers II. Effects of Linkage on Rates of Advance Under Selection. *Australian Journal of Biological Sciences*, 10(4):492–500, dec 1957.

[71] Fabio Furini, Ivana Ljubi, and Markus Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *European Journal of Operational Research*, 262(2):438 – 448, 2017.

[72] Chi-Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *Trans. Evol. Comp*, 13(1):103–127, February 2009.

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

[74] David E Goldberg. Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*, 1989.

[75] Erik D. Goodman. Introduction to genetic algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 671–692, New York, NY, USA, 2012. ACM.

[76] Fillipe Goulart and Felipe Campelo. Preference-guided evolutionary algorithms for many-objective optimization. *Information Sciences*, 329:236–255, 2016.

[77] Viviane Grunert da Fonseca, Carlos M. Fonseca, and Andreia O. Hall. Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function. 1993:213–225, 2001.

[78] Juha Mantysaari Hamalainen, Raimo P. A Dynamic Interval Goal Programming Approach to the Regulation of a Lake River System. *Journal of Multi-Criteria DEcision Analysis*, 86(December 1999):75–86, 2001.

[79] Mohammad Hamdan. On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms. *Computing and Informatics*, 29(5):783–800, 2010.

[80] Mohammad M. Hamdan. The Distribution Index in Polynomial Mutation for Evolutionary Multiobjective Optimisation Algorithms: An Experimental Study. *Proceedings of International Conference on Electronics Computer Technology*, 2012.

[81] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. *IMM Technical Report IMM-REP-1998-7*, page 31, 1998.

[82] Kyle Robert Harrison, Beatrice Ombuki-Berman, and Andries P. Engelbrecht. Knowledge transfer strategies for vector evaluated particle swarm optimization. In Robin C. Purshouse, Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco, and Jane Shaw, editors, *Evolutionary Multi-Criterion Optimization*, pages 171–184, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[83] Iason Hatzakis and David Wallace. Dynamic multi-objective optimization with evolutionary algorithms: A forward-looking approach. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 1201–1208, New York, NY, USA, 2006. ACM.

[84] Mardé Helbig. *Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation.* PhD thesis, University of Pretoria, 2012.

[85] Marde Helbig and Andries P. Engelbrecht. Archive management for dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. *Proc.IEEE CEC*, pages 2047–2054, 2011.

[86] Mardé Helbig and Andries P. Engelbrecht. Analyses of guide update approaches for vector evaluated particle swarm optimisation on dynamic multi-objective optimisation problems. *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, pages 10–15, 2012.

[87] Mardé Helbig and Andries P Engelbrecht. Analysing the performance of dynamic multi-objective optimisation algorithms. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 1531–1539, 2013.

[88] Marde Helbig and Andries P. Engelbrecht. Benchmarks for dynamic multi-objective optimisation. *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, 46(3):84–91, 2013.

[89] Mardé Helbig and Andries P Engelbrecht. Issues with performance measures for dynamic multi-objective optimisation. In *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2013 IEEE Symposium on*, pages 17–24. IEEE, 2013.

[90] Mardé Helbig and Andries P Engelbrecht. Performance measures for dynamic multi-objective optimisation algorithms. *Information Sciences*, 250:61–81, 2013.

[91] R. Hinterding. Gaussian mutation and self-adaption for numeric genetic algorithms. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, 1:384, 1995.

[92] Jesper Normann Hoffmeyer. *The Swarming Body*, pages 937–940. Mouton de Gruyter, 1997.

[93] John H Holland. ECHO: Explorations of Evolution in a Minature World. *Artificial Life II*, X, 1991.

[94] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[95] John H Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[96] Liang Huang, Il Hong Suh, and Ajith Abraham. Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants. *Information Sciences*, 181(11):2370–2391, 2011.

[97] Simon Huband, Phil Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[98] Cagatay Iris and Seyda Serdar Asan. Computational Intelligence Systems in Industrial Engineering. *Computational Intelligence Systems in Industrial Engineering*, 6:203–230, 2012.

[99] Amitay Isaacs, Tapabrata Ray, and Warren Smith. Memetic algorithm for dynamic bi-objective optimization problems. *2009 IEEE Congress on Evolutionary Computation*, pages 1707–1713, 2009.

[100] Alessio Ishizaka and Phillipe Nemery. *Multicriteria decision analysis: methods and software*. Wiley, 2013.

[101] Antonio López Jaimes, Alfredo Arias Montaño, and Carlos A Coello Coello. Preference incorporation to solve many-objective airfoil design problems. *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, pages 1605–1612, 2011.

[102] Paul A Jensen and Jonathan F Bard. *Operations research models and methods*. John Wiley & Sons Incorporated, 2003.

[103] F. Ji-Pyng Chiou and Wang. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. *1998 IEEE International*

Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pages 627–632, 1998.

[104] Lina Jia, Sanyou Zeng, and Dong Zhou. Dynamic Multi-objective Differential Evolution for Solving Constrained Optimization Problem. pages 2649–2654, 2011.

[105] Shouyong Jiang and Shengxiang Yang. A benchmark generator for dynamic multi-objective optimization problems. *2014 14th UK Workshop on Computational Intelligence, UKCI 2014 - Proceedings*, (Cci), 2014.

[106] Shouyong Jiang and Shengxiang Yang. A framework of scalable dynamic test problems for dynamic multi-objective optimization. *IEEE SSCI 2014: 2014 IEEE Symposium Series on Computational Intelligence - CIDUE 2014: 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, Proceedings*, (Cci):32–39, 2015.

[107] C Z Jonikow and Z Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, USA, July 1991*, 1991.

[108] R. Joshi and A.C. Sanderson. Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(1):63–76, 1999.

[109] Kahnemann and Tversky. Prospect Theory: an Analysis of Decision Under Risk. 47(2):263–291, 2009.

[110] James Kennedy. The behavior of particles. In V W Porto, N Saravanan, D Waagen, and A E Eiben, editors, *Evolutionary Programming VII*, pages 579–589, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[111] James Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3:1931–1938, 1999.

[112] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995.

[113] James Kennedy and Rui Mendes. Population structure and particle swarm performance.pdf. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, pages 1671–1676, 2002.

[114] Joshua D. Knowles. Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimzation. (February 2002), 2002.

[115] Wee Tat Koo, Chi Keong Goh, and Kay Chen Tan. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. *Memetic Computing*, 2(2):87–110, Jun 2010.

[116] Thiemo Krink and Rasmus K Ursem. Exploring the Performance of an Evolutionary Algorithm. pages 195–201, 2002.

[117] A. Kumar. Encoding Schemes in Genetic Algorithm. *International Journal of Advanced Research in IT and Engineering*, 2(3):1–7, 2013.

[118] R Lakshmi and K Vivekanandan. Performance analysis of a novel crossover technique on permutation encoded genetic algorithms. In *2014 International Conference on Advances in Engineering and Technology (ICAET)*, pages 1–4, may 2014.

[119] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231 – 247, 1992.

[120] Xiaodong Li, Jürgen Branke, and Tim Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 51–58, New York, NY, USA, 2006. ACM.

[121] Xiaodong Li and Khanh Hoa Dam. Comparing particle swarms for tracking extrema in dynamic environments. *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*, 3:1772–1779, 2003.

[122] I. L. Lopez Cruz, L. G. Van Willigenburg, and G. Van Straten. Efficient Differential Evolution algorithms for multimodal optimal control problems. *Applied Soft Computing Journal*, 3(2):97–122, 2003.

[123] Ilya G Loshchilov. Comparison of Multiobjective Evolutionary Algorithms., 2009.

[124] Mrinmoy Majumder. Impact of Urbanization on Water Shortage in Face of Climatic Aberrations. pages 35–48, 2015.

[125] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[126] Silvano Martello, David Pisinger, and Paolo Toth. New trends in exact algorithms for the 01 knapsack problem. *European Journal of Operational Research*, 123(2):325 – 332, 2000.

[127] Rui Mendes. DynDE : a Differential Evolution for Dynamic Optimization Problems. pages 2808–2815, 2005.

[128] Leon Messerschmidt and Andries P. Engelbrecht. Learning to play games using a PSO-based competitive learning approach. *IEEE Transactions on Evolutionary Computation*, 8(3):280–288, 2004.

[129] Computation Methods and Zbigniew Michalewicz. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. *Evolutionary Programming*, 4(Holland 1975):135–155, 1995.

[130] Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[131] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK, 1996.

[132] Amir Nakib and Patrick Siarry. *Performance Analysis of Dynamic Optimization Algorithms*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[133] Su Nguyen, Mengjie Zhang, and Kay Chen Tan. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings*, pages 2781–2788, 2015.

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

[135] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, second edition*. World Scientific, 2006.

[136] Isao Ono, Hajime Kita, and Shigenobu Kobayashi. Advances in Evolutionary Computing. chapter A Real-coded Genetic Algorithm Using the Unimodal Normal Distribution Crossover, pages 213–237. Springer-Verlag New York, Inc., New York, NY, USA, 2003.

[137] P.H. Raimo P and M. Juha. Dynamic multi-objective heating optimization. *European Journal of Operational Research*, 142(1):1–15, 2002.

[138] Nikhil Padhye, Pulkit Mittal, and Kalyanmoy Deb. Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. *Computational Optimization and Applications*, 62(3):851–890, 2015.

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

[140] V Pareto. *Cours D'Economie Politique*. F Rouge, 1896.

[141] K E Parsopoulos, D K Tasoulis, M N Vrahatis, and Key Words. Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization. In

In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004, pages 823–828. ACTA Press, 2004.

[142] K. E. Parsopoulos and M. N. Vrahatis. Particle Swarm Optimizer In Noisy And Continuously Changing Environments. *Methods*, pages 289–294, 2001.

[143] K E Parsopoulos and M N Vrahatis. Particle swarm optimization method in multiobjective problems. In *ACM Symposium on Applied Computing*, pages 603–607. ACM, 2002.

[144] Konstantinos E Parsopoulos, D K Tasoulis, N G Pavlidis, V P Plagianakos, and Michael N Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, 1:204–211, 2004.

[145] Martin Pelikan. *Genetic Algorithms*. 2010.

[146] Martin Peterson. *An Introduction to Decision Theory*. Cambridge Introductions to Philosophy. Cambridge University Press, 2009.

[147] Dilip Pratihar, Kalyanmoy Deb, and Amitabha Ghosh. Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots. *Engineering Optimization+A35*, 32:117–142, 09 1999.

[148] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[149] R Core Team. R: A Language and Environment for Statistical Computing, 2017.

[150] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M. Salama. Opposition-based differential evolution. *Studies in Computational Intelligence*, 143(1):155–171, 2008.

[151] Vitorino Ramos, Carlos Fernandes, and Agostinho C. Rosa. Social Cognitive Maps, Swarm Perception and Distributed Search on Dynamic Landscapes. 2005.

[152] Asanga Ratnaweera, Saman K. Halgamuge, and Harry C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255, 2004.

[153] I. Rechenberg. Cybernetic solution path of an experimental problem. In *Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans.* Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants, August 1965.

[154] Jon Reed, Robert Toombs, and Nils Aall Barricelli. Simulation of biological evolution and machine learning: I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology*, 17(3):319–342, 1967.

[155] J.-M. Renders and H. Bersini. Hybridizing genetic algorithms with hill-climbing methods for\nglobal optimization: two possible ways. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 1–6, 1994.

[156] Margarita Reyes-Sierra and C A Coello Coello. Multi-objective particle swarm optimizers: survey of the state-of-the-art. *International journal of computational intelligence research*, 2(3):287–308, 2006.

[157] R.G. Reynolds. Version space controlled genetic algorithms (VGA). *[1991] Proceedings. The Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pages 6–14, 1991.

[158] R.G. Reynolds and W. Sverdlik. Problem solving using cultural algorithms. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 645–650, 1994.

[159] Robert G Reynolds. New Ideas in Optimization. chapter Cultural Algorithms: Theory and Applications, pages 367–378. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[160] C D Rosin and R K Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.

[161] Christopher Rosin and Richard Belew. Methods for Competitive Co-evolution: Finding Opponents Worth Beating. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, 1995.

[162] Shahin Rostami, Dean O'Reilly, Alex Shenfield, and Nicholas Bowring. A novel preference articulation operator for the Evolutionary Multi-Objective Optimisation of classifiers in concealed weapons detection. *Information Sciences*, 295:494–520, 2015.

[163] B Roy. The Optimisation Problem Formulation: Criticism and Overstepping. *Journal of the Operational Research Society*, 32(6):427–436, jun 1981.

[164] Michael Ruse. Charles Darwin's On the Origin of Species. *Topoi*, 26(1):159–165, 2007.

[165] Briseida Sarasola and Enrique Alba. *Quantitative Performance Measures for Dynamic Optimization Problems*, pages 17–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[166] Yamini Sarathi, Khemraj Patel, Arti Tirkey, Prakash Kumar Sen, and Ritesh Sharma. Study on Wind Turbine and Its. 4(1):249–256, 2015.

[167] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithm. In *Proceeding of the First International Conference of Genetic Algorithms and Their Application*, pages 93–100, 1985.

[168] A S Sevenster and A P Engelbrecht. GARTNet: A Genetic Algorithm for Routing in Telecommunications Networks, 1996.

[169] Xiao-Ning Shen and Xin Yao. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 298(219):198–224, 2015.

[170] Y Shi and R Eberhart. A modified particle swarm optimizer. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, 1998.

[171] Yuhui Shi and Russell Eberhart. Empirical study of particle swarm optimization, 1999.

[172] Yuhui Shi and Russell C Eberhart. Parameter selection in particle swarm optimization. In V W Porto, N Saravanan, D Waagen, and A E Eiben, editors, *Evolutionary Programming VII*, pages 591–600, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[173] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, (February):515–519, 2015.

[174] Karthik Sindhya. An Introduction to Nature Inspired Algorithms. 2012.

[175] R Sivaraj and T Ravichandran. A review of selection methods in genetic algorithm. *Int J Eng Sci Tech*, 3(5):3792–3797, 2011.

[176] Mario Cámara Sola. *Parallel Processing for Dynamic Multi-objective Optimization*. PhD thesis, 2010.

[177] N Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[178] Rainer Storn, Siemens Ag, Z F E T Sn, Otto-hahn Ring, and D Muenchen. On the Usage of Differential Evolution for Function Optimization. 1996.

[179] Rainer Storn and Kenneth Price. Minimizing the real functions of the ICEC'96 contest by differential evolution. *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 842–844, 1996.

[180] Rainer Storn and Kenneth Price. Differential Evolution &Ndash; A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization*, 11(4):341–359, dec 1997.

[181] Sufian Sudenga and Naruemon Wattanapongsakornb. Incorporating decision maker preference in multiobjective evolutionary algorithm. *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIES 2014: 2014 IEEE Symposium on Computational Intelligence for Engineering Solutions, Proceedings*, pages 22–29, 2015.

[182] P. N. Suganthan. Particle swarm optimiser with neighbourhood operator. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3:1958–1962, 1999.

[183] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[184] Gilbert Syswerda. Schedule Optimization Using Genetic Algorithms. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349, New York, 1991. Van Nostrand Reinhold.

[185] Lothar Thiele, Lothar Thiele, Kaisa Miettinen, Kaisa Miettinen, Pekka J Korhonen, Pekka J Korhonen, Julian Molina, and Julian Molina. A Preference-Based Evolutionary Algorithm for Multi-Objective Optimization. *Evolutionary Computation*, 17(3):411–436, 2009.

[186] Marc R. Tool. John fagg foster 1907-1985. *Journal of Economic Issues*, 20(1):1–3, 1986.

[187] Ioan Cristian Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.

[188] Evangelos Triantaphyllou. Multi-Criteria Decision Making Methods : A Comparative Study.

[189] Nguyen Minh Triet, Nguyen Ngoc Viet, and Pham Manh Thang. Aerodynamic Analysis of Aircraft Wing. *VNU Journal of Science: Mathematics Physics*, 31(2):68–75, 2015.

[190] Shigeyoshi Tsutsui and David E. Goldberg. Simplex crossover and linkage identification: Single-stage evolution vs. multi-stage evolution. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, 1(2):974–979, 2002.

[191] F. Van Den Bergh and A. P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.

[192] D A van Veldhuizen. *Multiobjective evolutionary algorithms: classification, analyses, and new innovations*. PhD thesis, Graduate School of Engineering Air University, 1999.

[193] David Allen Van Veldhuizen and Gary B Lamont. On measuring multiobjective evolutionary algorithm performance. *Proceedings of the Congress on Evolutionary Computation, 2000.*, 1:204–211 vol.1, 2000.

[194] Jakob Vesterstrøm and R Thomsen. A Comparative Study of Differential Evolution , Particle Swarm Optimization , and Evolutionary Algorithms on Numerical Benchmark Problems. *Congress on Evolutionary Computation, 2004. CEC2004.*, 2:1980 – 1987, 2004.

[195] Shuzhen Wan and Diangang Wang. A Novel Differential Evolution for Dynamic Multiobjective Optimization with Adaptive Immigration Scheme. pages 502–507, 2013.

[196] Gai Yun Wang and Dong Xue Han. Particle swarm optimization based on self-adaptive acceleration factors. *3rd International Conference on Genetic and Evolutionary Computing, WGEC 2009*, (1):637–640, 2009.

[197] Darrell Whitley and Andrew M Sutton. *Genetic Algorithms — A Survey of Models and Methods*, pages 637–671. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[198] Darrell Whitley and Nam-Wook Yoo. Modeling Simple Genetic Algorithms for Permutation Problems. volume 3 of *Foundations of Genetic Algorithms*, pages 163–184. Elsevier, 1995.

[199] L Darrell Whitley, Timothy Starkweather, and D'Ann Fuquay. Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 133–140, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[200] Wikipedia. On the origin of species — wikipedia, the free encyclopedia, 2017. [Online; accessed 28-October-2017].

[201] Wikipedia contributors. Analytic hierarchy process — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Analytic_hierarchy_process&oldid=851253400, 2018. [Online; accessed 27-July-2018].

[202] Wikipedia contributors. Preference ranking organization method for enrichment evaluation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Preference_ranking_organization_method_for_enrichment_evaluation&oldid=833303943, 2018. [Online; accessed 28-July-2018].

[203] M. J. Willis, H. G. Hiden, P. Marenbach, B. McKay, and G. A. Montague. Genetic programming: An introduction and survey of applications. *Second International Conference on Genetic Algorithms in Engineering Systems*, (446):314 – 319, 1997.

[204] Alden H Wright. Genetic Algorithms for Real Parameter Optimization. In Gregory J Rawlins, editor, *Foundations of genetic algorithms*, pages 205–218. Morgan Kaufmann, San Mateo, CA, 1991.

[205] Zhengjia Wu and Jianzhong Zhou. A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. *Proceedings - 2007 International Conference on Computational Intelligence and Security, CIS 2007*, pages 133–136, 2007.

[206] Xiaohui Hu and R.C. Eberhart. Adaptive particle swarm optimization: detection and response to dynamic systems. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, 2:1666–1670.

[207] Xin She Yang. *Nature-Inspired Optimization Algorithms*. 2014.

[208] X Yao and Y Liu. Fast evolutionary programming. *Proceedings of the 6ˆ{th} Annual Conference on Evolutionary Programming*, pages 451–460, 1996.

[209] Xin Yao, Guangming Lin, and Yong Liu. An Analysis of Evolutionary Algorithms Based on Neighborhood and Step Sizes. In *Proceedings of the 6th International Conference on Evolutionary Programming VI*, EP '97, pages 297–307, London, UK, UK, 1997. Springer-Verlag.

[210] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary Programming Made Faster y 1 Introduction. *IEEE Transactions on Evolutionary Computation*, 3(July 1999):82–102, 1999.

[211] Liron Yedidsion and Dvir Shabtay. The resource dependent assignment problem with a convex agent cost function. *European Journal of Operational Research*, 261(2):486 – 502, 2017.

[212] Daniela Zaharie. Influence of Crossover on the Behavior of Differential Evolution Algorithms. *Appl. Soft Comput.*, 9(3):1126–1138, jun 2009.

[213] Weiwei Zhang, Gary G Yen, and Zhongshi He. Constrained Optimization via Artificial Immune System, 2013.

[214] Stanley Zionts. Mcdm—if not a roman numeral, then what? *Interfaces*, 9(4):94–101, August 1979.

[215] 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, 1999.

[216] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *Trans. Evol. Comp*, 7(2):117–132, April 2003.

[217] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, June 2000.

[218] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2001.

[219] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Why quality assessment of multiobjective optimizers is difficult. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 666–674, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[220] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Carlos M Fonseca, Viviane Grunert Da Fonseca, M C Fonseca, and G V Fonseca. Why quality assessment of multiobjective optimizers is difficult. *Gecco*, pages 666–674, 2002.

Appendix A

Acronyms

This appendix provides an alphabetical listing of all acronyms used in this dissertation. Each acronym is typeset in bold and its meaning is provided alongside.

2DEVENS

dynamic differential evolution vector-evaluated non-dominated sorting vi, vii, 2–5, 28, 72, 73, 83–94, 98–109, 128, 129, 134–136

AHP

Analytical Hierarchy Process vi, 65, 66

ALG:1

proportionate-penalty algorithm 3, 4, 111, 127–130, 132, 133, 135, 174–197

ALG:2

death-penalty algorithm 3, 4, 111, 127–130, 132, 133, 135, 174–197

ALG:3

restrict-search-to-feasible-region algorithm 3, 4, 111, 127–130, 132, 133, 135, 174–197

DE

differential evolution 2, 3, 48–50, 52, 73–75, 81, 92

decision-maker

decision-maker 1–4, 22, 23, 54–65, 67, 68, 110–122, 126–131, 134, 136

DMOA

dynamic multi-objective optimization algorithm vi, vii, 1–5, 8, 11, 24, 26, 28, 32, 72–74, 80, 92–94, 102–104, 106, 108, 109, 112, 115, 116, 126–131

DMOO

dynamic multi-objective optimization 2, 40, 72, 74, 111, 127, 131

DMOOP

dynamic multi-objective optimization problem vi, vii, 1, 2, 4, 8–12, 18–20, 22, 24, 32, 47, 72–74, 78–81, 85, 86, 92–98, 101–106, 108–112, 116, 122, 123, 127–130

DNSGA-II

dynamic non-dominated sort genetic algorithm version II 2, 3, 28, 39, 40, 72, 73, 81–95, 98–107, 134, 135

DVEPSO

dynamic vector-evaluated particle swarm optimization 2, 3, 28, 40, 47, 48, 73, 81, 83–95, 98–107, 134

EA

evolutionary algorithm 33, 34, 48

GA

genetic algorithm 2, 31–34, 39, 40

MCDM

multi-criteria decision making 62–68, 71

MOA

multi-objective optimization algorithm 25

MOO

multi-objective optimization 9

MOOP

multi-objective optimization problem 8–10, 13–15, 21, 22, 25, 39, 110

NSGA

non-dominated sort genetic algorithm 39, 40

NSGA-II

non-dominated sort genetic algorithm version II 40

PCX

Parent-Centric Recombination 38

POF

Pareto-optimal Front 2, 9, 11–16, 18–26, 73, 74, 79, 80, 85, 86, 96–98, 101, 110, 115, 116, 118, 123, 135, 166, 169, 171

POS

Pareto-optimal Set 9, 11, 12, 15, 16, 18–20, 74, 79, 80, 96, 97, 110, 123, 166, 169, 171

PROMETHEE

Preference Ranking Organization Method for Enrichment Evaluation 66–68

PSO

particle swarm optimization 2, 40–43, 47, 48, 73

TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution 68, 69

Appendix B

List Of Symbols

This appendix lists the mathematical symbols used throughout this dissertation, and their definitions. The symbols used within each chapter are listed under separate sections. Each section lists only newly introduced symbols.

B.1 Chapter 2: Optimization

$\mathbf{f}$	Multi-objective function
f_k	k-th objective function in a multi-objective function $\mathbf{f}$
$\mathbf{x}$	Vector of decision variables
x^*	Solutions of multi-objective problem
n_k	Number of objective functions
n_g	Number of inequality constraints
n_h	Number of equality constraints
n_x	Number of decision variables
x_{min}	Minimum value of decision variable
x_{max}	Maximum value of decision variable
g_m	Function defining inequality constraint
g_h	Function defining equality constraint
m	Index of constraint-defining function

$\mathcal{S}$	Search space
$\mathcal{F}$	Feasible search space
$\mathcal{O}$	Objective space
R	One-dimensional real space
R^{nx}	n_x-dimensional real space
P^*	Pareto-optimal Set (POS)
PF^*	Approximated POF
t	Variable representing time
$\mathbf{x_I}$	Vector of decision variables
$\mathbf{x_{II}}$	Vector of decision variables
r	Point's radial distance from a fixed origin
θ	Polar angle of a point in a spherical coordinate system
γ	Azimuth angle of a point in a spherical coordinate system
π	Pi, which is approximately 3.14
f^L	Lower bound of function f
f^U	Upper bound of function f
τ	Iteration number
τ_t	Frequency of change
n_t	Severity of change
Ω	Approximated POF
POF^*	Approximated POF
P	Performance measure
u	Utility function
U	Set of utility function
O_w	Weak outperformance relation
O_s	Strong outperformance relation
O_c	Complete outperformance relation
ND	Non-domination relation

HV	Hypervolume
HVR	Hypervolume ratio

B.2 Chapter 3: Nature-Inspired Algorithms

N	Population size
$\mathcal{O}$	Big-O notation
t	Variable representing Time
$x_{best}(t)$	Best vector in a population at time t used by DE algorithm
$x_i(t)$	Particle's position at time t
$v_i(t)$	Particle's velocity at time t
$v_{ij}(t)$	Particle's velocity on jth dimension at time t
ω	Inertia weight
c_1	Acceleration coefficient for self-knowledge
r_1	Random number
c_2	Acceleration coefficient for socially acquired knowledge
r_2	Random number
$x_i^{pbest}(t)$	Personal best position of the particle at time t
$x_i^{nbest}(t)$	Global best position of the particle at time t
x_i^{min}	Particle's minimum position
x_i^{max}	Particle's maximum position
$U(0,1)$	Uniform random number distribution
$V_{max,j}$	Maximum velocity for the j-th particle
u_i	Trial vector in DE algorithm
β	Scaling factor used to amplify difference vectors in DE algorithm
γ_r^*	DE control parameter balances between exploration and exploitation
γ^*	DE control parameter balances between exploration and exploitation
J	Set of crossover points
n_s	Population size
p_r	DE crossover probability

B.3 Chapter 4: Decision-Making Essentials

D	Map of acts and states to outcomes
A	Set of acts
B	Set of states
O	Set of outcomes
A^*	Comparative matrix of a criterion
a_{ij}^*	Local priorities associated with a pair of acts or alternatives in $\dot{A}$
r_i^*	Sum of elements of $\dot{A}$ on row i
p_i^*	Priority of alternative i for a given criterion
w_j^*	Priority or weight associated with criterion j
p_{ij}^*	Priority of alternative i for criterion j
P_i^*	Priority of alternative i aggregated over all criteria
$\mathbf{a}$	Set of acts or alternatives
$\mathbf{f}$	Set of objective functions
a_i	i-th act in a set of acts
d_k	Difference between pairs of acts for k-th criterion
p^{**}	Preference threshold
q^{**}	Indifference threshold
P_k	Unicriterion preference degree
π^*	Multi criteria preference degree
ϕ^+	Positive preference flow
ϕ^-	Negative preference flow
ϕ	Net preference flow
n	Number of alternatives or acts
q	Number of objectives or criteria
x_{ij}^*	Performance of alternative i for criterion j
r_{ij}^*	Normalized performance of alternative i for criterion j
v_{ij}^*	Weight normalized performance of alternative i for criterion j

A^+	Set of ideal alternatives
A^-	Set of anti-ideal alternatives
d_a^+	Alternative's distance from ideal alternatives over all criteria
d_a^-	Alternative's distance from anti-ideal alternatives over all criteria
C_a	Alternative's relative closeness coefficient

B.4 Chapter 5: Differential Evolution Algorithm I

β	Scaling factor used to amplify difference vectors in DE algorithm
p_r	DE crossover probability
τ_t	Frequency of change
n_t	Severity of change
P_{gen}	Population in a generation
P	Population of vectors
F	Multi-objective function
V	Set of vectors
t	Variable representing Time
v	Vector
$v^{'}$	Trial vector
$v^{''}$	Child vector
P_t	Population at time t
POF_t	POF at time t
POS_t	POS at time t
$c(f(x))$	Count of function evaluations per iteration
$\sigma(runs)$	Number of runs per configuration

B.5 Chapter 6: Differential Evolution Algorithm II

τ_t	Frequency of change

n_t	Severity of change
t	Variable representing Time
$c(f(x))$	Count of function evaluations per iteration
$\sigma(runs)$	Number of runs per configuration

B.6 Chapter 7: Decision-maker's Preference Driven Optimization Problem

F	Multi-objective function
Ω_x	Decision variable space
Ω_t	Time variable space
O	Objective space
τ	Iteration number
τ_t	Frequency of change
n_t	Severity of change
t	Variable representing Time
$\Re^n$	N-dimensional real space
$\Re$	Real line
$\Re^2$	2-dimensional real space
$\Re^3$	3-dimensional real space
f_i	Single-objective function
z	Objective vector
d	Distance measure
p	Center of a circle or sphere
r	Radius of a circle or sphere
λ	Penalty control parameter
N	Set of natural numbers
Z	Set of integers

β	Scaling factor used to amplify difference vectors in DE algorithm
POF_t	POF at time t
POS_t	POS at time t
$c(f(x))$	Count of function evaluations per iteration
$\sigma(runs)$	Number of runs per configuration

Appendix C

Derived Publications

This appendix lists the publications that are derived from this dissertation.

- Adekoya A.R, Helbig M. A Differential Evolution Algorithm for Dynamic Multi-Objective Optimization. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). ; 2017:1-10.
- Adekoya A.R. Helbig M. Decision-Maker's Preference Driven Multi-Objective Optimization in a Dynamic Environment. In: Journal of European Operational Research(Submission in progress).

Appendix D

Experiment III: Detailed Results

Table D.1: Third Experimental Study: Detailed Results - 1

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	acc	0.71	0.736	**0.6633**
fda1	10	4	acc	**0.5959**	0.6533	0.7089
fda1	10	4	acc	**0.6236**	0.6581	0.6701
fda1	10	4	acc	0.6355	**0.5756**	0.6087
fda1	10	5	acc	**0.6645**	0.6721	0.7355
fda1	10	5	acc	0.6443	0.6303	**0.6217**
fda1	10	5	acc	**0.6204**	0.663	0.6485
fda1	10	5	acc	0.5634	**0.5295**	0.5401
fda1	10	2	acc	0.6993	**0.5643**	0.6647
fda1	10	2	acc	**0.5952**	0.6773	0.6496
fda1	10	2	acc	**0.6311**	0.6796	0.6332
fda1	10	2	acc	**0.5669**	0.5971	0.5843
fda1	1	4	acc	0.9975	0.9936	**0.9823**
fda1	1	4	acc	0.7261	**0.6684**	0.7465
fda1	1	4	acc	**0.9671**	0.9961	0.9836
fda1	1	4	acc	**0.6606**	0.6645	0.7412
fda1	1	5	acc	0.9644	**0.7128**	0.7965
fda1	1	5	acc	**0.7928**	0.8884	0.825
fda1	1	5	acc	0.9139	**0.6744**	0.829
fda1	1	5	acc	0.77	0.8612	**0.7521**
fda1	1	2	acc	0.958	0.937	**0.8967**
fda1	1	2	acc	**0.8832**	0.9017	0.884
fda1	1	2	acc	0.9587	0.9519	**0.8972**
fda1	1	2	acc	0.8834	**0.7915**	0.8299

Table D.2: Third Experimental Study: Detailed Results - 2

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	stab	**0**	**0**	**0**
fda1	10	4	stab	**0.1913**	0.2382	0.2055
fda1	10	4	stab	**0.1583**	0.1845	0.1725
fda1	10	4	stab	0.2044	**0.1675**	0.1932
fda1	10	5	stab	**0**	**0**	**0**
fda1	10	5	stab	0.1826	0.1801	**0.1188**
fda1	10	5	stab	0.2357	0.1388	**0.132**
fda1	10	5	stab	0.2073	0.2242	**0.1906**
fda1	10	2	stab	**0**	**0**	**0**
fda1	10	2	stab	**0.1347**	0.1697	0.15
fda1	10	2	stab	**0.1091**	0.1621	0.1531
fda1	10	2	stab	0.2105	**0.1919**	0.1923
fda1	1	4	stab	**0**	**0**	**0**
fda1	1	4	stab	**0**	2e-04	**0**
fda1	1	4	stab	0.0329	**0.0039**	0.0164
fda1	1	4	stab	4e-04	**0**	**0**
fda1	1	5	stab	**0**	**0**	**0**
fda1	1	5	stab	**0**	**0**	**0**
fda1	1	5	stab	**0.0845**	0.278	0.103
fda1	1	5	stab	**0**	**0**	0.0012
fda1	1	2	stab	**0**	**0**	**0**
fda1	1	2	stab	**0**	**0**	**0**
fda1	1	2	stab	0.041	**0.0252**	0.0882
fda1	1	2	stab	**0**	**0**	**0**

Table D.3: Third Experimental Study: Detailed Results - 3

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	hvr	1.74	1.7865	**1.8379**
fda1	10	4	hvr	1.5609	1.7001	**1.9762**
fda1	10	4	hvr	**1.7016**	1.6661	1.5562
fda1	10	4	hvr	1.7977	**2.0063**	1.5598
fda1	10	5	hvr	1.6588	1.592	**1.9064**
fda1	10	5	hvr	**1.7341**	1.7048	1.5894
fda1	10	5	hvr	1.6116	**1.7846**	1.7651
fda1	10	5	hvr	1.4383	**1.733**	1.624
fda1	10	2	hvr	**1.7493**	1.5363	1.7156
fda1	10	2	hvr	**1.6243**	1.5522	1.5465
fda1	10	2	hvr	**1.596**	1.5437	1.4597
fda1	10	2	hvr	**1.6373**	1.5162	1.4294
fda1	1	4	hvr	1.4955	1.5851	**1.7424**
fda1	1	4	hvr	2.9606	2.7286	**2.9838**
fda1	1	4	hvr	1.4713	1.6797	**1.7775**
fda1	1	4	hvr	2.7961	2.7168	**2.898**
fda1	1	5	hvr	**2.0055**	1.4014	1.4275
fda1	1	5	hvr	3.704	**4.3578**	3.6041
fda1	1	5	hvr	**1.8095**	1.2873	1.2637
fda1	1	5	hvr	3.6045	**4.2051**	3.5012
fda1	1	2	hvr	2.0333	1.7847	**2.0553**
fda1	1	2	hvr	**4.268**	4.2409	4.0464
fda1	1	2	hvr	1.9415	1.7069	**2.4759**
fda1	1	2	hvr	**3.9993**	3.5754	3.7061

Table D.4: Third Experimental Study: Detailed Results - 4

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	react	**13**	**13**	**13**
fda1	10	4	react	**9**	**9**	**9**
fda1	10	4	react	**5**	**5**	**5**
fda1	10	4	react	**1**	**1**	**1**
fda1	10	5	react	**16**	**16**	**16**
fda1	10	5	react	**11**	**11**	**11**
fda1	10	5	react	**6**	**6**	**6**
fda1	10	5	react	**1**	**1**	**1**
fda1	10	2	react	**7**	**7**	**7**
fda1	10	2	react	**5**	**5**	**5**
fda1	10	2	react	**3**	**3**	**3**
fda1	10	2	react	**1**	**1**	**1**
fda1	1	4	react	**13**	**13**	**13**
fda1	1	4	react	8.3	**8.0667**	8.4
fda1	1	4	react	**5**	**5**	**5**
fda1	1	4	react	**1**	**1**	**1**
fda1	1	5	react	12.5	**7.5**	9.5
fda1	1	5	react	10.4	10.6333	**10.3667**
fda1	1	5	react	4.8333	**3**	3.1667
fda1	1	5	react	**1**	**1**	**1**
fda1	1	2	react	4.2	3.5	**2.8**
fda1	1	2	react	4.6333	**4.5667**	**4.5667**
fda1	1	2	react	2.8	2.6	**2.3333**
fda1	1	2	react	**1**	**1**	**1**

Table D.5: Third Experimental Study: Detailed Results - 5

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	nNVD	99.4333	98.1333	**99.8**
fda1	10	4	nNVD	**99.5**	96.1667	96.0667
fda1	10	4	nNVD	**99.5**	94.3667	95.6
fda1	10	4	nNVD	**99.0333**	96.3	96.8333
fda1	10	5	nNVD	**100**	**100**	**100**
fda1	10	5	nNVD	**100**	**100**	**100**
fda1	10	5	nNVD	**100**	99.9667	**100**
fda1	10	5	nNVD	**100**	**100**	**100**
fda1	10	2	nNVD	64.9667	64.9	**67.1333**
fda1	10	2	nNVD	**67.4333**	58.6667	57.6667
fda1	10	2	nNVD	**67.5667**	58.7	56
fda1	10	2	nNVD	**67.7**	57.4667	57.7333
fda1	1	4	nNVD	26.2667	**27.6333**	25.6333
fda1	1	4	nNVD	26.2333	**26.5667**	21.6333
fda1	1	4	nNVD	**26.1667**	23.1333	22.7
fda1	1	4	nNVD	25	25.3667	**25.6**
fda1	1	5	nNVD	**100**	**100**	**100**
fda1	1	5	nNVD	71.4333	99.0333	**100**
fda1	1	5	nNVD	**100**	98.9333	99.3
fda1	1	5	nNVD	76.2667	**98.1667**	95.7667
fda1	1	2	nNVD	66.0333	65.7	**66.2**
fda1	1	2	nNVD	31.7	30.2333	**40.3**
fda1	1	2	nNVD	**62.6**	43.2333	43.4333
fda1	1	2	nNVD	**31.4667**	31.3667	28.8667

Table D.6: Third Experimental Study: Detailed Results - 6

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	sNVD	0.0297	**0.0301**	0.0296
fda1	10	4	sNVD	0.0298	**0.0308**	**0.0308**
fda1	10	4	sNVD	0.0298	**0.0314**	0.031
fda1	10	4	sNVD	0.0299	**0.0308**	0.0306
fda1	10	5	sNVD	**0.0295**	**0.0295**	**0.0295**
fda1	10	5	sNVD	**0.0295**	**0.0295**	**0.0295**
fda1	10	5	sNVD	0.0295	0.0295	**0.0296**
fda1	10	5	sNVD	**0.0295**	**0.0295**	**0.0295**
fda1	10	2	sNVD	0.0457	**0.0458**	0.0441
fda1	10	2	sNVD	0.0442	0.0506	**0.0514**
fda1	10	2	sNVD	0.044	0.0505	**0.0532**
fda1	10	2	sNVD	0.0438	**0.0518**	0.0512
fda1	1	4	sNVD	0.0671	0.0648	**0.0686**
fda1	1	4	sNVD	0.0719	0.0777	**0.0853**
fda1	1	4	sNVD	0.0675	0.0773	**0.0784**
fda1	1	4	sNVD	0.0735	0.0756	**0.0769**
fda1	1	5	sNVD	**0.0296**	0.0295	0.0295
fda1	1	5	sNVD	**0.0422**	0.0301	0.0296
fda1	1	5	sNVD	0.0296	**0.0299**	0.0298
fda1	1	5	sNVD	**0.0394**	0.0303	0.0291
fda1	1	2	sNVD	0.0448	**0.0453**	0.0449
fda1	1	2	sNVD	0.0973	**0.1029**	0.0726
fda1	1	2	sNVD	0.0478	**0.069**	0.067
fda1	1	2	sNVD	0.0981	0.1001	**0.1089**

Table D.7: Third Experimental Study: Detailed Results - 7

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	nVD	**0**	**0**	**0**
fda1	10	4	nVD	**0**	**0**	**0**
fda1	10	4	nVD	**0**	**0**	**0**
fda1	10	4	nVD	**0**	**0**	**0**
fda1	10	5	nVD	**0**	**0**	**0**
fda1	10	5	nVD	**0**	**0**	**0**
fda1	10	5	nVD	**0**	**0**	**0**
fda1	10	5	nVD	**0**	**0**	**0**
fda1	10	2	nVD	**0**	**0**	**0**
fda1	10	2	nVD	**0**	**0**	**0**
fda1	10	2	nVD	**0**	**0**	**0**
fda1	10	2	nVD	**0**	**0**	**0**
fda1	1	4	nVD	**0**	**0**	**0**
fda1	1	4	nVD	**0**	**0**	**0**
fda1	1	4	nVD	**0**	**0**	**0**
fda1	1	4	nVD	0.0667	**0**	**0**
fda1	1	5	nVD	**0**	**0**	**0**
fda1	1	5	nVD	**0**	**0**	**0**
fda1	1	5	nVD	**0**	**0**	**0**
fda1	1	5	nVD	**0**	**0**	**0**
fda1	1	2	nVD	**0**	**0**	**0**
fda1	1	2	nVD	**0**	**0**	**0**
fda1	1	2	nVD	**0**	**0**	**0**
fda1	1	2	nVD	**0**	**0**	**0**

Table D.8: Third Experimental Study: Detailed Results - 8

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda1	10	4	dVD	**0**	**0**	**0**
fda1	10	4	dVD	**0**	**0**	**0**
fda1	10	4	dVD	**0**	**0**	**0**
fda1	10	4	dVD	**0**	**0**	**0**
fda1	10	5	dVD	**0**	**0**	**0**
fda1	10	5	dVD	**0**	**0**	**0**
fda1	10	5	dVD	**0**	**0**	**0**
fda1	10	5	dVD	**0**	**0**	**0**
fda1	10	2	dVD	**0**	**0**	**0**
fda1	10	2	dVD	**0**	**0**	**0**
fda1	10	2	dVD	**0**	**0**	**0**
fda1	10	2	dVD	**0**	**0**	**0**
fda1	1	4	dVD	**0**	**0**	**0**
fda1	1	4	dVD	**0**	**0**	**0**
fda1	1	4	dVD	**0**	**0**	**0**
fda1	1	4	dVD	0.0667	**0**	**0**
fda1	1	5	dVD	**0**	**0**	**0**
fda1	1	5	dVD	**0**	**0**	**0**
fda1	1	5	dVD	**0**	**0**	**0**
fda1	1	5	dVD	**0**	**0**	**0**
fda1	1	2	dVD	**0**	**0**	**0**
fda1	1	2	dVD	**0**	**0**	**0**
fda1	1	2	dVD	**0**	**0**	**0**
fda1	1	2	dVD	**0**	**0**	**0**

Table D.9: Third Experimental Study: Detailed Results - 9

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	acc	0.6404	0.5962	**0.538**
fda5	10	4	acc	**0.6751**	0.7731	0.7557
fda5	10	4	acc	**0.4445**	0.4619	0.6266
fda5	10	4	acc	0.2282	0.227	**0.1946**
fda5	10	5	acc	**0.2683**	0.408	0.8316
fda5	10	5	acc	**0.4298**	0.6803	0.857
fda5	10	5	acc	**0.514**	0.6998	0.8183
fda5	10	5	acc	**0.5382**	0.6913	0.7904
fda5	10	2	acc	**0.6239**	0.6727	0.9835
fda5	10	2	acc	**0.8799**	0.8925	0.9526
fda5	10	2	acc	**0.6285**	0.6549	0.9234
fda5	10	2	acc	**0.4285**	0.432	0.9186
fda5	1	4	acc	**0.9984**	1	1
fda5	1	4	acc	0.9986	**0.9955**	0.997
fda5	1	4	acc	1	**0.9991**	**0.9991**
fda5	1	4	acc	0.9996	**0.9945**	0.9956
fda5	1	5	acc	1	0.9993	**0.9958**
fda5	1	5	acc	0.9953	0.9915	**0.9746**
fda5	1	5	acc	1	**0.9971**	0.9976
fda5	1	5	acc	0.9963	0.9962	**0.9793**
fda5	1	2	acc	1	0.9993	**0.9948**
fda5	1	2	acc	0.9963	0.999	**0.9781**
fda5	1	2	acc	1	0.9993	**0.998**
fda5	1	2	acc	1	0.9987	**0.9853**

Table D.10: Third Experimental Study: Detailed Results - 10

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	stab	**0**	**0**	**0**
fda5	10	4	stab	0.1793	0.1457	**0.0884**
fda5	10	4	stab	0.4251	0.4261	**0.2747**
fda5	10	4	stab	0.4743	0.53	**0.4098**
fda5	10	5	stab	**0**	**0**	**0**
fda5	10	5	stab	0.1992	0.1543	**0.0168**
fda5	10	5	stab	0.2266	0.11	**0.0883**
fda5	10	5	stab	0.2461	0.1599	**0.1331**
fda5	10	2	stab	**0**	**0**	**0**
fda5	10	2	stab	0.0985	0.086	**0**
fda5	10	2	stab	0.3367	0.2923	**0.0277**
fda5	10	2	stab	0.4402	0.4291	**0.0455**
fda5	1	4	stab	**0**	**0**	**0**
fda5	1	4	stab	**0.0014**	0.0045	0.003
fda5	1	4	stab	**0**	9e-04	9e-04
fda5	1	4	stab	**4e-04**	0.0055	0.0044
fda5	1	5	stab	**0**	**0**	**0**
fda5	1	5	stab	**0.0047**	0.0085	0.0254
fda5	1	5	stab	**0**	0.0029	0.0024
fda5	1	5	stab	**0.0037**	0.0038	0.0207
fda5	1	2	stab	**0**	**0**	**0**
fda5	1	2	stab	0.0037	**0.001**	0.0219
fda5	1	2	stab	**0**	7e-04	0.002
fda5	1	2	stab	**0**	0.0013	0.0147

Table D.11: Third Experimental Study: Detailed Results - 11

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	hvr	**1.6654**	1.3814	1.2592
fda5	10	4	hvr	**2.4693**	1.711	1.5734
fda5	10	4	hvr	**3.2302**	1.8911	1.64
fda5	10	4	hvr	**2.0206**	2.0017	1.7762
fda5	10	5	hvr	2.1537	**2.7315**	2.4955
fda5	10	5	hvr	2.2018	**2.6023**	1.9044
fda5	10	5	hvr	2.267	**2.3623**	2.0773
fda5	10	5	hvr	2.1636	**2.4252**	1.8647
fda5	10	2	hvr	1.5414	**1.6826**	1.2248
fda5	10	2	hvr	**3.3034**	3.2674	1.4156
fda5	10	2	hvr	4.9233	**5.1433**	1.1358
fda5	10	2	hvr	3.5145	**4.0314**	1.27
fda5	1	4	hvr	**4.4565**	4.0032	3.8261
fda5	1	4	hvr	2.4313	**2.5719**	2.2178
fda5	1	4	hvr	**4.7446**	3.4347	3.5272
fda5	1	4	hvr	2.1979	**2.7897**	2.2656
fda5	1	5	hvr	**2.9987**	2.6455	1.9198
fda5	1	5	hvr	**2.6475**	2.1308	1.0979
fda5	1	5	hvr	**3.088**	2.8665	2.5523
fda5	1	5	hvr	**2.5349**	2.3717	1.346
fda5	1	2	hvr	3.0785	**3.5184**	1.8674
fda5	1	2	hvr	2.2679	**3.1541**	1.0209
fda5	1	2	hvr	2.9413	**3.3106**	3.2396
fda5	1	2	hvr	3.2166	**3.2704**	2.2137

Table D.12: Third Experimental Study: Detailed Results - 12

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	react	12.9667	12.9667	**12.5**
fda5	10	4	react	8.9333	**8.8333**	**8.8333**
fda5	10	4	react	5	**4.9667**	5
fda5	10	4	react	**1**	**1**	**1**
fda5	10	5	react	16	16	**15.9333**
fda5	10	5	react	11	**10.8**	10.9333
fda5	10	5	react	**6**	**6**	**6**
fda5	10	5	react	**1**	**1**	**1**
fda5	10	2	react	**6.9**	7	7
fda5	10	2	react	5	**4.9667**	5
fda5	10	2	react	**3**	**3**	**3**
fda5	10	2	react	**1**	**1**	**1**
fda5	1	4	react	**12.6**	13	13
fda5	1	4	react	7.7667	**7.3**	7.5333
fda5	1	4	react	5	**4.8667**	**4.8667**
fda5	1	4	react	**1**	**1**	**1**
fda5	1	5	react	16	15	**11.5**
fda5	1	5	react	8.2	7.3	**2.8**
fda5	1	5	react	6	**5.3333**	5.5
fda5	1	5	react	**1**	**1**	**1**
fda5	1	2	react	4	3.9	**3**
fda5	1	2	react	3.3	3.7	**1.2**
fda5	1	2	react	3	2.9333	**2.6667**
fda5	1	2	react	**1**	**1**	**1**

Table D.13: Third Experimental Study: Detailed Results - 13

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	nNVD	39.5	**50.0333**	45.3
fda5	10	4	nNVD	57.1667	67.3333	**68.4333**
fda5	10	4	nNVD	63.5	**74.6333**	63.1333
fda5	10	4	nNVD	71.9	**72.9333**	58.8667
fda5	10	5	nNVD	42	**49.9667**	27.4
fda5	10	5	nNVD	**72.0667**	70.2	35.0667
fda5	10	5	nNVD	**74.4333**	65.4667	36.4
fda5	10	5	nNVD	**79.0667**	60.1667	34.0333
fda5	10	2	nNVD	**12.2667**	11.8333	2.9333
fda5	10	2	nNVD	**13.1667**	9.1	1.5667
fda5	10	2	nNVD	**19.3**	9.8	2.6333
fda5	10	2	nNVD	**41.5667**	33.8	2.6
fda5	1	4	nNVD	**12.9667**	12.7	9.6333
fda5	1	4	nNVD	**32.4667**	18.6667	12.3667
fda5	1	4	nNVD	**13.6667**	12.9667	9.3667
fda5	1	4	nNVD	**29.2333**	28.9	16.5333
fda5	1	5	nNVD	19.4	**23.2**	21.7333
fda5	1	5	nNVD	35.8333	34.2	**67.6333**
fda5	1	5	nNVD	21.0333	**23.4333**	21.0333
fda5	1	5	nNVD	39.1	33.6333	**62.8333**
fda5	1	2	nNVD	22.1667	23.6333	**26.4333**
fda5	1	2	nNVD	**20.8667**	17.8667	6.6
fda5	1	2	nNVD	**18.8**	17.5	17.4333
fda5	1	2	nNVD	**20.2667**	18.0333	11.4667

Table D.14: Third Experimental Study: Detailed Results - 14

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	sNVD	**0.1175**	0.0966	0.1008
fda5	10	4	sNVD	**0.0906**	0.0791	0.0726
fda5	10	4	sNVD	**0.11**	0.0915	0.0962
fda5	10	4	sNVD	0.1237	0.1122	**0.1245**
fda5	10	5	sNVD	**0.1652**	0.1327	0.101
fda5	10	5	sNVD	**0.1105**	0.1015	0.0915
fda5	10	5	sNVD	**0.1089**	0.1005	0.094
fda5	10	5	sNVD	0.1017	**0.1079**	0.1059
fda5	10	2	sNVD	**0.2712**	0.2546	0.004
fda5	10	2	sNVD	0.1928	**0.291**	0.0032
fda5	10	2	sNVD	0.22	**0.3729**	5e-04
fda5	10	2	sNVD	0.1413	**0.1716**	0.0062
fda5	1	4	sNVD	0.1967	**0.2117**	0.0279
fda5	1	4	sNVD	**0.2258**	0.1087	0.0253
fda5	1	4	sNVD	**0.2683**	0.2654	0.0122
fda5	1	4	sNVD	**0.2238**	0.0853	0.0124
fda5	1	5	sNVD	**0.0897**	0.0592	0.0568
fda5	1	5	sNVD	**0.0946**	0.0687	0.0368
fda5	1	5	sNVD	**0.1374**	0.1011	0.0651
fda5	1	5	sNVD	**0.0757**	0.0693	0.0407
fda5	1	2	sNVD	**0.16**	0.1365	0.0837
fda5	1	2	sNVD	0.0726	**0.0964**	0.0577
fda5	1	2	sNVD	0.2013	**0.2357**	0.1891
fda5	1	2	sNVD	0.0617	**0.0747**	0.0391

Table D.15: Third Experimental Study: Detailed Results - 15

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	nVD	0.0333	**0**	**0**
fda5	10	4	nVD	**0**	**0**	**0**
fda5	10	4	nVD	**0**	**0**	**0**
fda5	10	4	nVD	**0**	**0**	**0**
fda5	10	5	nVD	**0**	**0**	**0**
fda5	10	5	nVD	**0**	**0**	**0**
fda5	10	5	nVD	**0**	**0**	**0**
fda5	10	5	nVD	**0**	**0**	**0**
fda5	10	2	nVD	**0**	**0**	**0**
fda5	10	2	nVD	**0**	**0**	**0**
fda5	10	2	nVD	0.0333	**0**	**0**
fda5	10	2	nVD	0.0333	**0**	**0**
fda5	1	4	nVD	0.0667	**0**	**0**
fda5	1	4	nVD	**0**	**0**	**0**
fda5	1	4	nVD	**0**	**0**	**0**
fda5	1	4	nVD	**0**	**0**	**0**
fda5	1	5	nVD	**0**	**0**	**0**
fda5	1	5	nVD	**0**	**0**	**0**
fda5	1	5	nVD	**0**	**0**	**0**
fda5	1	5	nVD	**0**	**0**	**0**
fda5	1	2	nVD	0.0333	**0**	**0**
fda5	1	2	nVD	**0**	**0**	**0**
fda5	1	2	nVD	**0**	**0**	**0**
fda5	1	2	nVD	**0**	**0**	**0**

Table D.16: Third Experimental Study: Detailed Results - 16

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
fda5	10	4	dVD	0.0333	**0**	**0**
fda5	10	4	dVD	**0**	**0**	**0**
fda5	10	4	dVD	**0**	**0**	**0**
fda5	10	4	dVD	**0**	**0**	**0**
fda5	10	5	dVD	**0**	**0**	**0**
fda5	10	5	dVD	**0**	**0**	**0**
fda5	10	5	dVD	**0**	**0**	**0**
fda5	10	5	dVD	**0**	**0**	**0**
fda5	10	2	dVD	**0**	**0**	**0**
fda5	10	2	dVD	**0**	**0**	**0**
fda5	10	2	dVD	0.0333	**0**	**0**
fda5	10	2	dVD	0.0333	**0**	**0**
fda5	1	4	dVD	0.0667	**0**	**0**
fda5	1	4	dVD	**0**	**0**	**0**
fda5	1	4	dVD	**0**	**0**	**0**
fda5	1	4	dVD	**0**	**0**	**0**
fda5	1	5	dVD	**0**	**0**	**0**
fda5	1	5	dVD	**0**	**0**	**0**
fda5	1	5	dVD	**0**	**0**	**0**
fda5	1	5	dVD	**0**	**0**	**0**
fda5	1	2	dVD	0.0333	**0**	**0**
fda5	1	2	dVD	**0**	**0**	**0**
fda5	1	2	dVD	**0**	**0**	**0**
fda5	1	2	dVD	**0**	**0**	**0**

Table D.17: Third Experimental Study: Detailed Results - 17

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	acc	0.87	**0.7811**	0.8707
dmop2	10	4	acc	**0.8284**	0.8287	0.8715
dmop2	10	4	acc	**0.8124**	0.8805	0.884
dmop2	10	4	acc	0.998	**0.9717**	0.9995
dmop2	10	5	acc	0.7858	**0.768**	0.9103
dmop2	10	5	acc	**0.6857**	0.7187	0.9217
dmop2	10	5	acc	0.8374	**0.7731**	0.908
dmop2	10	5	acc	0.9773	**0.9755**	0.9951
dmop2	10	2	acc	**0.8127**	0.849	0.8298
dmop2	10	2	acc	0.8232	0.8581	**0.7979**
dmop2	10	2	acc	0.9443	**0.8277**	0.8558
dmop2	10	2	acc	0.998	**0.9975**	0.9988
dmop2	1	4	acc	0.9266	0.3449	**0.3097**
dmop2	1	4	acc	0.9309	0.8833	**0.5509**
dmop2	1	4	acc	1	**0.3802**	0.6802
dmop2	1	4	acc	0.5236	**0.4389**	0.4764
dmop2	1	5	acc	0.9478	**0.2882**	0.9615
dmop2	1	5	acc	**0.9393**	0.9508	0.9574
dmop2	1	5	acc	0.9991	**0.3737**	0.9778
dmop2	1	5	acc	0.4861	**0.3884**	0.7584
dmop2	1	2	acc	0.9548	**0.7587**	0.9164
dmop2	1	2	acc	0.9833	**0.8949**	0.9059
dmop2	1	2	acc	0.9942	**0.8278**	0.9069
dmop2	1	2	acc	**0.4735**	0.5391	0.7011

Table D.18: Third Experimental Study: Detailed Results - 18

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	stab	**0**	**0**	**0**
dmop2	10	4	stab	**0**	0.0164	**0**
dmop2	10	4	stab	**0**	0.011	2e-04
dmop2	10	4	stab	6e-04	0.0165	**1e-04**
dmop2	10	5	stab	**0**	**0**	**0**
dmop2	10	5	stab	0.0099	0.0261	**0**
dmop2	10	5	stab	**0**	0.0159	**0**
dmop2	10	5	stab	0.0209	0.0226	**0.0036**
dmop2	10	2	stab	**0**	**0**	**0**
dmop2	10	2	stab	**0**	**0**	**0**
dmop2	10	2	stab	**0**	**0**	**0**
dmop2	10	2	stab	9e-04	3e-04	**2e-04**
dmop2	1	4	stab	**0**	**0**	**0**
dmop2	1	4	stab	**0.0224**	0.1167	0.4491
dmop2	1	4	stab	**0**	0.6198	0.3198
dmop2	1	4	stab	0.0073	**0.0045**	0.0396
dmop2	1	5	stab	**0**	**0**	**0**
dmop2	1	5	stab	**0.0177**	0.0492	0.0426
dmop2	1	5	stab	**9e-04**	0.6263	0.0222
dmop2	1	5	stab	**0.0103**	0.071	0.2271
dmop2	1	2	stab	**0**	**0**	**0**
dmop2	1	2	stab	**0.0059**	0.1051	0.0941
dmop2	1	2	stab	**0.0058**	0.1722	0.0931
dmop2	1	2	stab	**0.0169**	0.0625	0.2519

Table D.19: Third Experimental Study: Detailed Results - 19

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	hvr	**1.2872**	1.2594	1.2577
dmop2	10	4	hvr	1.1982	1.2284	**1.3295**
dmop2	10	4	hvr	**1.595**	1.5624	1.4643
dmop2	10	4	hvr	1.3038	**1.4336**	1.4014
dmop2	10	5	hvr	**1.9143**	1.8882	1.1107
dmop2	10	5	hvr	1.673	**1.7919**	1.1055
dmop2	10	5	hvr	**1.7423**	1.65	1.0242
dmop2	10	5	hvr	1.8847	**2.0522**	1.1175
dmop2	10	2	hvr	1.3575	**1.5524**	1.4858
dmop2	10	2	hvr	1.3303	1.4589	**1.4937**
dmop2	10	2	hvr	**1.7886**	1.3657	1.3977
dmop2	10	2	hvr	**1.5989**	1.593	1.4986
dmop2	1	4	hvr	1.5361	**2.0163**	1.6181
dmop2	1	4	hvr	2.4648	**9.2045**	4.37
dmop2	1	4	hvr	0.9826	6.7302	**10.8249**
dmop2	1	4	hvr	2.0165	**5.4407**	2.7435
dmop2	1	5	hvr	1.651	**4.8469**	0.9488
dmop2	1	5	hvr	2.1529	**14.7819**	0.9485
dmop2	1	5	hvr	0.9861	**6.5701**	0.9669
dmop2	1	5	hvr	1.7927	**6.4797**	0.7462
dmop2	1	2	hvr	1.9387	**7.6131**	1.4606
dmop2	1	2	hvr	2.3369	**5.3866**	1.5312
dmop2	1	2	hvr	0.9523	**10.4355**	2.5348
dmop2	1	2	hvr	**1.7974**	1.3771	0.7159

Table D.20: Third Experimental Study: Detailed Results - 20

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	react	12.8667	12.9667	**12.3667**
dmop2	10	4	react	**8.0333**	8.6333	8.6
dmop2	10	4	react	4.9667	**4.8333**	**4.8333**
dmop2	10	4	react	**1**	**1**	**1**
dmop2	10	5	react	**16**	**16**	**16**
dmop2	10	5	react	**11**	**11**	**11**
dmop2	10	5	react	6	**5.9333**	5.9667
dmop2	10	5	react	**1**	**1**	**1**
dmop2	10	2	react	**6.8**	6.9	**6.8**
dmop2	10	2	react	**4.8**	4.9333	4.8333
dmop2	10	2	react	2.9333	**2.8**	2.9
dmop2	10	2	react	**1**	**1**	**1**
dmop2	1	4	react	8.6	2.6	**1.4**
dmop2	1	4	react	7.8	5.2667	**2.3333**
dmop2	1	4	react	5	**2.4667**	4.6
dmop2	1	4	react	**1**	**1**	**1**
dmop2	1	5	react	12	**2.5**	12.5
dmop2	1	5	react	9.6333	**8.3333**	8.6667
dmop2	1	5	react	6	**2.8333**	6
dmop2	1	5	react	**1**	**1**	**1**
dmop2	1	2	react	3.3	**1.5**	2.9
dmop2	1	2	react	4.7667	3.1333	**2.8667**
dmop2	1	2	react	3	**2.9333**	3
dmop2	1	2	react	**1**	**1**	**1**

Table D.21: Third Experimental Study: Detailed Results - 21

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	nNVD	5.3667	5.8333	**6.4333**
dmop2	10	4	nNVD	**4.0667**	3.9333	3.6
dmop2	10	4	nNVD	**3.1333**	3	3.1
dmop2	10	4	nNVD	**4.3**	3.3333	3.0667
dmop2	10	5	nNVD	12.6	**13.3333**	3.8333
dmop2	10	5	nNVD	**10.4**	9.8667	3
dmop2	10	5	nNVD	**11**	9.3333	2.9667
dmop2	10	5	nNVD	**9.6667**	7.9333	2.5667
dmop2	10	2	nNVD	5.8333	5.8667	**6.5667**
dmop2	10	2	nNVD	**3.9**	3.1333	2.8667
dmop2	10	2	nNVD	**3.7333**	2.9333	2.8333
dmop2	10	2	nNVD	**4.6333**	3.2	3.0333
dmop2	1	4	nNVD	68.1667	91.5667	**94.4667**
dmop2	1	4	nNVD	49.3	24.8	**60.7**
dmop2	1	4	nNVD	50.8	**63.1667**	30.2333
dmop2	1	4	nNVD	0	**0.0333**	0
dmop2	1	5	nNVD	**81.0667**	75.4333	0.9333
dmop2	1	5	nNVD	**59.5333**	3.4333	1
dmop2	1	5	nNVD	60.6333	**66.3333**	0.9
dmop2	1	5	nNVD	**0**	**0**	**0**
dmop2	1	2	nNVD	31.4333	**32.0333**	5.8
dmop2	1	2	nNVD	**13.4333**	1.7	2.1
dmop2	1	2	nNVD	**19.9333**	12.5333	0.6333
dmop2	1	2	nNVD	**0**	**0**	**0**

Table D.22: Third Experimental Study: Detailed Results - 22

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	sNVD	**0.0597**	0.0511	0.0466
dmop2	10	4	sNVD	0.0698	**0.0835**	0.0716
dmop2	10	4	sNVD	0.0775	**0.1051**	0.0987
dmop2	10	4	sNVD	0.0721	0.0806	**0.0951**
dmop2	10	5	sNVD	**0.13**	0.1264	0.0291
dmop2	10	5	sNVD	0.1548	**0.1585**	0.0395
dmop2	10	5	sNVD	0.1465	**0.171**	0.0382
dmop2	10	5	sNVD	0.174	**0.2079**	0.0364
dmop2	10	2	sNVD	0.115	**0.118**	0.1052
dmop2	10	2	sNVD	0.1543	0.1751	**0.231**
dmop2	10	2	sNVD	0.1752	0.1745	**0.2025**
dmop2	10	2	sNVD	0.1419	0.1825	**0.2077**
dmop2	1	4	sNVD	**0.0441**	0.0326	0.0319
dmop2	1	4	sNVD	0.0633	0.1534	**0.1541**
dmop2	1	4	sNVD	0.0585	0.066	**0.1982**
dmop2	1	4	sNVD	**0**	**0**	**0**
dmop2	1	5	sNVD	0.0366	**0.0596**	0
dmop2	1	5	sNVD	0.0519	**0.0929**	0
dmop2	1	5	sNVD	0.0492	**0.0532**	0
dmop2	1	5	sNVD	**0**	**0**	**0**
dmop2	1	2	sNVD	0.1038	**0.1133**	0.006
dmop2	1	2	sNVD	0.229	**0.357**	0.0454
dmop2	1	2	sNVD	0.1599	**0.5381**	0.0247
dmop2	1	2	sNVD	**0**	**0**	**0**

Table D.23: Third Experimental Study: Detailed Results - 23

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	nVD	0.0667	**0**	**0**
dmop2	10	4	nVD	0.0333	**0**	**0**
dmop2	10	4	nVD	**0**	**0**	**0**
dmop2	10	4	nVD	**0**	**0**	**0**
dmop2	10	5	nVD	0.0333	**0**	**0**
dmop2	10	5	nVD	**0**	**0**	**0**
dmop2	10	5	nVD	0.1333	**0**	**0**
dmop2	10	5	nVD	0.0667	**0**	**0**
dmop2	10	2	nVD	**0**	**0**	**0**
dmop2	10	2	nVD	**0**	**0**	**0**
dmop2	10	2	nVD	0.0333	**0**	**0**
dmop2	10	2	nVD	0.0333	**0**	**0**
dmop2	1	4	nVD	**0**	**0**	**0**
dmop2	1	4	nVD	**0**	23.3333	**0**
dmop2	1	4	nVD	**0**	**0**	3.3333
dmop2	1	4	nVD	**1**	96.6667	100
dmop2	1	5	nVD	**0**	6.6667	6.6667
dmop2	1	5	nVD	**0**	66.6667	**0**
dmop2	1	5	nVD	**0**	13.3333	10
dmop2	1	5	nVD	**1**	100	100
dmop2	1	2	nVD	**0**	**0**	6.6667
dmop2	1	2	nVD	**0**	26.7333	1.7
dmop2	1	2	nVD	**0**	**0**	24.1667
dmop2	1	2	nVD	**1**	62.4667	57.6333

Table D.24: Third Experimental Study: Detailed Results - 24

DMOOP	n_t	τ_t	PM	ALG:1	ALG:2	ALG:3
dmop2	10	4	dVD	0.0667	**0**	**0**
dmop2	10	4	dVD	0.0333	**0**	**0**
dmop2	10	4	dVD	**0**	**0**	**0**
dmop2	10	4	dVD	**0**	**0**	**0**
dmop2	10	5	dVD	0.0333	**0**	**0**
dmop2	10	5	dVD	**0**	**0**	**0**
dmop2	10	5	dVD	0.1333	**0**	**0**
dmop2	10	5	dVD	0.0667	**0**	**0**
dmop2	10	2	dVD	**0**	**0**	**0**
dmop2	10	2	dVD	**0**	**0**	**0**
dmop2	10	2	dVD	0.0333	**0**	**0**
dmop2	10	2	dVD	0.0333	**0**	**0**
dmop2	1	4	dVD	**0**	**0**	**0**
dmop2	1	4	dVD	**0**	8.0838	**0**
dmop2	1	4	dVD	**0**	**0**	1.1525
dmop2	1	4	dVD	78.3002	**64.215**	85.5742
dmop2	1	5	dVD	**0**	1.0863	1.0156
dmop2	1	5	dVD	**0**	25.4565	**0**
dmop2	1	5	dVD	**0**	2.597	1.7496
dmop2	1	5	dVD	83.0633	**65.1873**	93.9007
dmop2	1	2	dVD	**0**	**0**	1.11
dmop2	1	2	dVD	**0**	9.0173	0.4143
dmop2	1	2	dVD	**0**	**0**	7.6244
dmop2	1	2	dVD	**85.1725**	90.4603	93.0091