Multi-Objective Particle Swarm Optimisation: Methods and Applications

Noura Al Moubayed

A Thesis submitted in partial fulfilment of the
requirements of the
Robert Gordon University
for the degree of Doctor of Philosophy

February, 2014

Abstract

Solving real life optimisation problems is a challenging engineering venture. Since the early days of research on optimisation it was realised that many problems do not simply have one optimisation objective. This led to the development of multi-objective optimizers that try to look at the optimisation problem from different points of view and reach a set of compromised solutions among the different objectives. The presented research brings together recent advances in the field of multi-objective optimisation and particle swarm optimisation raising several challenges. This is tackled from different aspects including the proposal of new archiving techniques to developing new methods and quality measures. Smart Multi-objective Particle Swarm Optimisation based on Decomposition (SDMOPSO) is first proposed to incorporate multi-objective problem decomposition techniques with PSO. A novel archiving technique is developed using a clustering based mapping approach between the objective and solution spaces and is applied to general multi-objective optimizers. D^2MOPSO is introduced as a new MOPSO that uses problem decomposition and a new archive utilising dominance based mapping between objective and solution spaces. Finally the thesis presents a novel multi-objective quality measure that uses mutual information to compare among solutions generated by different algorithms. The contributions are all tested on standard test suits and are used to solve two real-life problems: a) Channel selection for Brain-Computer Interfaces, and b) Effective cancer chemotherapy treatments. The two problems are real challenges in the two respective fields. Two different modelling approaches of the channel selection problem are presented: one is based on binary representation of the channels, while the other is continuous in a projected space of the channel locations. The results are very competitive with the commonly used methods.

List of Publications

- N. Al Moubayed, A. Petrovski and J. McCall, D 2 MOPSO: Multi-Objective Particle Swarm Optimizer Based on Decomposition and Dominance, Evolutionary Computation, MIT Press.
- N. Al Moubayed, A. Petrovski and J. McCall, Mutual information for performance assessment of multi objective optimisers: Preliminary results, The 14th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'2013).
- N. Al Moubayed, A. Petrovski and J. McCall, D 2 MOPSO: Multi-Objective Particle Swarm Optimizer Based on Decomposition and Dominance, Evolutionary Computation in Combinatorial Optimization (EvoCop 2012) Volume 7245 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp 75-86.
- N. Al Moubayed, B. Awwad Shiekh Hasan, J. Q. Gan, A. Petrovski and J. McCall, Continuous Presentation for Multi-Objective Channel Selection in Brain Computer Interfaces, proceedings of the World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, IEEE.
- N. Al Moubayed, A. Petrovski and J. McCall, Clustering-Based Leaders Selection in Multi-Objective Particle Swarm Optimisation, Intelligent Data Engineering and Automated Learning (IDEAL 2011), Volume 6936 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 100-107.
- N. Al Moubayed, A. Petrovski and J. McCall, Clustering based Framework for Leaders Selection in Multi-Objective Evolutionary Algorithms, Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO 2011), Dublin, Ireland, ACM, pp. 96-96.

- N. Al Moubayed, A. Petrovski and J. McCall, Multi-Objective Optimisation of Cancer Chemotherapy using Smart PSO with Decomposition, In 3rd IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making in conjunction with IEEE Symposium Series on Computational Intelligence (SSCI 2011), April 2011, Paris, France, IEEE.

- N. Al Moubayed, A. Petrovski and J. McCall, A Novel Smart Multi-Objective Particle Swarm Optimisation using Decomposition, In Parallel Problem Solving from Nature (PPSN XI), 2010, volume 6239 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 1-10.

- N. Al Moubayed, B. Awwad Shiekh Hasan , J. Q. Gan, A. Petrovski and J. McCall, Binary-SDMOPSO and its Application in Channel Selection for Brain-Computer Interfaces, In 10th Annual Workshop on Computational Intelligence (UKCI 2010), September 2010, Colchester, UK, IEEE.

Acknowledgements

I would like to thank my PhD advisors, Doctor Andrei Petrovski and Professor John McCall, for supporting me during these past four years. Andrei was always supportive and has given me the freedom to take decisions and follow my research interests. He has always given me his invaluable advice and opinion. Without his support and encouragement I would not have achieved what I have achieved and I definitely would not be able to publish as much. John is a very sweet person, his kindness will make you instantly love him and never forget him. He is very smart, funny, friendly and has a real Scottish spirit. I enjoyed every single discussion and learnt from his great experience.

I will forever be thankful to my beloved husband who was and still is a true and great supporter. Bashar, thank you for standing by me, being positive at all times and most of all thanks for being patient with my, sometimes, irritable mood and stubbornness, and giving me all the love and care in the world. I love you loads and all the words in the world cannot express my appreciation. you are the meaning of my life and with you I am not afraid of anything, The outcome of my PhD is not only a thesis and some papers but also an awesome baby girl. Julie, you made my life bright and beautiful, you are my sunshine. I cannot be anything but happy and excited when you are around.

I especially thank my mom, dad, and brothers. they have never stopped supporting me for a second. They put their faith in me and gave me all the support in the world. Dad and mum, you are great parents and I would not imagine my life without you both. Samer, thank you for your support, advice and funny jokes, thanks for being so loving and caring. Hsnee, my

little angel, thanks for your care, love and amazing music, you rocked my life dude.

I thank my lovely dog, Lassie, who kept my feet warm while writing up and accompanied me in my walks and breaks. Lassie you are a piece of my heart.

I dedicate this thesis to my family, my husband, Bashar, my beautiful baby, Julie and my dog, Lassie for their great support and love. I love you all from all of my heart.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Human beings, viewed as behaving systems, are quite simple. The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves."

-Herbert A. Simon, The Sciences of the Artificial

1.1 Motivation

In the late 1980s, *Eshel Ben-Jacob* began to study the self-organization of bacteria in the hope of understanding more complex biological systems. He developed new pattern-forming bacteria species, **Paenibacillus vortex** and **Paenibacillus dendritiformis**. **P. dendritiformis** revealed an intriguing collective ability, which is viewed as a precursor of collaborative intelligence, the ability to switch between different morphotypes to better adapt with the environment (Ben-Jacob (2003)). Scientists have only recently started to understand, how bacteria can quickly adapt to changes in the environment, distribute tasks, learn from experience, prepare for the future and make decisions. Bacteria in a colony, numbering many times the population on Earth, exchange chemical messages to synchronize their behavior.

Ants were first characterized by entomologist **W. M. Wheeler** as cells of a single superorganism . What appears to be independent individuals can actually work so closely as to become indistinguishable from a single organism (Wheeler (1912)). Later research regarded some insect colonies as examples of collective intelligence. The concept of ant colony optimization algorithms, introduced by **Marco Dorigo**, became a popular theory of evolutionary computation. Colonies allocate workers to different tasks, and workers switch from one task to another in response to changing conditions.

Kennedy & Eberhart (1995) studied the collective behaviour of a flock of birds and showed that they manifest a collective intelligence behaviour in the environment. His simulation of the bird flocks led to a family of optimisation algorithms that can be collectively called: Particle Swarm Optimization (PSO).

Collective intelligence in principle assumes the individuals are simple in nature but the interaction among the individuals yields sophisticated behavior. This principle is exploited in order to solve complex optimisation problems by making the individuals navigate the search space of the problem in order to cover the surface of the function to be optimized. This simple but revolutionary idea lead to great advances in optimisation and metaheuristics with applications in many fields of science and engineering.

However, many real-life applications are far more complicated than the assumption that there is one ultimate goal of the optimisation process. Some problems can have several competing goals, so that getting closer to one goal may lead the individuals further away from the other goals. To tackle these problems a compromise must be

reached among all the goals of the optimisation process. This has shown to be a real challenge to the optimization community.

The motivation of this research is to look at PSO from a multi-objective perspective, i.e. modify the original PSO method so that it can handle complex problems with conflicted optimisation goals. The thesis builds on the state-of-the-art of both PSO and multi-objective optimisation and contributes to both fields on different levels and through the different stages of the optimisation process. As the real judge of an optimizer is its applicability in the real world, the thesis applies the newly developed methods on real-life applications showing the potential of these methods and their impact.

1.2 Aims and Objectives

The main goal of the thesis is to enrich the discipline of multi-objective optimisation with a set of new techniques to tackle challenging problems in the extension of PSO to the multi-objective domain. These new techniques will cover the main steps in a multi-objective PSO from neighbourhood definition of the particles to archiving and quality testing. The thesis builds on recent development in multi-objective optimisation which uses decomposition with genetic algorithms as the way to break down the multi-objective problem into simpler single objective ones and then solve these problems simultaneously. Decomposition is incorporated in the two main algorithms presented (SDMOPSO and D^2MOPSO), in addition to a new concept for archiving which creates a mapping between objective and solution spaces.

The secondary aim of the thesis is to test the developed techniques on real-life problems. To this end two problems were solved using the new algorithms: 1) The channel selection in Brain-Computer Interfaces 2) The dose regulation in Cancer chemotherapy. The thesis presents several solutions to these two problems in terms of problem design and results interpretation. The work on these two problems was in close collaboration with experts in the respected fields to ensure the quality of the data analysis.

The third, and final, objective of the thesis is to develop a new quality assessment measure for multi-objective optimisers. For the first time, the thesis presents a novel measure that uses mutual information as the basis for an indicator that sees the Pareto

fronts as images and applies mutual information to measure the quality of the approximated PF given the true one. This novel approach is much robust against outliers than the other commonly used measures as demonstrated in the thesis.

It should be stressed that although the thesis uses PSO as the basic optimiser, the developed techniques are general and can be used by other evolutionary optimisers (e.g. GA) as demonstrated in Section 3.3 where the new archive is applied on NSGAII instead of PSO.

1.3 Methodology

The research carried out in this thesis followed a strict procedure to guarantee high quality results and robust conclusions. This is demonstrated by my publications in high impact journals and conferences. The procedure consists of the following steps:

- problem identification: identify interesting research questions to answer and study the possibility of making significant contribution to address these questions.
- development: a solution(s) of the problem is considered and algorithms and methods are developed to tackle the problem in hand. The newly developed methods are tentatively tested in order to get feedback and adjust the methods accordingly.
- experimental design: to thoroughly test the developed methods experiments are run using standard problems. The results are compared to those of the state-of-the-art and based on the outcome of the comparison going back to development might be necessary.
- publication: the results are written and submitted to appropriate medium of publication in order to get feedback from the community to enhance the work.

The first identified problem was the definition of particles' neighbourhood within multi-objective PSO. The state-of-the-art MOPSO used techniques borrowed from the single objective PSO. I , on the other hand, have incorporated the decomposition approach into PSO in order to redefine the neighbourhood in PSO to better solve multi-objective problems. This work resulted in the development of Smart Multi-Objective PSO based on Decomposition (SDMOPSO).

Secondly the issue of archiving the generated solutions was tackled. The commonly used archives rely only on information in the objective space alone without into consideration the information in the solution space. Hence, I developed two solutions to map the two spaces and building an archive based on either clustering or $\epsilon-$dominance. This lead to Clustering based framework for Leaders Selection (CLS), and later the development of a new MOPSO algorithm: MOPSO based on Decomposition and Dominance (D^2MOPSO).

The last theoretical identified challenge was the assessment of multi-objective optimizers. I have introduced a novel measure based on mutual information which is robust towards outliers.

A major evaluation part of any newly developed approach is the choice of evaluation problems. All the newly developed methods and algorithms were tested on commonly used test suits for multiobjective problems and real-life problems.

1.4 Summary of Contributions

The thesis contains four main tracks:

1. Archiving: this is a very important part of any evolutionary algorithm. Maintaining the solutions found within the optimisation process, deciding which solutions to dismiss and how to select the current leaders are all very sensitive part of any optimization process especially in the multi-objective paradigm. The thesis proposes a novel approach of archiving that maps the search and objective spaces to enhance the output solutions and their diversity.

2. Algorithms: two distinct algorithms are proposed in the thesis, namely SD-MOPSO and D^2MOPSO. They both share the inclusion of the concept of decomposition, but they differ in how to deploy this concept and integrate it within the framework of PSO.

3. Quality Measures: the thesis proposes a new quality measure of the Pareto Front, i.e. the solution set of the optimiser, in order to compare among different algorithms. The measure uses mutual information capable of circumventing a major drawback of most of the trendy measures by using not the distance calculation (Euclidean or otherwise), but statistical information instead.

4. Applications: The real test of an optimisation algorithm is by testing it on real-life problems. Since the beginning of the work, I have collaborate with the Brain-Computer Interfaces (BCI) group in the university of Essex and have applied my algorithms on some technical problems related to selecting the best channels of a BCI system for control. I have also tested part of my work on the problem of deciding the dose of chemotherapy treatment for patients in order to enhance their quality of life while on treatment.

1.5 Thesis Outline and Organization

Next chapter reviews the basic concepts behind Particle swarm optimisation and multi-objective optimisation leading to identifying the problems to be addressed in the following chapters. Chapter 3 describes the developed methods in this thesis including a new archiving technique and new algorithms for MOPSO. Chapter 4 applies some of the methods developed on real-life problems, while Chapter 5 introduces novel quality assessment method for multi-objective optimisers. Chapter 6 concludes the thesis.

Chapter 2

Literature Review

"Artificial Intelligence, IT'S HERE. "

-Business Week cover, July 9, 1984

2.1 Introduction

Intuitively, optimisation is the selection of the best element, with regard to some criteria, from a set of possible elements. In its simplest form, an optimisation problem is about finding the value that minimizes (or maximizes) a given function, i.e. objective function,(Gershenfeld (1998)). Practically, the shape of the objective function is unknown and the optimisation is usually restricted with constraints in the search space complicating the optimisation process.

Many optimisation methods have been developed over the years. Rothlauf (2011) provided detailed taxonomy of optimisation methods and the principles of solving an optimisation problem. Here we focus only on a particular type of optimizers, namely evolutionary algorithms, which are usually seen as metaheuristics. Metaheuristics make few or no assumptions on the problem to be solved which does not guarantee finding the optimal solution(s) to the problem. However, this same property allows for solving complicated optimisation problems.

Many problems in science and engineering require the simultaneous optimisation of more than one objective simultaneously. The challenge of solving multi-objective problems raises from the complexity of these problems and the difficulty of finding reasonable solutions for all objectives. It is usually the case that enhancing the performance in terms of one objective would result in the deterioration of the other objectives. The solutions generated by the optimizer are then trade-off solutions among the different objectives (Coello Coello *et al.* (2007)).

Coello Coello *et al.* (2007) argued that the use of evolutionary algorithms (EAs) to solve multi-objective problems is mainly motivated by their population-based nature. This allows the generation of optimal solutions (called Pareto optimal solution as will be discussed later) in a single run. In addition, some multi-objective problems can be very complicated to be solved by conventional deterministic optimizers.

2.2 Single Objective Optimisation

Historically single objective optimisation was first developed in order to solve problems where there is only one goal to optimise. Most, if not all, evolutionary algorithms started by a single objective version and then some where extended to account for the multiobjective case.

A minimisation/maximization single objective optimisation problem is defined as the minimization/maximization of a function $f(x) : \Omega \subseteq R^n \rightarrow R, \Omega \neq \emptyset$ subject to $g_i(x) \leq 0$, $i = \{1, \ldots, m\}$, and $h_j(x) = 0$, $j = \{1, \ldots, p\}$, $x = (x_1, \ldots, x_n) \in \Omega$ (Coello Coello *et al.* (2007)).

$g_i(x) \leq 0$ and $h_j(x) = 0$ are constraints that must be fulfilled while optimizing $f(x)$. Ω contains all possible x that can be used to evaluate $f(x)$ and satisfy its constraints. The method for finding the global optimum (minimum or maximum) which may not be unique is referred to as global optimisation.

For a minimization problem, the value $f^* = f(x^*) > -\infty$ is called a global minimum (or global optima in general) if and only if

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

x^* is by definition the global minimum solution , f is the objective function, and the set Ω is the feasible region of x.

The shape of $f(x)$ determines the difficulty of the optimisation problem. For a relatively simple function exact optimisation can be sufficient (e.g. analytical and numerical methods, Dynamic programming, etc... Rothlauf (2011)). In practice $f(x)$ can be much more difficult to optimize so heuristic based methods are needed (Rothlauf (2011)), which in turn can cause the optimizer to fall in what is called a local optima. The local optima is similar in definition to the global one, f^*, except that it is applied to a smaller region of the search space. It is worth noting that local optima is not a unique value for $f(x)$ which could complicate the optimisation problem. Local optima can be a problem for a lot of optimizers and especially the deterministic ones. Evolutionary algorithms may perform better to tackle this particular issue as they are stochastic in nature and then may have better chance of getting out of a local optima (El-Ghazali (2009)).

There is a comprehensive literature on single objective optimisation: numeric linear algebra (Ciarlet (1989)), Dynamic programming (King (2002)), and many others. Here we are only interested in a type of approximate optimizers namely the population-based metaheuristics (Back (1996); El-Ghazali (2009)).

2.3 Population-based Metaheuristics

Population-based metaheuristics are stochastic optimizers and contain a considerable number of algorithms that are mostly inspired by nature, e.g. Evolutionary Algorithms (EAs), Particle Swarm Optimisation (PSO), estimation of distribution algorithms (EDAs) and many others. Despite their intrinsic differences these algorithms share some common concepts:

- a population of potential solutions is initialized. The population and its members are called differently in different algorithms.
- a new generation of the population is generated based on the experience of the previous generation.
- the new generation is integrated with the old one using some selection criteria.
- this search process continues until a stopping condition is met, e.g. a pre-set number of iterations is reached.

The implementation of these concepts varies significantly among different algorithms. Understanding how each of these steps is tuned and implemented is essential to get the best out of any population-based optimizer. These concepts are further discussed in some detail in the following sections.

2.3.1 Initial Population

By definition population-based metaheuristics are exploration search algorithms, i.e. the algorithm starts by exploring large areas of the search space and then narrows down the search throughout the optimisation process.

Maaranen *et al.* (2007) argued that the initialization step plays a crucial role in the effectiveness of the algorithm. When generating the initial solutions, the most important aspect is the diversity of these solutions. If the initial solutions are not diverse enough, i.e. do not cover large areas of the search space, they could lead the algorithm to a premature convergence, i.e. falling into a local optima (El-Ghazali (2009)).

Unlike other approaches, in the context of Evolutionary algorithms and PSO the most common population initialization method is the random generator (El-Ghazali

(2009)). The random generator is usually performed using pseudo-random numbers utilising classical generators, e.g. recursive lagged Fibonacci, (Gentle (2003)).

2.3.2 Generation

In this phase, a new population is generated following the adopted generation strategy. There are mainly two generation strategies here:

- Evolutionary based: the solutions are selected and reproduced using a variation operators (e.g. mutation) acting directly on the current population, so that the newly generated solutions are directly obtained from the variables representing the solutions in the current population. Evolutionary algorithms and scatter search are examples of such algorithms (El-Ghazali (2009)).

- Backboard based: In this approach the members of the population contribute to a shared memory of the system. This share memory is then used to generate the new solutions. Particle swarm optimisation is an example of an algorithm that adopts this strategy. In PSO the best particle in the current population is considered the leader and affects the evolution of the solutions of the future particles in the swarm. The shared memory is then formed by this information interaction among the particles in the swarm (Section 2.4).

2.3.3 Selection

Once the new solutions are generated then the algorithm has to select the new population from the old population and the generated solutions. The simplistic method would be to use the generated solutions as the new population. Evolutionary algorithms usually applies a selection mechanism where the best solutions of the two sets are selected to form the new population. In backboard-based heuristics there is not a direct selection mechanism as the information exchange among the populations' members dictates the generation of the new population.

2.3.4 Stopping Criteria

Deciding on the stopping criteria is important as the algorithm should stop when it has converged on one side but it should not continue much after convergence is reached

as it causes wasting computational power. Stopping criteria can be static, i.e. the search ends when a fixed condition is satisfied, e.g. number of iterations, number of objective evaluations, etc. Alternatively an adaptive stopping criteria can be used such as a measure of objective approximation, or some statistics regarding the diversity of the reached solutions so far.

Next the Particle Swarm Optimisation is described in detail as it is the dominantly used algorithm in this work.

2.4 Particle Swarm Optimisation

Kennedy & Eberhart (1995) first proposed the idea of Particle Swarm Optimisation (PSO) as a method to attain computational intelligence based on the social interaction of individuals rather than the cognitive and intelligence competency of one individual. They established this type of intelligence by simulating the behavior of flocks of birds or fish and then developed a powerful optimisation method, namely PSO, (Eberhart *et al.* (1996); Kennedy & Eberhart (1995, 1997a)).

In principle PSO distributes a set of entities, called particles, in the search space of an optimisation problem. Each of these particles evaluates the objective function at its current location. In order to determine its next step the particle combines its experience with the locations of the particles with the best fit, i.e. leaders, with some random perturbations. In one iteration all the particles are moved and then the swarm as a whole moves, as a flock of birds converging for food, hopefully towards an optimum value of the function. The first application to PSO was to tune the parameters of a neural network (Eberhart *et al.* (1996)) but soon it gained popularity especially in problems with continuous search spaces (Engelbrecht (2007)), despite a binary version of the algorithm (Kennedy & Eberhart (1997a)).

Angeline (1998) and Eberhart & Shi (1998) compared the mostly used evolutionary algorithms, genetic algorithms (GA) and PSO and two main distinctions can be made between the two:

1. Evolutionary algorithms rely on three main mechanisms: parent representation, selection of individuals, and the fine tuning of parameters. On the other side, PSO does not use an explicit selection mechanism but rather uses leader(s) to

guide the optimisation process. The offspring generation in PSO is different from GA in terms of using the backboard approach rather than the evolutionary one.

2. Another key difference between PSO and evolutionary algorithms is the way the particles are manipulated. PSO direct particles by manipulating their velocity and direction as a result using both the particle's personal best and the global best (the leader's). On the other hand EA uses mutation operators that can set the direction of the individual in any direction.

According to Reyes-Sierra & Coello (2006) there are two main reasons for the popularity of PSO:

1. The PSO algorithm is relatively simple. The use of only one operator to create new solutions makes its implementation straightforward. In addition, there are plenty of reliable implementations online.

2. PSO is found to be very effective in many application domains producing competitive results at a low computational expense.

To solve a problem in a D dimensional search space using PSO, each particle is composed of three D dimensional vectors: the current position $\overrightarrow{x_i}$, the previous best position $\overrightarrow{pbest_i}$, and the velocity $\overrightarrow{v_i}$ (Kennedy & Eberhart (1995); Poli *et al.* (2007)). $\overrightarrow{x_i}$ is designed to be a point in the search space. During optimisation the particle positions are evaluated after each iteration as solutions to the problem. If a new position is better than what has been found so far, then it is stored in $\overrightarrow{pbest_i}$. The goal of the particle within the optmisation is then to keep enhancing $\overrightarrow{pbest_i}$. The particle moves to a new position in the search space by adding $\overrightarrow{v_i}$ to $\overrightarrow{x_i}$. The algorithm adjusts $\overrightarrow{v_i}$ which in returns changes the direction and speed of the change in position.

By definition the particle alone can not solve the optimisation problem. It can only operate in collaboration with the other particles in the swarm. The interaction among the particles in the swarm is governed by the neighbourhood definition, usually referred to as topology or social network. Each particle communicates with the other particles in its neighbourhood and is affected by the best particle in this neighbourhood, denoted local best $\overrightarrow{p_{lbest}}$ to distinguish it from the global best of the whole swarm $\overrightarrow{p_{gbest}}$.

Similar to other population-based metaheuristics PSO starts by randomly initializing the particles in the swarm. The position of each particle is then changed according to its own experience and that of its neighbours. The position of particle p_i at time t is changed by adding the velocity $\overrightarrow{v_i}(t)$ to the current position:

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t+1) \tag{2.2}$$

The velocity vector reflects the socially exchanged information and is defined in the following way:

$$\overrightarrow{v_i}(t+1) = W\overrightarrow{v_i}(t) + C_1 r_1(\overrightarrow{x}_{pbest_i} - \overrightarrow{x_i}(t+1)) + C_2 r_2(\overrightarrow{x}_{leader} - \overrightarrow{x_i}(t+1)) \tag{2.3}$$

where $r_1, r_2 \in [0,1]$ are random values. *leader* refer to either the global leader or local leader depending on the topology used. C_1, C_2 are learning factors, and W is the inertia weight.

2.4.1 Parameters

PSO has a very few number of parameters to be set. The first parameter is the size of the population. This is usually set empirically and it would normally increase with the increase of dimensionality.

C_1, C_2 are called the learning factors or acceleration coefficients and represent the magnitude of the particle in the direction of its personal best and its neighbourhood. The values of C_1 and C_2 can affect the optimization significantly as it can either make the PSO more responsive to change or unstable. These parameters in short control the exploration exploitation balance of the algorithm and hence must be carefully chosen.

The inertia weight, $W \in [0,1]$, controls the influence of the previous velocity vectors on the calculation of the current velocity. W is seen as a friction coefficient (Poli *et al.* (2007)), and so can be considered as the fluidity of the medium in which a particle moves. It might then be useful to start with a relatively high W (e.g. 0.9) which causes the particles to behave in an explanatory mode, and then gradually reducing W to around 0.4 where the system would be more exploitive. The strategy of updating W, or not at all, is very important to the optimisation process. Eberhart & Shi (2000) used a fuzzy system to adapt W. Zheng *et al.* (2003) showed that increasing inertia weight can produce good results.

Eberhart *et al.* (2001) showed that based on the network topology of the swarm, the movement of the particle can be greatly influenced by the neighbouring particles where the neighbourhood is defined by the swarm's topology. Following are some of the commonly used topologies (Engelbrecht (2007)).

- Empty Topology: here the particles are only driven by their experience, *pbest*, which means that the particle is isolated from all the other particles. In this case C_2 is set to zero. Fig. 2.1 shows an example of such a topology.
- Local best Topology: Each particle is directed by its neighbourhood. The neighbourhood has a fixed size (n) and the movement of each particle is influenced by the best performing particle in this neighbourhood. Fig. 2.2 presents an example of this network.
- Global best Topology: Also called fully connected topology. Each particle is influenced by a global leader of the swarm, i.e. the particle with the best performance, in addition to its own experience. In order to achieve this each particle is connected to every other particle in the swarm, Fig. 2.3.
- Star Topology: In this case only one particle, $focal$, is considered the head of the swarm and all other particles are directly connected to it but are isolated from each other. $focal$ compares performance among all particles in the swarm and adjusts its direction accordingly, Fig. 2.4.
- Tree Topology: All particles are arranged in a tree where each node of the tree contains only one particle. Each particle is influenced by its own experience, *pbest*, and that of the particle just above it in the tree. A child particle will replace its parent if it had reached a better solution, Fig. 2.5.

There are other network topologies that can be used and these are well discussed in (Poli *et al.* (2007)).

Figure 2.6 illustrates the general PSO algorithm (single optimization). In line with the population-based paradigm, PSO starts with an initialization step which includes both velocities and positions. The corresponding *pbest* of each particle is also initialized and the leader is identified (the leader definition depends on the network topology used). Then and for a maximum number of iterations the particles move in the search space

Figure 2.1: Diagram of an empty topology. A circle represents a particle and the directed arrow represents a direct link indicates a self connection.

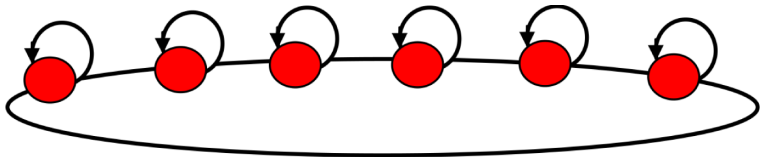


Figure 2.2: Diagram of a local best topology. A circle represents a particle and the directed arrow represents a direct link indicates a self connection, while the non-directed link represents a two-way link between two particles.

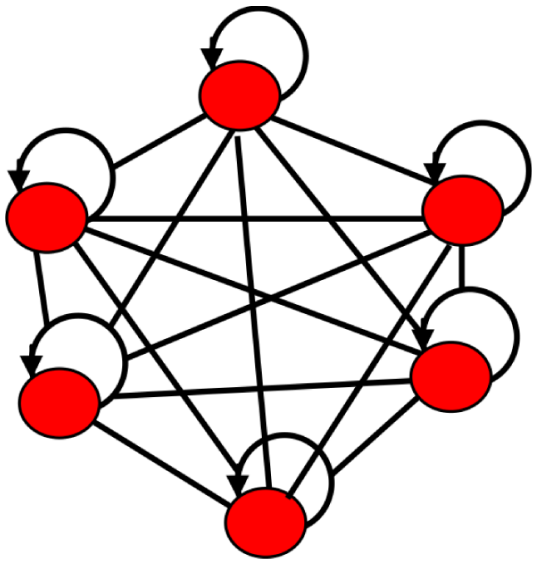


Figure 2.3: Diagram of a global best topology. A circle represents a particle and the directed arrow represents a direct link indicates a self connection, while the non-directed link represents a two-way link between two particles.

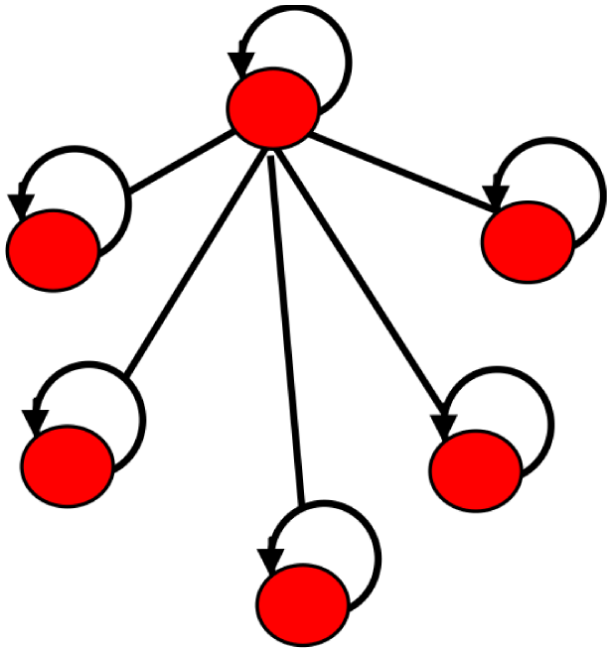


Figure 2.4: Diagram of a start topology. A circle represents a particle and the directed arrow represents a direct link indicates a self connection, while the non-directed link represents a two-way link between two particles.

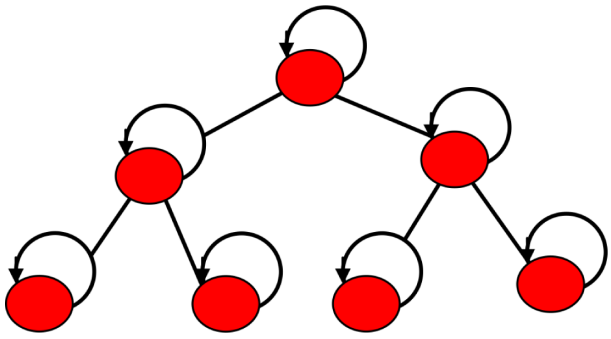


Figure 2.5: Diagram of a tree topology. A circle represents a particle and the directed arrow represents a direct link indicates a self connection, while the non-directed link represents a two-way link between two particles.

directed by their experience and the leader constrained by the network topology, the particles then update their velocity, position, and *pbest* using Eq. 2.2, and Eq.2.3. Finally the leader is updated and the iterations continue until the maximum number of iterations is reached.

2.4.2 Theoretical analyses and Convergence

Behind the apparently simplicity of PSO, it raises serious challenges for the theoretical understanding of its behaviour. Firstly, PSO consists of a large number of interacting particles. The particles themselves are simple to model but the interaction among these particles makes the modelling of such dynamic a complicated issue. Secondly, the particles has a memory which adds unpredictable factor to the modelling of the particles dynamics. This stochastic nature prevents the use of standard mathematical tools. Thirdly, the fitness functions can vary a lot and with them the behaviour of PSO.

For these reasons there is still no full mathematical modelling of PSO that captures fully its behaviour, however there are some attempts to tackle this issue (e.g. Blackwell (2007); Engelbrecht (2005); Kadirkamanathan *et al.* (2006)). Poli *et al.* (2007) provides a comprehensive review of these modelling attempts.

Engelbrecht (2005) showed that PSO is sensitive to the choice of control parameters. Most theoretical work on PSO made several simplification assumptions, the swarm is usually considered consisting of one particle in a one dimensional space, *pbest* and *gbest* are assumed constant throughout the process and so are $\phi_1 = C_1 r_1$ and $\phi_2 = C_2 r_2$. Under these conditions the swarm convergence is defined as follows.

Definition 1 *Giving the sequence of global best solutions* $\{gbest\}_{t=0}^{\infty}$*, the swarm is converged iff:* $\lim_{t\to\infty} gbest_t = p$

where p is an arbitrary position in the search space.

Ozcan & Mohan (1998) studied PSO under the previous conditions without considering the inertia weight. They showed that the trajectory of the particle can be seen as a sinusoidal wave where the initial conditions and parameter setting dictates the amplitude and frequency giving that $0 < \phi < 4$ where $\phi = \phi_1 + \phi_2$.

Van Den Bergh *et al.* (2002) built a similar model that accounts for inertia weight as well. Van Den Bergh *et al.* (2002) showed that when $W > (C_1 + C_2) - 1$, the particle

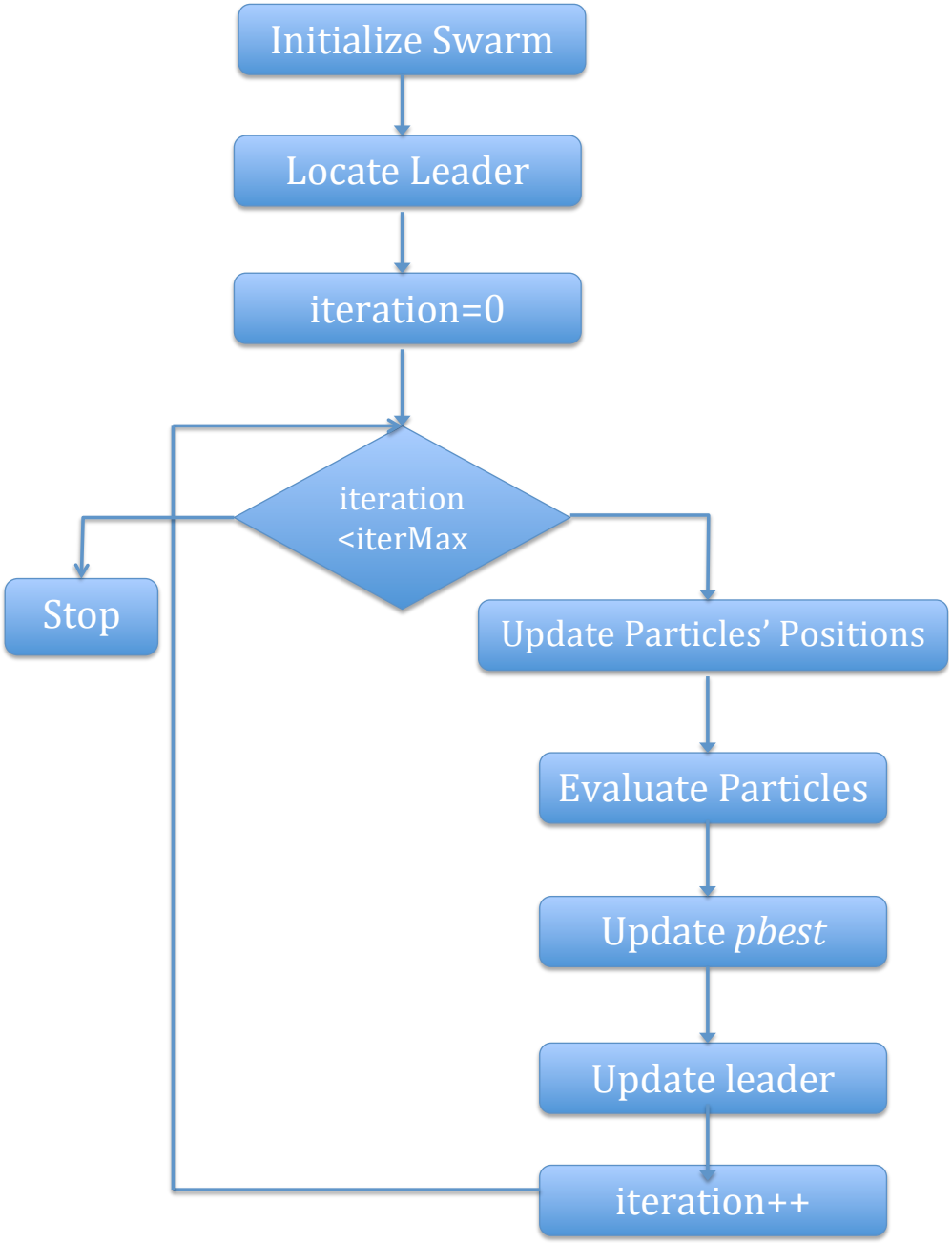


Figure 2.6: A chart diagram of the general PSO algorithm.

converges to $\frac{\phi_1 pbest + \phi_2 gbest}{\phi_1 + \phi_2}$. Then he generalized to the case where ϕ_1 and ϕ_2 are not constant and he concluded that the swarm converges to $(1-a)pbest + agbest$ where $a = \frac{C_2}{C_1 + C_2}$.

Clerc & Kennedy (2002) presented an analytical analysis of PSO with constriction coefficients to prevent the velocity from growing out of bounds.

The analysis discussed so far prove, under certain assumptions, the convergence of PSO, but to ensure the convergence to the local or global optimum, two conditions must be met: 1) monotonicity, i.e. *gbest* at a generation t can not be worse than *gbest* at the previous generation $t-1$. 2) the algorithm should generate a solution in the neighbourhood of the optimum from any solution $\overrightarrow{x}$ in the search space.

Van Den Bergh *et al.* (2002) showed that PSO satisfies the first condition but not the second one so the original PSO can not be considered a global, or local, optimizer as there is no guarantee that it will reach a global/local optimum. The solutions suggested for this premature convergence problem of PSO include using mutation operators, or sub-swarms, or the reinitialisation of the swarm when the algorithm is converged. These simple solutions are usually effective enough for PSO to work in practice.

So far the algorithms and methods described are all interested in solving problems with a fitness function containing only one objective. Next section describes a more sophisticated type of problems: multi-objective optimisation problems, where more than one objective is involved in the optimisation problem complicating the job for the optimizer.

2.5 Multi-objective Optimisation

The traditional way of looking at optimisation problems as single objective, i.e. the fitness function is only concerned with one quantity to optimize, can lead in some real-life situations to sub-optimal solutions of the problem due to the misrepresentation of the optimisation problem. An alternative type of optimisation is described as multi-objective optimisation in which there are more than one objective. These objectives are usually conflicting in nature so any improvement in one objective often happens at the expense of deterioration in other objective(s). The optimisation challenge therefore is to find the entire set of trade-off solutions that satisfy all conflicting objectives (Coello Coello *et al.* (2007)). Multi-objective optimization has roots in mathematical

optimization with the first multi-objective evolutionary algorithm (MOEA) presented in 1985, Schaffer (1985). The field has grown rapidly in the last 10 to 15 years and is now known a dominant area of research within the optimization community.

Coello Coello *et al.* (2007) presented a sophisticated description of the formal definition of multi-objective problems (MOPs). Here I use simplified definitions that is commonly used in the literature (Reyes-Sierra & Coello (2006)).

MOP is an optimization problem in the solution/search space $\Omega \subset R^n$ where an n-dimensional vector $\overrightarrow{x} = [x_1, x_2, \ldots, x_n]^T \in \Omega$ is defined and is referred to as the decision variables vector. The problem's objective is then defined as a vector of objectives in the objective space: $\overrightarrow{f}(\overrightarrow{x}) \in \Delta \subset R^m$:

$$\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_m(\overrightarrow{x})]^T \tag{2.4}$$

where $m \geq 2$ is the number of objectives, $f_i : R^n \rightarrow R, i = 1, \ldots, m$ are the objective functions. For a minimization problem, without any loss of generality, the goal of the optimizer is to minimize $\overrightarrow{f}(\overrightarrow{x})$ subject to

$$g_i(\overrightarrow{x}) \leq 0, i = 1, 2, \ldots, k, \tag{2.5}$$

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

where $g_i, h_j : R^n \rightarrow R, i = 1, \ldots, k, j = 1, \ldots, p$ are the constraint functions of the problem. From now on, $\overrightarrow{f}(\overrightarrow{x})$ and $\overrightarrow{f}$ are used interchangeably.

Definition 2 *Given two vectors $\overrightarrow{x}, \overrightarrow{y} \in R^m$, then $\overrightarrow{x}$ is said to dominate $\overrightarrow{y}$ if $\overrightarrow{x} \neq \overrightarrow{y}$ and $f(\overrightarrow{x}) \leq f(\overrightarrow{y})$ and is denoted by $\overrightarrow{x} \prec \overrightarrow{y}$. $\overrightarrow{x} \leq \overrightarrow{y} \iff x_i \leq y_i$, for $\forall i = 1 \ldots, m$.*

Definition 3 *A vector of decision variable $\overrightarrow{x} \in \Omega$ is said to be nondominated with respect to Ω if there does not exist another $\overrightarrow{y} \in \Omega$ such that $\overrightarrow{f}(\overrightarrow{y}) \prec \overrightarrow{f}(\overrightarrow{x})$.*

Definition 4 *Let $F \subset R^n$ be the set of feasible solutions in the search space, i.e. the solutions do not violate the constraints. $\overrightarrow{x^*} \in F$ is Pareto optimal if it is nondominated with respect to F.*

The Pareto optimality of a solution guarantees that any enhancement of one objective would results in the worsening of at least one other objective.

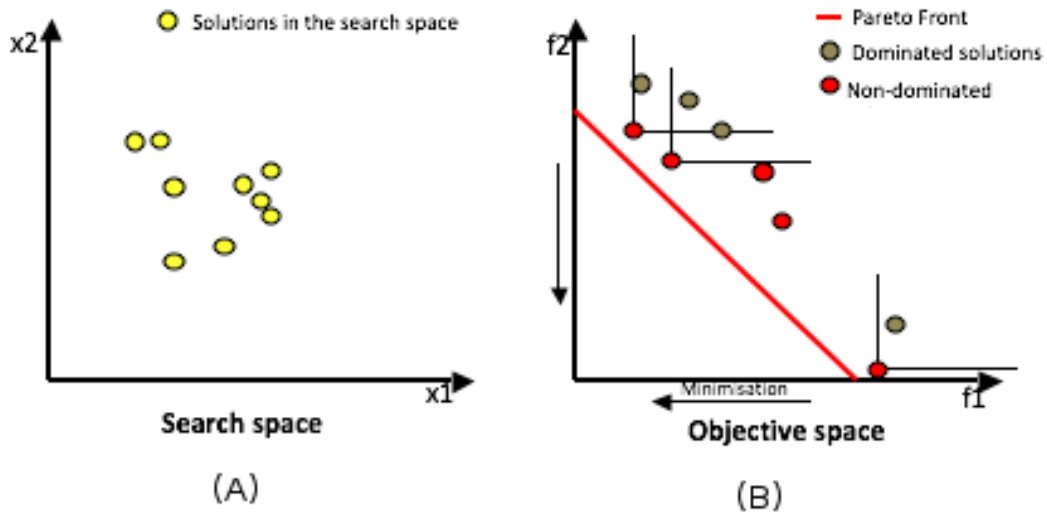


Figure 2.7: An example of a multi-objective optimisation problem: A) the particles/solutions in the search space B) the corresponding Pareto front.

Definition 5 *The Pareto optimal set* P^* *is defined as* $P^* = \{\overrightarrow{x} \in F | \overrightarrow{x}$ *is Pareto optimal*$\}$.

Definition 6 *A Pareto Front* PF^* *is the image of* P^* *in the objective space:* $PF^* = \{\overrightarrow{f}(\overrightarrow{x}) \in R^m | \overrightarrow{x} \in P^*\}$

The goal of the multi-objective optimizer is then to locate the Pareto optimal set in F which, in practice might be unachievable or partly undesirable. Fig. 2.7 shows an example of particles (represented in solutions) in the search space (Fig. 2.7 (A)) and the Pareto front they form (Fig. 2.7 (B)).

Solving MOPs is highly dependent on the structure of the PF, in addition to the number of objectives as the number of optimal solutions necessary to find a good approximation of the PF tends to grow with the increase in the number of objectives. A multi-objective evolutionary algorithm aims at producing an approximated PF that fully covers the PF.

2.6 Multi-objective Evolutionary Algorithms

MOPs can be solved using a wide range of algorithms and approaches, classified in detail by Cohon (2004). However, here we are only interested in multi-objective evolutionary algorithms (MOEA).

The basic algorithm design is based on using the Pareto-based fitness assignment to identity the nondominated individuals of the current population. All MOEAs however, share four abstract goals:

- Preserve nondominated points.
- Progress the current PF towards the true PF.
- Maintain diversity of the points on the approximated PF.
- Prevent loss of good solutions by archiving them and maintaining the archive.

Following is a description of some common MOEAs that are more related to this thesis with a more detailed description given in (Coello Coello *et al.* (2007)):

2.6.1 Algorithms based on Aggregation Functions

This is the oldest and probably the simplest approach to solve MOPs (Tucker (1957)). It is known that a Pareto optimal solution to a MOP can be seen as the optimal solution of a scalar optimisation problem in which the objective is an aggregation of all f_i's. It transforms a multi-objective problem into a single objective one using an aggregation function $A : R^m \rightarrow R$, where m is the number of objectives. The weighted average is the most commonly used function. A minimization MOP is transformed to the form:

$$min \sum_{i=1}^{m} \lambda_i f_i(\overrightarrow{x}) \tag{2.7}$$

where $\lambda_i \geq 0$ and $\sum_{i=1}^{m} \lambda_i = 1$. Assigning the weights to the objectives can be a challenging task taking into consideration that unbalanced weights could lead to biased optimisation to the objective with the largest weight.

The weighted average is part of the family of linear aggregation functions. Other nonlinear function have also been used in the literature. Miettinen (1999) reviewed

several aggregation functions, the most popular of those include Tchebycheff and the boundary intersection method (Das & Dennis (1998); Mattson *et al.* (2004)).

In the Tchebycheff approach the scalar optimization problem is formed as:

$$min_{g^{te}}(\overrightarrow{x}|\lambda, \overrightarrow{z}^*) = min_{1\leq i\leq m}\{\lambda_i|f_i(\overrightarrow{x}) - z_i^*|\} \tag{2.8}$$

where $\overrightarrow{z}^* = (z_1^*, \ldots, z_m^*)^T$ is the reference point, i.e. $z_i^* = min\{f_i(\overrightarrow{x})\}$. For each optimal point $\overrightarrow{x}^*$ there is a weight vector λ such that $\overrightarrow{x}^*$ is the optimal solution for Eq. 2.8 so it is possible to obtain Pareto optimal solutions by changing the weight vector λ.

Tchebycheff does not generate a smooth enough function for problems with a continuous PF. An approach that is designed for continuous problems is the Boundary Intersection (BI) approach (Das & Dennis (1998)). Geometrically BI approaches aim to find intersection points between the most top boundary and a set of lines radiating from the reference point. If these lines are evenly distributed then a good approximation of PF is expected.

$$min_{g^{bi}}(\overrightarrow{x}|\lambda, \overrightarrow{z}^*) = d \tag{2.9}$$

subject to $\overrightarrow{f}(\overrightarrow{x}) - \overrightarrow{z}^* = d.\lambda$. The constraint $\overrightarrow{f}(\overrightarrow{x}) - \overrightarrow{z}^* = d.\lambda$ ensures that $\overrightarrow{f}(\overrightarrow{x})$ is always on the line with direction λ and passing through $\overrightarrow{z}^*$. The goal is to push $\overrightarrow{f}(\overrightarrow{x})$ towards the boundary of the attainable objective set. One difficulty of this approach is solving the equality constraint. An equivalent definition can be used called penalty BI (PBI) as follows:

$$min_{g^{pbi}}(\overrightarrow{x}|\lambda, \overrightarrow{z}^*) = d_1 + \theta d_2 \tag{2.10}$$

subject to $\overrightarrow{x}$, where

$$d_1 = \frac{||(\overrightarrow{f}(\overrightarrow{x}) - \overrightarrow{z}^*)^T\lambda||}{\lambda} \tag{2.11}$$

and

$$d_2 = ||\overrightarrow{f}(\overrightarrow{x}) - (\overrightarrow{z}^* + d_1\lambda)|| \tag{2.12}$$

$\theta > 0$ is a preset penalty parameter. If $\overrightarrow{y}$ is the projection of $\overrightarrow{f}(\overrightarrow{x})$ on line L then d_1 is the distance between $\overrightarrow{z}^*$ and $\overrightarrow{y}$ and d_2 is the distance between $\overrightarrow{f}(\overrightarrow{x})$ and L. PBI, and BI, usually produces better distributed approximated PF than Tchebycheff

and usually ensures that if $\overrightarrow{x}$ dominates $\overrightarrow{y}$ then they do not have the same g^{pbi} value unlike Tchebycheff (Zhang & Li (2007)).

2.6.2 Non-dominated Sorting Genetic Algorithm (NSGA) and NSGAII

NSGA (Srinivas & Deb (1994a)), and its extension NSGAII (Deb *et al.* (2002)), is one of the early and most commonly used MOEAs. NSGA ranks the individuals in several layers. The first of which contains all the non-dominated individuals in the population and considered the highest rank. These are then removed from the optimisation process and then the same ranking procedure is repeated until all the individuals are put into these layers. To maintain the diversity, a dummy fitness function is shared among all individuals in one layer. The individuals in the layers closer to the PF have better ranking, hence they get higher chance of reproduction than the rest of the population.

NSGAII is similar in principle to NSGA but uses a different ranking mechanism. The ranking of an individual is determined by the total number of the individuals dominated by it and also the number of individuals dominating it. To maintain the diversity of the points on the PF a crowding distance is utilised in order to spread the points over the whole PF. NSGAII compares two individuals based on the ranking as in NSGA in addition to the crowding distance. If two individuals have the same rank then the one with less crowding distance is selected for mating and producing the next generation.

2.6.3 Multi-objective Evolutionary Algorithms based on Decomposition (MOEA/D)

Following the aggregation function approach the approximation of the PF can be decomposed into a number of scalar objective optimization sub-problems (Zhang & Li (2007)). This is the idea behind a family of mathematical programming methods for multi-objective optimization. In evolutionary algorithms each individual solves one aggregation problem, i.e. different weights for the objectives in the weighted average function, then each individual is assigned a relative fitness that reflects its utility for selection and hence the scalar optimizers can easily be extended to MOPs with their techniques easily applied (Zhang & Li (2007)).

Zhang & Li (2007) presented MOEA/D which explicitly decompose the MOP into N scalar optimization sub-problems and then solves them simultaneously by evolving a population of solutions. The population at any generation is composed of the best solutions found so far for all sub-problems. A neighbourhood relationship is defined among the individuals based on their aggregation coefficients, i.e. λ_i in Eq.2.7. An individual uses the information of its neighbouring sub-problems to optimise its own. In the following description of the algorithm the Tchebycheff approach is used for decomposition.

Let $\lambda^1, \ldots, \lambda^N$ be a set of evenly spread weight vectors, which are fixed throughout the optimization process, and $\overrightarrow{z}^*$ be the reference point. The problem of approximating the PF it is decomposed to N scalar sub-problems using the Tchebycheff approach. Similar to Eq. 2.8 the objective function of the jth sub-problem is

$$g^{te}(\overrightarrow{x}|\lambda^j, \overrightarrow{z}^*) = min_{1\leq i\leq m}\{\lambda_i^j|f_i(\overrightarrow{x}) - z_i^*|\} \tag{2.13}$$

where $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$. MOEA/D minimizes all the N objective functions simultaneously in a single run. An important feature of MOEA/D is that if two weight vectors λ^i and λ^j are close then $g^{te}(\overrightarrow{x}|\lambda^i, \overrightarrow{z}^*)$ and $g^{te}(\overrightarrow{x}|\lambda^j, \overrightarrow{z}^*)$ must be close to each other.

The neighbourhood of λ^i is the set of closest weight vectors which is translated as the neighbourhood of the i^th sub-problem consists of the sub-problems with the weight vectors of the neighbourhood of λ^i. The population at one generation consists of the best solution found so far for each sub-problem.

In the initialization phase of MOEA/D the Euclidean distance is measured between any two weight vectors. Then for each weight vector the closest T weight vectors form the neighbourhood. The neighbourhoods are used to exchange information among the individuals and used to generate the offspring of the current population. Unlike other MOEA, MOEA/D does not use crowding distances to maintain the diversity of the population (see the next section for more details on diversification) because MOEA/D decomposes the MOP into scalar optimization sub-problems. The diversity among the sub-problems is thought to lead to the diversity in the population, so when the decomposition method and the weight vectors are properly chosen then the resulted sub-problems will be evenly distributed. Algorithm 2.1 summarizes the steps of MOEA/D.

Algorithm 2.1 MOEA/D

1: Initialize external archive EP.
2: Compute the Euclidean distances between any two weight vectors and then work out the closest T weight vectors to each weight vector.
3: Generate an initial population randomly
4: Randomly Initialize z, the best value found so far for all objectives.
5: **for** for each sub-problem i **do**
6: randomly select two individuals and generate a new solution y using genetic operators
7: Apply a problem-specific improvement heuristic on the produced solution
8: update z if new solution is better
9: update neighbouring solutions
10: remove all vectors in EP dominated by F(y)
11: add F(y) to EP if no vector in EP dominates F(y)
12: **end for**
13: Repeat Steps 5-12 until stopping criteria is reached.

MOEA/D, and its extension for expensive problem (Zhang *et al.* (2010)), has found a good success in the field and is considered one of the dominant algorithms for the last few years.

Next, the multi-objective PSO is reviewed in detailed as it is the basic algorithm used throughout this thesis.

2.7 Multi-Objective Particle Swarm Optimisation

Applying PSO to multi-objective optimisation problems requires changing some aspects of the original algorithm to adapt to the special requirements of such problems and hence to achieve the abstract goals defined in Section 2.6.

Considering PSO is a population-based metaheuristic, it would be beneficial to generate several distinct nondominated solutions in a single run of the algorithm. In line with other MOEAs three points must be addressed (Coello Coello *et al.* (2007)):

- Leaders selection: this is one of the most sensitive issues in multi-objective PSO. The selection mechanism should give preference to the nondominated solutions over dominated ones.

- Archiving: the nondominated solutions produced throughout the optimization process should be retained , or at least a subset of them, to report the nondominated solutions on several runs and not only the current one. It is also desirable that the kept solutions are well spread over the PF.

- Diversity: the algorithm should maintain a certain level of diversity in the solutions produced throughout the optimization process to avoid converging into a single solution, or a small set of solutions.

Although these issues are also important in any MOEA they are particularly important for Multi-objective PSO (MOPSO), and especially when it comes to maintaining the diversity of the solutions as PSO does not have the random element available to Genetic Algorithms, (using mutation for instance).

In the single objective case the leader(s) is predetermined by the topology of the social network of the swarm. In the multiobjective case things become much more complicated as each particle can have several leaders from which only one can be used to update the particle's position. This set of leaders is usually kept outside the swarm and is referred to as 'external archive'. The external archive contains the nondominated solutions found so far. The solutions in the external archive are considered leaders to the particles in the swarm. By the end of the optimization run the content of the external archive is considered the output of the algorithm.

Algorithm 2.2 outlines the general structure of a MOPSO. As in the single objective version, MOPSO starts by initializing the particles in the swarm, but here the external archive is also initialized by the nondominated initialized particles. Next, some sort of quality measure is calculated for all the leaders in order to decide on one leader for each particle of the swarm (Step 12: of Algo. 2.2). At every generation a leader is selected for each particle (Section 2.8.1) and the particle moves using Eq. 2.2 and Eq. 2.3. The particle is then evaluated and *pbest* is updated. A new particle replaces its *pbest* when the *pbest* is dominated or the two are not comparable, i.e. they are both nondominated to each other. The leaders archive is then updated and the same quality measure is recalculated. This process continues until a fix number of iterations is reached (Reyes-Sierra & Coello (2006)).

Algorithm 2.2 General MOPSO Algorithm

```
1: Initialize swarm
2: Initialize leaders in the external archive
3: Measure the quality of the leaders
4: for iteration=1 to maxItr do
5:    for each particle do
6:       Select leader
7:       Update position
8:       Evaluation
9:       Update pbest
10:   end for
11:   Update leaders in the external archive
12:   Measure the quality of the leaders
13: end for
14: Return the content of the external archive
```

2.8 Archiving in MOPSO

Looking closely at the intrinsic characteristics of PSO and the general structure of MOPSO there seems to be three main issues to address (Pulido (2005)): 1) the leaders selection and update, 2) maintaining the leaders archive, 3) the creation of new solutions. These issues are usually discussed as archiving approaches in MOPSO. Next I discuss these three points in more detail:

2.8.1 Leaders in MOPSO

As mentioned above, the traditional definition of a leader in PSO does not apply directly in the multi-objective case as there are several leaders who are all equally good solutions. One could circumvent this issue altogether by using an aggregation function as described in Section 2.6.1, or by optimizing each objective separately. However, most methods redefine the concept of the leader.

The key issue though is the selection of the leader for each particle. The common approach is to consider every nondominated particle as a leader and then either select one randomly to update the particle's position or using a quality measure to compare among these leaders.

A popular family of quality measures of the leaders is based on density measures. Several other measures are also used in the literature which will be discussed in more detail in the next section and Section 2.10.

2.8.2 Archiving and Spreading of Nondominated Solutions

Nondominated solutions generated throughout the whole search process should be kept so that they are returned at the end of the optimization process. Rudolph (1998) provided a theoretical and analytical motivation for archiving the nondominated solutions.

An external archive is typically used by MOPSO to retain the nondominated solutions. In principle the external archive operates on two main constraints for adding a new particle to the archive: 1) the particle should be nondominated with respect to the particles already in the archive. 2) the particle dominates at least one particle in the archive in which case the latter particle is deleted and replaced by the former one.

This definition of the external archive have two main drawbacks (Coello Coello *et al.* (2007)). First, the size of the archive may increase very quickly. Secondly, the particles in the external archive can crowd in certain regions of the objective space while leaving empty areas in the PF. The increasing size of the archive can raise a real challenge to the optimizer as it makes updating the archive on every generation a very computationally expensive process. If the size of the swarm is not large enough, MOPSO can be in a situation where all particles of the swarm are feasible candidates to enter the archive making the total complexity of update the archive $O(kMN^2)$ where N is the size of the swarm, k is the number of objectives, and M is the number of iterations. Storing large number of non-dominated solutions does not only have high computational cost but also seems to be useless for most of the decision makers when it exceeds a reasonable size limit.

To circumvent this issue the external archives are bounded to a fixed size (e.g. Raquel & Naval Jr (2005)) making it necessary to add a criterion for retaining non-dominated solutions when the archive is full. MOEAs provide some techniques that can be used for this purpose, with the most commonly used ones described next. The goal of these methods is not only to reduce and control the size of the archive but also to retain the leaders of the swarm that will generate a well spread PF. The particles in the archive must not be crowded in small regions in the objective space but rather

spread out and be diverse enough to find novel solutions. Most of the leader selection techniques then can also be viewed as the ways to diversify the generated solutions.

It must be noted that, practically, three archives should be used when extending PSO for multi-objective optimization: one for storing the global best solutions, one for the personal best values and a third one for storing the local best (if applicable). However, few authors report the use of more than one archive in their MOPSOs and the same leader selection techniques are applied to the three archives, so only external archive is mentioned here without any loss of generality.

2.8.2.1 Kernel:

Kernel methods (Fonseca & Fleming (1993)) define the neighborhood of a solution using a kernel function that takes the distance between two solutions as the argument. The density estimator of a solution is represented by the sum of the kernel function values (usually referred to as crowding distance). The individuals with the highest crowding distance are preferred. The crowding distance is a value assigned to each individual indicating the density of the individual's location.

2.8.2.2 Adaptive Grid Algorithm

This technique is first introduced for Pareto Archived Evolution Strategy (PAES) ,Knowles & Corne (2000). The algorithm divides the objective space in a recursive manner and favours the less crowded cells. Each solution is placed in a certain grid location based on the values of its objectives which are used as coordinates. A map of such a grid is maintained, indicating the number of solutions that reside in each grid location. The algorithm has knowledge of the objectives limits and continues updating the solutions' geographical information when a new solution is introduced that breaks one or more of the objective limits. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space).

2.8.2.3 Niche Count

In function optimization, the location of each optimum is referred to as niche. The idea behind niche count is that stable sub-populations at each optimum can be formed by suitably sharing the fitness associated with each niche. When a particle is sharing

resources with others, its fitness degrades in proportion to the number and closeness to particles that surround it within a certain perimeter. A neighbourhood of a particle is defined in terms of a parameter called σ that indicates the radius of the neighbourhood (Deb & Goldberg (1989); Goldberg & Richardson (1987)).

2.8.2.4 Clustering

The problem of clustering, also known as unsupervised learning, is the problem of identifying groups, or clusters, of data points in a multidimensional space (Bishop *et al.* (2006)). Assuming we have a dataset $\{x_1, \ldots, x_N\}$ of N data points in a D-dimensional space. The goal is then to partition the data into K clusters. Intuitively, the cluster consists of data points with inter-point distances smaller than the distance between points of different clusters. K is usually unknown but assumption can be made depending on the clustering algorithm, e.g. K-means requires K to be set in before clustering while average linkage method uses the data itself to decide on K. For leader selection the solutions in the external archive are clustered with every cluster is represented by one particle usually at its centre. These representative particles are considered the new reduced none-dominated set (Morse (1980); Zitzler & Thiele (1999)). Figure 2.8 is an example of the output of a clustering algorithm (Gaussian Mixture Model is this case,Bishop *et al.* (2006)) where each cluster's data are coloured in a distinct colour and the boundaries represent the spread of a Gaussian distribution that models the cluster.

2.8.2.5 $\epsilon-$dominance

$\epsilon-$dominance is a soft dominance approach, in which the dominance region of a solution is extended by a parameter ϵ (Figure 2.9 explains graphically the concept of $\epsilon-$dominance). This concept has been used for the leader selection problem in MOPSO from the external archive (Laumanns *et al.* (2002)). By using $\epsilon-$dominance the objective space is divided into boxes of size ϵ. In each of these boxes only one particle is kept according to some criteria (Alvarez-Benitez *et al.* (2005); Moore & Chapman (1999)), Fig. 2.10 shows an example of a two dimensional objective space with $\epsilon-$dominance in application. $\epsilon-$ dominance as illustrated here guarantees that the selected solutions are nondominated with respect to all solutions generated during the run. From Fig. 2.10 we can see that the parameter ϵ controls the number of boxes in the space and

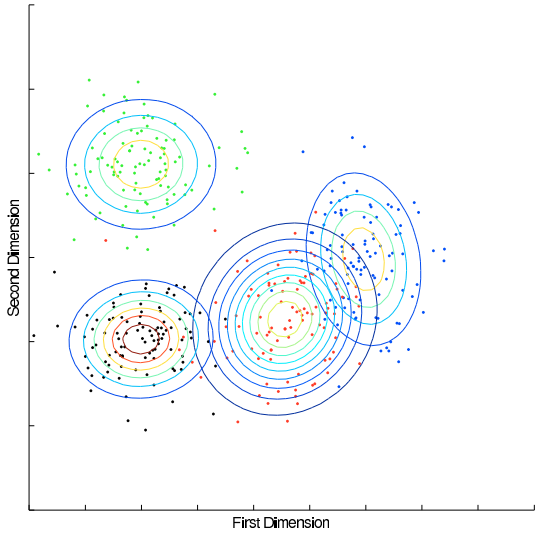


Figure 2.8: Example of the result of a clustering algorithm (Gaussian Mixture Model).

hence the size of the resulted external archive. Mostaghim & Teich (2003a) studied the differences in terms of computational cost and convergence between clustering and $\epsilon-$dominance based management of the archive and found that $\epsilon-$dominance is much less computationally expensive and does not affect the convergence of the algorithm.

2.8.2.6 Nearest Neighbour Density Estimator

This concept was first used in NSGAII (Deb *et al.* (2002)). In this approach the distance between a given solution i and its kth nearest neighbour is taken into account to estimate the density of the solution. Solutions in the archive are ranked according to their crowding distance. The crowding distance is defined as the circumference of the cuboid defined by the particle's left and right neighbours. The particles with the highest crowding distance are preferred as they are in a less crowded regions. Figure 2.11 demonstrates an example of the nearest neighbour density estimator in two objectives space. For one solution (i) the nearest two non-dominated solutions along the two objectives are $i+1$ and $i-1$ on each sides. Cuboid is the rectangle surrounding i. Solutions with larger cuboid circumference are preferred to those with smaller circumference.

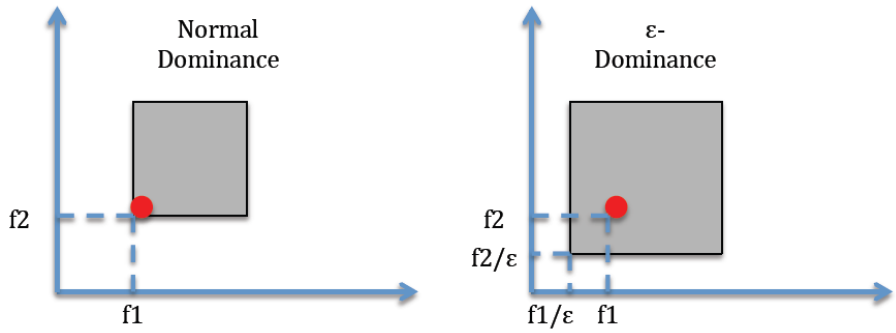


Figure 2.9: To the left, the area dominated, in a minimization problem, by a certain solution is highlighted. To the right the area being dominated has been extended by a value proportional to the ϵ parameter.

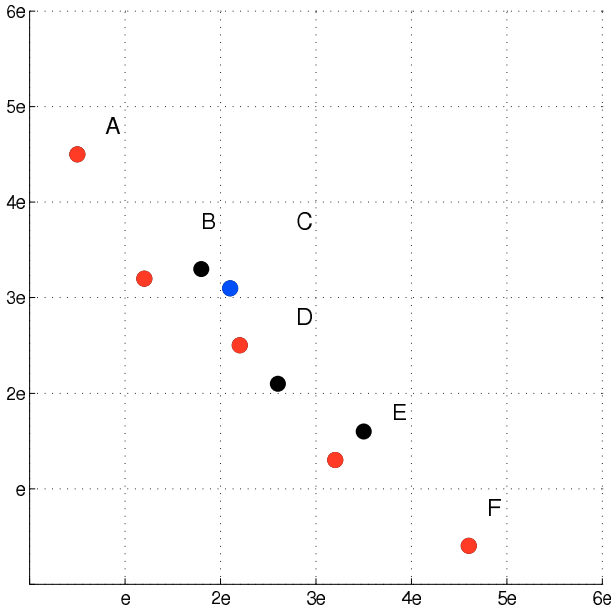


Figure 2.10: $\epsilon-$dominance in two dimensional objective space. The red dots are the selected particles. In box B the black particle is removed as it is dominated by the red one. In box D the red and the black particles are not comparable, i.e. no one dominates the other, so the red particle is chosen as it is more to the left. Box C is discarded because it is dominated by B and D.

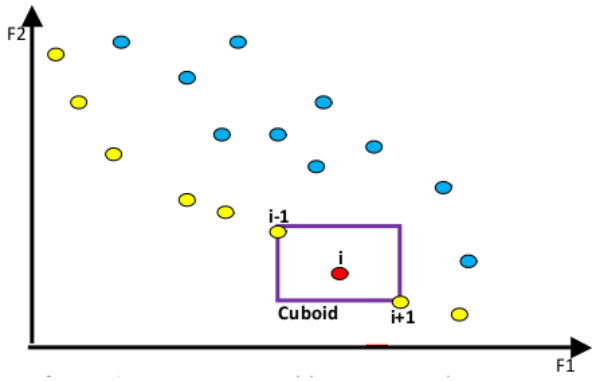


Figure 2.11: Example of nearest neighbour density estimator in two dimensional space.

2.8.3 Diversification and Avoidance of Local Optima

One of the most attractive features of PSO is its fast convergence. Unfortunately, this comes at the cost of risking premature convergence, i.e. falling in a local optima (Section 2.2). Reyes-Sierra & Coello (2006) argued that the main reason behind the premature convergence of PSO is the lack of diversity within the swarm which can be credited to the topology of the network or the inappropriate number of particle. It is then crucial to promote diversity in PSO to control its convergence.

It is clear from the leader selection mechanisms, discussed in the previous section, that they intrinsically promote diversity by trying to retain in the archive those solution that are not crowded and are well spread on the PF. However there are two other approaches in which one can promote diversity and avoid local optima during the creation of new particles:

2.8.3.1 Position Update

As argued before, the swarm neighbourhood topology determines the speed of information transfer among the particles in the swarm. For example in the case of fully connected network the information is transferred faster than within a tree topology since the particles have smaller neighbourhoods. Following the same argument the topology determines how fast the diversity is lost within the swarm. The fully connected network would lose diversity much faster than other topologies with smaller

neighbourhoods. So the choice of the topology is essential to the preservation of the diversity in the swarm.

The Inertia weight is another parameter that can affect diversity, as it controls the impact of the particle experience on the current velocity (Eq. 2.3). In other words it controls the trade-off between exploration (looking at the global perspective of the search space) and exploitation (focusing on local information) (Shi & Eberhart (1998)). A large inertia weight allows for the exploration for new regions in the search space, while a small inertia weight focuses the search and fine-tunes it in a local area.

Shi & Eberhart (1999) argued that the linear decrease of the inertia weight from a large value at the beginning of the optimisation to a small value at the end would give PSO a more global look at the beginning and fine-tuned results at the end.

Shi & Eberhart (1998) presented a view of the velocity update of the particles in PSO as a "conscience" mutation, because they argued PSO's update mechanism is similar to mutation in evolutionary algorithms but it is guided by the experience of the particles in PSO.

2.8.3.2 Turbulence

Despite the "conscience" mutation of PSO, some craziness might be necessary according to Kennedy & Eberhart (1995). This irrational behaviour of some particles in the swarm is referred to as turbulence and is reflected as a change in a particle's flight that is out of control.

During the optimisation process the swarm might stagnate, i.e. the velocity vectors are almost zero for all particles so there is no change in the swarm and PSO is stuck in a local optima. Because in PSO the leader(s) drives all the swarm, if the leader(s) is stuck in a local optima and there is not enough velocity for any other particle to become the new leader then the whole swarm will come to a halt. The turbulence provides the swarm a chance of escaping this local optima by changing the trajectories of some particles in random directions which would hopefully generate new leader(s) and hence escape the local optima, thereby potentially speeding up the convergence (Stacey *et al.* (2003)).

Turbulence, or mutation, seems to provide a good answer for immature convergence in PSO. However, similar to genetic algorithms the application of mutation is not straightforward. The first choice to make is the type of mutation operator to apply.

There are many mutation operators used in the literature. The choice of the operator is crucial to the optimization process, and it is usually not very easy to decide on one, taking into account the significant impact it can have on the optimization. Once the mutation operator is chosen, it is applied with certain probability on some solutions but then another questions raise: how to set this probability, when to apply mutation, and in which component of the particle. Despite some use of this approach in the literature (e.g. Stacey *et al.* (2003)), I have decide in this work to avoid using it in order to keep PSO itself very simple with less parameters to tune.

2.9 Multi-objective PSO algorithms

The taxonomy of MOPSO algorithms is very close to that of the general MOEA. In this thesis a slightly modified taxonomy is used (Coello Coello *et al.* (2007)) which classifies MOPSOs as follows:

- Aggregating approaches.
- Pareto-based approaches.
- Combined approaches.
- Decomposition-based approaches.

Next we discuss these different classes with an emphasis on decomposition-based approaches as they are the most related to the work undertaken.

2.9.1 Aggregating Approaches

These approaches uses directly the techniques described in Section 2.6.1 to aggregate the several objectives into one objective and then treat the multi-objective optimisation problem as a single optimisation problem. Parsopoulos & Vrahatis (2002) presented a MOPSO algorithm that adopts three aggregation functions: 1) the weighted sum discussed in Section 2.6.1 2) a dynamic aggregation function that gradually modifies its weights during optimisation 3) the weights of the aggregated function change abruptly during optimisation. Baumgartner *et al.* (2004) divides the swarm into a set of subswarms, each of which has a different set of weights for the aggregation function. These

sub-swarms evolve independently to find their own leaders, and then a gradient technique is used to identify the Pareto optimal solutions.

2.9.2 Pareto-based Approaches

In these approaches Pareto dominance is used for leader selection. In other words leaders are considered the nondominated particles with respect to the swarm. However, as discussed before, other criteria for leader selection and diversification are usually used giving a variety of MOPSO algorithms.

Ray & Liew (2002) used the Pareto dominance concept and combined it with the nearest neighbourhood density estimator to promote diversity (based on a roulette selection scheme of leaders utilizing this value). Fieldsend *et al.* (2002) uses an unconstrained elite external archive with a special data structure called "dominated tree" which is used to define *gbest*. The archive interacts with the population to determine leaders.

Coello Coello & Lechuga (2002); Coello Coello *et al.* (2004) developed a MOPSO algorithm that uses an external archive in which every particle adds its current location after each cycle. The search space is divided into hypercubes. Each hypercube has a fitness related to the number of particles in it. To select a leader the roulette-wheel selection mechanism is used first to select the hypercube then the leader is randomly chosen from within the hypercube. Mutation operator is also applied on the particles. Pulido & Coello Coello (2004) used the Pareto dominance to determine the particle's flight direction. Clustering is used to split the swarm into sub-swarms in order to enhance the distribution of particles in the search space. The external archive is replaced by an elitism mechanism where sub-swarms exchange their leaders.

Sigma MOPSO was proposed by Mostaghim & Teich (2003b), in which each particle $\overrightarrow{x}$ is assigned a value, σ, based on its location in the objective space:

$$\sigma = \frac{f_1^2(\overrightarrow{x}) - f_2^2(\overrightarrow{x})}{f_1^2(\overrightarrow{x}) + f_2^2(\overrightarrow{x})} \tag{2.14}$$

for a bi-objective problem, where f_1, f_2 are the objectives (Fig.2.12). Using this definition: all the particles where $f_1(\overrightarrow{x}) = af_2(\overrightarrow{x})$, i.e. are located in the objective space on a line with slope a, would have the same σ. The leader for the corresponding solution $\overrightarrow{x}$ is the one that has σ_{leader} with the closest distance to σ. The clustered particles in

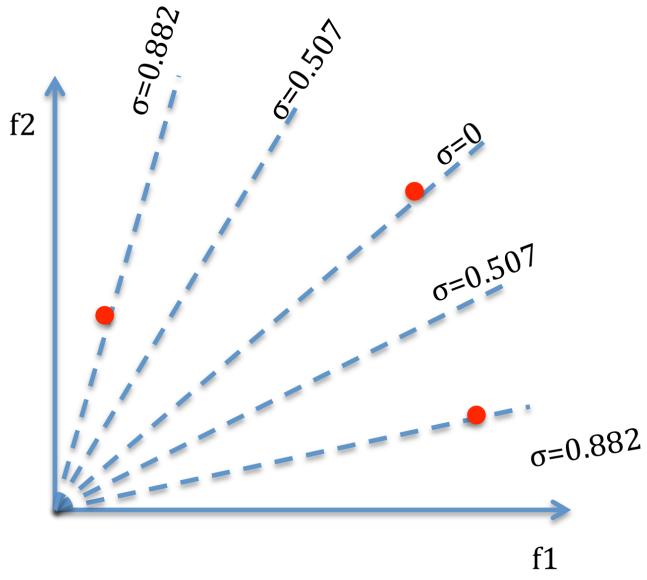


Figure 2.12: An example of Sigma-MOPSO.

the swarm have similar σ, making them moving in the same direction, as a result of selecting a set of clustered leaders. This might compromise the coverage and diversity of the approximated PF. Hence Sigma-MOPSO requires a large swarm (Parsopoulos & Vrahatis (2008)). Turbulence operator is also used in this algorithm within the search space. Mostaghim & Teich (2004) proposed a covering MOPSO algorithm (cv-MOPSO) as an extension to sigma MOPSO, which successively improve the previous external archive so in the first phase cvMOPSO runs with a restricted size external archive to get a good approximation of PF. On the second phase the non-dominated solutions obtained from the first phase are considered the input external archive of the cvMOPSO. The particles are divided into sub-swarms around each non-dominated solution after the first generation. The motivation of having the sub-swarms is to cover the gaps between the non-dominated solutions.

Bartz-Beielstein *et al.* (2003) analysed the application of archiving techniques into MOPSO and proposed a method which uses deletion and selection techniques, e.g. the adaptive grid. The deletion paradigm is based on the contribution of each particle to the diversity of the PF. Selecting methods are either inversely related to the fitness of the particles or the previous success of the particle.

Li (2003) adapted techniques from NSGAII into MOPSO. When the particle updates its position, instead of using its *pbest* all the *pbest* of the swarm and the newly reached solutions are combined into one set. The algorithm then selects solutions of this set to form the current swarm using non-dominated sorting. The leaders are selected randomly from an external leaders archive in which the leaders are stored based on two mechanisms: niche count and a nearest neighbour density estimator. Mutation operator is also used at each iteration to the particle with the lowest density estimator.

OMOPSO (Sierra & Coello Coello (2005)) is a popular approach and is used in this thesis to compare with the new approaches proposed. OMOPSO is based on Pareto dominance and uses a nearest density estimator for leader selection using a binary tournament approach. Two external archives are used: one for storing the leaders during the optimization process, and the second to store the final solutions. The density estimator is utilized to filter out the list of leaders when the maximum limit of the external archive is reached so that only the leaders with the highest density estimator are kept in the archive. ϵ-dominance decides on the particle to stay in the archive of the final solutions. A fully connected network is used. The swarm is divided into three sub-swarms temporarily in order to apply a mutation operator separately to each sub-swarm then the mutated sub-swarms are put back together to perform the leader selection. Algorithm 2.3 outlines the algorithm of OMOPSO as described by Sierra & Coello Coello (2005).

Algorithm 2.3 OMOPSO

```
1: Initialize swarm. Initialize leaders. Send leaders to ϵ-archive
2: crowding(leaders), g=0
3: while g¡gmax do
4:    for each particle do
5:       Select leader. Flight. Mutation. Evaluation. Update pbest
6:    end for
7:    Update leaders, Send leaders to ϵ-archive
8:    crowding(leaders), g++
9: end while
10: Report results in ϵ-archive
```

Ho *et al.* (2005) proposed a novel formula to update the velocity and position of the particles by changing three main features of Eq. 2.3: 1) r_1, r_2 are considered dependent,

and r_2 is set to $1 - r_1$, 2) (1-W) is incorporated in the second and third terms of Eq. 2.3, where W is a random variable between 0 and 1; 3) an assumption is made that the particle might have to fly back so the sign of the first term in Eq. 2.3 is flipped with a probability of 50%.

A new diversification mechanism was proposed by Villalobos-Arias *et al.* (2005) which utilizes stripes that are applied on the objective function space. The main idea is the assumption that the Pareto front of the problem is similar to the line determined by the minimal points of the objective functions. In this way, several points (called stripe centers) are uniformly distributed around a strip, and the particles are associated with the nearest strip center which is considered the leader. The leader is selected minimizing a weighted sum of the minimal points of the objective functions. The authors showed that their approach can circumvent some drawbacks of other popular mechanisms such as ϵ-dominance and sigma.

ClustMPSO is a hybrid MOPSO proposed by Janson & Merkle (2005) which combines PSO with clustering technique, namely K-means, to divide the swarm into sub-swarms. Each sub-swarm has it is own non-dominated front and then the fronts from all sub-swarms are combined into the approximated PF. The leaders of the particles are randomly selected from the non-dominated front of the sub-swarm they are associated with. *pbest* is updated for each particle based on dominance relations. A domination relationship is also defined among the sub-swarms. A sub-swarm is dominated if all its particles do not belong to the combined PF.

2.9.3 Combined Approaches

These approaches try to combine different types of approaches to benefit from the advantages of more than one approach and to circumvent the drawbacks of either approach.

Mahfouf *et al.* (2004) introduced the adaptive weighted PSO (AWPSO) algorithm, in which the velocity increases with time. The goal is to enhance the global search ability at the end of the run and to help avoiding local optima. A weighted aggregation function is used for performance evaluation and to guide the selection of the personal and global bests. Dynamic weights used to generate Pareto optimal solutions. To maintain the diversity of the solutions, a mutation operator is applied to the positions of some particles.

2.9.4 Decomposition-based Approaches

Since the introduction of the decomposition approach in evolutionary algorithm (Zhang & Li (2007)) it drew a lot of interest from the evolutionary algorithms community. The first trial to incorporate MOEA/D with PSO was MOPSO/D followed by my algorithm SDMOPSO, and then dMOPSO. Section 3.4.10 presents analytical comparison among the different methods while following we briefly present these methods (except for SDMOPSO which will be discussed in detail in Section 3.2).

MOPSO/D (Peng & Zhang (2008)) is a multi-objective optimization method that uses the MOEA/D framework to solve continuous MOPs. MOPSO/D substitutes the genetic algorithm in MOEA/D with PSO. It relies fully on decomposition to update the personal and global information. Each particle is associated with one global best, so an update of a particle position can trigger position update in its neighbour's global best(s) resulting in duplications and making the algorithm prone to falling into local optima. Hence, mutation is employed trying to escape the local optima. Algorithm 2.4 presents the main body of MOPSO/D.

Algorithm 2.4 MOPSO/D

```
1: Initialize the weight vectors (Section 2.6.3)
2: Generate an initial swarm and the reference point z*
3: Initialize pbest and gbest
4: iteration=0
5: for iteration=1 to maxItr do
6:    for each particle do
7:       Update position
8:       Improve solution if distance between old and new positions is less than a thresh-
         old
9:       Update z*, pbest, and gbest
10:      remove dominated solutions from external archive
11:   end for
12: end for
13: Return the content of the external archive as the final output
```

dMOPSO: dMOPSO (Martínez & Coello Coello (2011)) uses decomposition to update the leaders' archive and to select the swarm leader(s). The archive stores the particles with the best aggregation values for each particle in the swam, whereas the

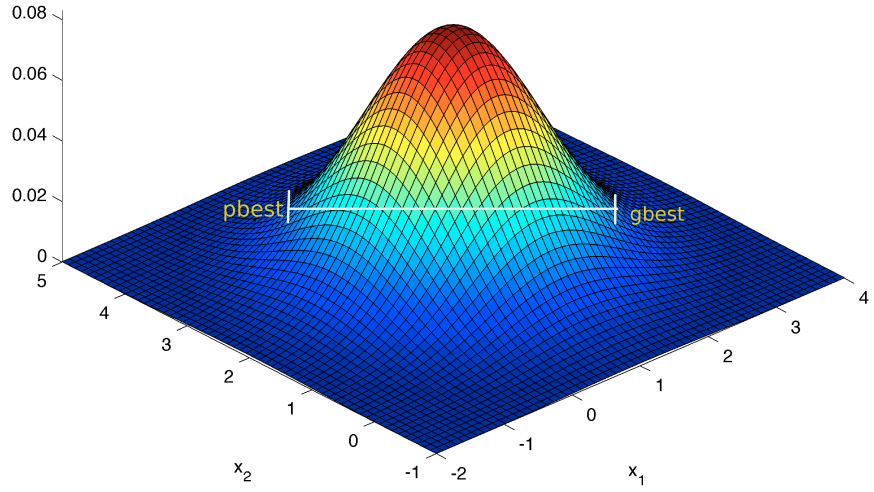


Figure 2.13: Illustration of the Gaussian distribution used by dMOPSO for generating a new particle. The personal best and the global best are used to define the mean and variance of the Gaussian distribution.

particles' personal memory stores the position with the best aggregated value found so far. To maintain the diversity of the swarm and to avoid local optima, dMOPSO re-initializes the particles' memory using a Gaussian normal distribution (Fig. 2.13 shows an example of such a distribution) when the particle exceeds a certain age (i.e. number of iterations with no update). This may lead to losing all the experience gained throughout the exploration process, as well as adding more complexity to the algorithm. Besides, it uses decomposition as a way to substitute dominance. With the absence of dominance, the decomposition strategy is confined to leading the swarm into a limited number of destinations equal to the swarm size (the number of λ vectors). With complicated Pareto fronts (i.e. disconnected) and the limited size of the swarm, dMOPSO tends to fail to cover the entire PF. Algorithm 2.5 presents the outlines of the dMOPSO algorithm.

2.9.5 Convergence Properties of MOPSO

So far there is not any study of the convergence of MOPSO. However following the discussion in Section 2.4.2 it is possible to choose the algorithm parameters in a way to guarantee convergence (Reyes-Sierra & Coello (2006)). The two conditions required to ensure convergence in the single objective PSO (Section 2.4.2) still hold for the multi

Algorithm 2.5 dMOPSO Algorithm

```
1: Initialize weight vectors
2: Initialize the swam with randomly generated particles
3: Initialize velocity, pbest, and gbest
4: iteration=0
5: for iteration=1 to maxItr do
6:    for each particle do
7:       if particle has aged then
8:          reset particle's memory
9:       else
10:         update velocity and position
11:      end if
12:      Repair particle's bounds
13:      Evaluate particle and update pbest and z*
14:   end for
15:   Update gbest
16: end for
17: Return the content of the external archive
```

objective case with the first condition (monotonicity) is changed to: The solutions contained in the external archive at iteration $t + 1$ should be non-dominated with respect to the solutions generated in all previous iterations.

ϵ-dominance based archiving ensures that this condition is satisfied as it guarantees that for any solution discarded from the archive one with equal or dominating objective vector is accepted. For a given MOPSO approach there is a roadmap to ensure the satisfaction of the first condition but it remains to explore if it satisfies the second condition in order to ensure global convergence to the true Pareto front.

2.10 Quality Measures

As with single objective optimisation, two factors are important when assessing a multi-objective optimiser: the quality of the found solutions, and the time spent to find them. However the stochastic nature of evolutionary algorithms results in the relation between time and quality not fixed, but rather represented by a probability distribution function. Hence, when discussing quality in evolutionary algorithm we need to look at it from a

probabilistic point of view. Additionally having a set of solutions (Pareto front) instead of a single outcome of the multi-objective optimisation process makes quantifying the quality of these solutions much harder. This is added to having multiple runs and the necessity to statistically quantify the behaviour of the optimiser over these runs increasing the difficulty of quality assessment (Fonseca *et al.* (2005b)).

According to a review, Fonseca *et al.* (2005b), there are two main approaches for quality assessment: a) model the outcome of the optimizer as a probability density function in the objective space. b) the indicator approach which quantifies the outcome of a run with a number with statistical analysis applied to these performance values. Several studies emphasized the importance of design and application of quality measures and especially how the statistical tests are applied and interpreted (Knowles & Corne (2002); Okabe *et al.* (2003); Zitzler *et al.* (2003b)).

In principle the easiest way to compare between the performances of two multi-objective optimizers is by comparing directly the resulted output of the two methods (e.g. using an indicator of quality like the ones described later). This would work if the optimizers are deterministic, i.e. running the optimizer twice will return the same results.

MOPSO, and multi-objective evolutionary algorithms in general, are stochastic in nature, due to the random element in the algorithms, i.e. running the algorithm twice would most likely produce a different set of results. For this reason the optimizer should be run repeatedly and the probability density function is then empirically estimated. Comparing two optimizers would then mean comparing their probability density functions which then implicate the issue of statistical hypothesis testing (Fonseca *et al.* (2005a)).

In the literature, there are two main approaches to assess the quality of produced PFs. The most common one is the indicator approach where a PF is mapped, using a defined function, to a real number then a standard statistical hypothesis test is applied on the indicator values (Fonseca *et al.* (2005b)). The second approach is usually referred to as the attainment function method in which for each objective vector there is a probability p that the produced approximation set contains an objective vector that weakly dominates z. The attainment function then gives a probability estimate of z to be attained in one optimization run with a statistical test procedure to count for all the runs (Grunert da Fonseca *et al.* (2001)).

A comprehensive survey on quality indicators can be found in (Fonseca *et al.* (2005b)). However, in the following sub sections we list in brief the commonly used indicators:

2.10.1 Error Ratio

The error ratio metric I_{ER} reports the number of vectors in the approximated PF, PF_{approx} that are not in the true PF ,PF_{true} (Van Veldhuizen (1999)). The metric requires that PF_{true} is known and that the MOEA approaches the PF. Formally speaking, the metric is represented as in:

$$I_{ER} = \frac{\sum_{i=1}^{|PF_{approx}|} e_i}{|PF_{approx}|} \tag{2.15}$$

where $||$ is the determinant operator and e_i is zero when the i^{th} vector of PF_{approx} is an element of PF_{true} and $e_i = 1$ when the i^{th} vector is not an element of PF_{true}. The smaller the value of I_{ER} the better the approximation of PF.

2.10.2 Generational Distance

The generational distance indicator, I_{GD}, measure how far on average PF_{approx} from PF_{true} (Coello Coello *et al.* (2007)). As with the error ration it requires the knowledge of the true PF and is mathematically defined as follows:

$$I_{GD} = \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{|PF_{approx}|} \tag{2.16}$$

where d_i is the Euclidean distance between each member i of PF_{approx} and the closest member of PF_{true}, and p is usually set to 2. $||$ is the cardinality operator.

2.10.3 Inverted Generational Distance

The inverted generational distance, I_{IGD}, (Van Veldhuizen & Lamont (1998)) measures the uniformity of distribution of the obtained solutions in terms of dispersion and extension. The average distance is calculated for each point of the actual PF (PF_{true}), denoted as A, and the nearest point of the approximated PF (PF_{approx}), denoted as B.

$$I_{IGD(A,B)} = \frac{(\sum_{a \in A} (\min_{b \in B} \parallel F(a) - F(b) \parallel^2))^{1/2}}{|A|} \tag{2.17}$$

2.10.4 Hypervolume

The hypervolume indicator, I_{hv}, (Zitzler & Thiele (1998)) measures the volume of the objective space that is weakly dominated by a PF approximation (A). I_{hv} uses a reference point v^* which denotes an upper bound over all objectives. v^* is defined as the worst objective values found in A (i.e. v^* is dominated by all solutions in A). Using the Lebesgue measure (Λ), I_{hv} is defined as:

$$I_{hv}(A) = \Lambda\Big(\bigcup_{a \in A} \{x \,|a \prec x \prec v^*|\}\Big). \tag{2.18}$$

2.10.5 ϵ Indicator

The ϵ indicator, I_ϵ, (Zitzler *et al.* (2003c)) measures the minimum distance which a PF approximation (A) has to be translated in the objective space to weakly dominate the actual PF B. The ϵ-Indicator is defined as:

$$I_\epsilon(A, B) = \min_{\epsilon \in \mathbb{R}} \{\forall b \in B, \exists b'_i - \epsilon \leq b_i, \forall 1 \leq i \leq n\} \tag{2.19}$$

Using quality indicators is an attractive approach of quality assessment due to its simplicity. It has , however, some shortcomings: 1) each indicator looks at the performance from only one perspective, e.g. spread, diversity, or dominance, which may skew the conclusions drawn. 2) In the case of incomparable PFs an indicator will actually give an inaccurate result. 3) For indicators that use distance functions, outliers can cause a real problem in disturbing the calculation of the indicator. 4) Quality indicators do not take the statistics of the data in the objective space into account which can be vital information to properly assess the quality of the performance. Fonseca *et al.* (2005b) discussed other important aspect that should be considered when using quality measures such as scaling and normalization and the possible combination of quality measures.

The attainment function method on the other hand distances itself from the indicator approach by estimating probability density functions of the attained objectives and then apply statistical tests to compare amongst the different PFs which circumvent most of the issues raised by the use of quality indicators but at the expense of a high computational requirements that they cannot be used for large number of objectives.

Chapter 5 takes an approach which models the output of the optimizer directly as an empirical probability density function and then calculates the mutual information between the approximated PF and the theoretical one. The higher the mutual information, the closer the approximation to the theoretical PF.

2.11 Benchmark Problems

Artificially constructed test problems offer many advantages over real-world problems for the purpose of general performance testing. Test problems can be designed to be easy to describe, easy to understand and visualize, easy to implement, fast, and their optima are often known in advance.

In the literature, many test problems have been designed over the years with several different properties to test for convex, concave, constrained and unconstrained real-life problems. Huband *et al.* (2006) provided detailed analysis of the commonly used test problems. Table 2.1 briefly summarizes the problems used in this thesis with their main features:

- F1: Number of Objectives.
- F2: Pareto Optimal Geometry. The geometry of the Pareto optimal front can be convex, linear, concave, mixed, degenerate, connected, disconnected, or some combination of the former.
- F3: Bias. A test problem may or may not be biased.
- F4: Number of constraints.

The methods introduced in this thesis use different combinations of these problems as a general check of the performance of the proposed methods. However, tests on real-life problems are also applied to study the realistic application of these methods.

Next chapter takes the state of the art and the concepts discussed in this chapter and advances them by introducing a novel method to incorporate decomposition and multi-objective particle swarm optimisation, a new archiving technique based on clustering, and a MOPSO that uses decomposition and dominance to outperform both.

Table 2.1: A comparison among commonly used problems: F1 is the number of objectives. F2 is the geometry of the Pareto Front. F3 does the problem have any bias (+) or not (-). F4 the number of constraints.

Problem	F1	F2	F3	F4
ZDT1	2	convex	-	0
ZDT2	2	concave	-	0
ZDT3	2	disconnected	-	0
ZDT4	2	convex	-	0
ZDT6	2	concave	+	0
DTLZ1	3	linear	-	0
DTLZ2	3	concave	-	0
DTLZ3	3	concave	-	0
DTLZ4	3	concave	+	0
DTLZ5	3	arch embedded	-	0
DTLZ6	3	arch embedded	+	0
DTLZ7	3	linear	-	0
WFG1	2	convex	+	0
WFG2	2	convex	-	0
WFG3	2	linear	-	0
WFG4	2	concave	-	0
WFG5	2	concave	-	0
WFG6	2	concave	-	0
WFG7	2	concave	parameter dependent	0
WFG8	2	concave	parameter dependent	0
WFG9	2	concave	parameter dependent	0
Viennet2	3	concave	-	0
Viennet3	3	convex	-	0
Viennet4	3	convex	-	2
Kursawe	2	disconnected	-	0
Schaffer	2	convex	-	0
Fonseca	2	convex	-	0
Tanaka	2	disconnected	-	2
Srinivas	2	linear	-	1
Osyczka2	2	convex	-	6
Golinski	2	convex	-	11
ConstrEx	2	convex	-	1

Chapter 3

Methods

"The wheel needs reinventing, but not just yet. "

Nir Oren

3.1 Introduction

Following the success of MOEA/D, it was thought of as a good basis for developing MOPSOs due to its simplicity and efficiency (Section 2.9.4). However, the early attempts to build such hyper algorithms did not perform well as the main MOEA/D algorithm was not changed to take into consideration the special properties of PSO. I developed in this chapter two MOPSO based decomposition methods (SDMOPSO, and D^2MOPSO) as promising alternatives to the state-of-the-art; they also tackle a delicate issue of mutiobjective optimization in general, and MOPSO in particular: archiving. SDMOPSO provides a new neighbouring technique for MOPSO based on decomposition. A novel approach of archiving is studied and then incorporated into a new hybrid of MOPSO and decomposition: D^2MOPSO. The advantage of these new algorithms is demonstrated by comparing with the state-of-the-art on benchmark test problems.

Chapter 4 demonstrate the usefulness of these algorithms in two real-world applications.

3.2 SDMOPSO

SDMOPSO (or Smart Multi-Objective Particle Swarm Optimisation using Decomposition) follows MOEA/D (Zhang & Li (2007); Zhang *et al.* (2009)) in decomposing the MOP into scalar aggregation problems. Decomposition transforms the MOP into a set of distinct scalar aggregation problems. Every particle solves the corresponding problem by applying priorities to each objective according to its weighting vector (λ_i, see Section 2.6.1 for the definition of λ_i). This assists the optimisation process in finding potential solutions that are evenly distributed along the PF and to mitigate against premature convergence. By associating every particle with a distinct scalar aggregation problem, the exploration activity of each particle will be focused on a specific region in the objective space and aimed at reducing the distance to the reference point.

SDMOPSO introduces a new approach for information exchange between neighbouring particles without a need for extra evaluations. The motivation being that the same solution in the objective space may have different aggregated values depending on its λ; thus, a solution (i.e. a position in the decision space) is assigned to the particle that uses it to produce the best aggregation value. If the new calculated position does

not enhance the aggregated fitness of one particle, then the particle shares the new position with its neighbours as this could enhance their aggregated fitness. In other words, useless information for one particle can be effectively utilised by other particles, depending on their λ. This leads to fewer objective function evaluations and results in wider dissemination of the discovered information, facilitating thereby simultaneous optimisation of the scalar problems. Taking into account the topological structure of the PSO population, sharing the information with neighbours will help relaying the discoveries of one particle to the entire swarm.

Many scalar approaches have been proposed to aggregate the objectives of a MOP, discussed in Section 2.6.1. The weighted Tchebycheff method is used here. The reference point $\overrightarrow{z}^*$ is determined by SDMOPSO as the vector of best values for each objective found so far by the optimisation process. Each particle is evaluated according to Eq. 2.8 using the associated λ.

SDMOPSO uses a crowding archive to store the set of swarm leaders (Section 2.8). The size of the crowding archive is fixed using ϵ-dominance (Sierra & Coello Coello (2005)). At the end of each iteration the crowding archive is updated with the new non-dominated particles in the current population, and the corresponding crowding values are adjusted in accordance with the number of new updates. This approach limits the size of the crowding archive and determines which particles to be deleted when the maximum size is exceeded. This is done in such a way that the diversity of Pareto optimal solutions is always maintained (Sierra & Coello Coello (2005)).

3.2.1 The Algorithm

SDMOPSO starts by initializing the population and initializing N vectors: $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, where m is the number of objectives and N is the swarm size. λ vectors are uniformly distributed in $[0,1]^m$ subject to $\sum_{i=1}^{m} \lambda_i = 1$. Every particle is assigned a unique λ_i. λ_i is selected so that it gives the best aggregated fitness value for the initialized particle. In the case of a minimization problem the particle will be assigned to λ_i that minimizes the aggregated fitness, taking into account that each λ_i is unique and will be assigned to only one particle of the swarm. The particles' memories *pbest* are then initialized, and the initial velocity of each particle is set to zero. The size of the crowding archive is set to the swarm size, and then is initialized using the non-dominated particles in the swarm. The reference point $\overrightarrow{z^*}$ is the vector in the objective

space with the best objective values found so far. The neighbourhood will be initialized by defining the neighbourhood size $\mathcal{N}$. The neighbourhood of a particle j is defined by the $\mathcal{N}$ particles that have associated λ_i vectors with the closest Euclidean distance to λ_j.

Algorithm 3.1 SDMOPSO

1: Initialize the swarm with N particles and N λ vectors
2: **for** $i = 1$ to N **do**
3: assign the particle i to a fixed λ vector which minimizes the aggregate of the objective functions.
4: initialize $pbest_i$
5: **end for**
6: Initialize velocities $\overrightarrow{V} = \{\overrightarrow{v_1}, \ldots, \overrightarrow{v_N}\}$, archive, neighbourhood and $\overrightarrow{z^*}$
7: Crowding(archive)
8: **for** $i = 1$ to $MaxIteration$ **do**
9: **for** $j = 1$ to N **do**
10: define particle j future Velocity, $\overrightarrow{v_j}(t+1)$
11: define particle j future Position, $\overrightarrow{x_j}(t+1)$
12: calculate scalar aggregate function for j
13: update the current population with the new particle j
14: update $pbest_j$, archive, and z^*
15: **end for**
16: **end for**
17: Return the final result in the crowding archive

The second phase of the optimisation process is repeated for a pre-defined number of iterations. During every iteration each particle defines a local view in the objective space. The particle determines the next move by finding the new velocity and new position using Eq. 3.1 and Eq. 2.2. The new velocity is calculated using the *pbest* values of a randomly selected neighbour particle and that of the current particle. The particle will then offer this local information (i.e. the decision vector and the corresponding objective vector) to its neighbours (including itself) so that every particle of the neighbourhood uses the new position and the evaluated objectives to calculate a new aggregated fitness value. If the new position enhances the particle's scaled fitness value, then it is adopted as the new position of the particle. Only up to two particles

are allowed to update (as suggested in Zhang *et al.* (2009)) their information and take advantages of this local information in order to avoid duplication of particles in the swarm. Evaluating the new information using different λ_i will not involve additional objective function evaluations as it only reads the stored values.

$$\overrightarrow{v_i}(t+1) = W\overrightarrow{v_i}(t) + C_1 r_1(\overrightarrow{x}_{pbest_i} - \overrightarrow{x_i}(t+1)) + C_2 r_2(\overrightarrow{x}_{nbest_i} - \overrightarrow{x_i}(t+1)) \quad (3.1)$$

where $nbest_i$ is the index of a random neighbouring particle from the set of $\mathcal{N}$ neighbours of $particle_i$, $r_1, r_2 \in [0.1, 1]$ are random values, $W \in [0.1, 0.5]$ is the inertia weight, and $C_1, C_2 \in [1.2, 2.0]$ are the learning factors that take uniformly distributed random values in their defined ranges.

After the particle updates its position and velocity, it has to update its *pbest* as well. The *pbest* value will be replaced with the new position only if the new position dominates *pbest*, or if both are mutually non-dominate. The crowding archive is then updated with new non-dominated particles, if found, subject to the crowding restriction. Finally, the reference point will be updated if needed. The final result of the optimisation will be the content of the crowding archive when the run of SDMOPSO is complete. The pseudo-code of the SDMOPSO Algorithm is listed in Algorithm 3.1.

3.2.2 Why SDMOPSO?

SDMOPSO introduces the following improvements to the basic MOPSO:

- SDMOPSO enhances the approximation of the PF for a MOP by decomposing the original MOP into scalar aggregation problems and facilitating simultaneous optimisation of these scalar problems. As will be observed in the next section when it is compared to other MOPSOs.

- SDMOPSO associates every particle with a λ vector according to the best scalar aggregated fitness value. This will potentially enhance the distribution of the initial population and, together with the way the information is exchanged between the particles in the swarm, can eventually lead to saving processing time.

- SDMOPSO uses the crowding archive to retain the diversity of the leaders, and hence the distribution of the final solutions. This is implemented using crowding-based selection method to choose the solutions to be deleted or replaced when the archive is full.

- SDMOPSO conserve the simplicity of PSO while improving the MOPSO by better exploitation of the particles' local information. Each particle pre-processes its next move and exchanges the discovered information with the entire swarm in order to facilitate simultaneous optimisation of all scalar problems. This could also help mitigating premature convergence to local optima.

3.2.3 Experiments and results

The SDMOPSO method is tested on several standard problems defined in the test suite (Coello Coello *et al.* (2007)) - 9 representative problems were chosen (Schaffer, Fonseca, Kursawe, Viennet2, Viennet3, ZDT1-4, and ZDT6 . They cover diverse MOPs with convex, concave, connected and disconnected PFs. The method is then compared to MOEA/D (Zhang *et al.* (2009)) and OMOPSO (Sierra & Coello Coello (2005)).

jMetal framework (Durillo *et al.* (2006)) is used to implement MOEA/D and OMOPSO because it is a general framework that implements the state-of-art multi-objective algorithms. Each algorithm is run 30 times for each test problem. For the bi-objective problems each algorithm uses 300 iteration per run, and 150 individuals per generation. For the three-objective problems the corresponding values of 600 iterations and 300 individuals were used. All compared algorithms adopt real encoding and perform the same number of objective evaluations. For the sake of a fair comparison, the number of the non-dominated solutions found by each algorithm is limited to a fixed threshold (100 for bi-objective problems and 1000 for the three-objective problems). MOEA/D uses differential evolution crossover (DE) with (probability = 1.0) and (differential weight = 0.5), polynomial mutation with (probability = 1/number of decision variables), the mutation distribution index is equal to 20, and the neighbourhood size is set to 30. OMOPSO uses turbulence probability of 0.5. C_1, C_2 were set to a random value in the range [1.5, 2.0]. r_1, r_2 are set to a random value in [0, 1], and W to a random value in [0.1, 0.5]. SDMOPSO uses the parameters explained in the previous section and neighbourhood size $\mathcal{N}$ =30. Both OMOPSO and SDMOPSO use ϵ=0.0075, the crowding archive of size 150 for bi-objective problems and 300 for three-objective problems. The PF produced by each algorithm is the union of PFs after 30 runs ($PF_{approximated}$).

To validate our approach, two indicators are used for estimating the convergence and diversity of the solutions. The first performance indicator is a generational distance

(GD) (El-Ghazali (2009); Knowles & Corne (2002)), explained in Section 2.10.2. To apply this measure, all the objective values are scaled to be in the range [0,1].

The second indicator is the R-metrics (El-Ghazali (2009); Knowles & Corne (2002)). R-metrics is a hybrid indicator that simultaneously measures the convergence and diversity of the found solutions. R-metrics uses a set of utility functions u (e.g. aggregation function), which can be any scalar functions. In this thesis, we use a weighted Tchebycheff function with a sufficiently large number of evenly distributed normalized weighting vectors λ. R-metrics compare two reference sets in our experiments: A is PF_{true} related to the problem under test, and B is $PF_{approximated}$. Then the indicator is defined as follows:

$$I_{R_2}(A,B) = \frac{\sum_{\lambda\in\Lambda} u^*(\lambda,A) - u^*(\lambda,B)}{|\Lambda|} \tag{3.2}$$

where $\Lambda = \{\lambda_1,\ldots,\lambda_m\}$, $\lambda_i \in [0,1]$ and $\sum_{i=1}^{m}\lambda_i = 1$.

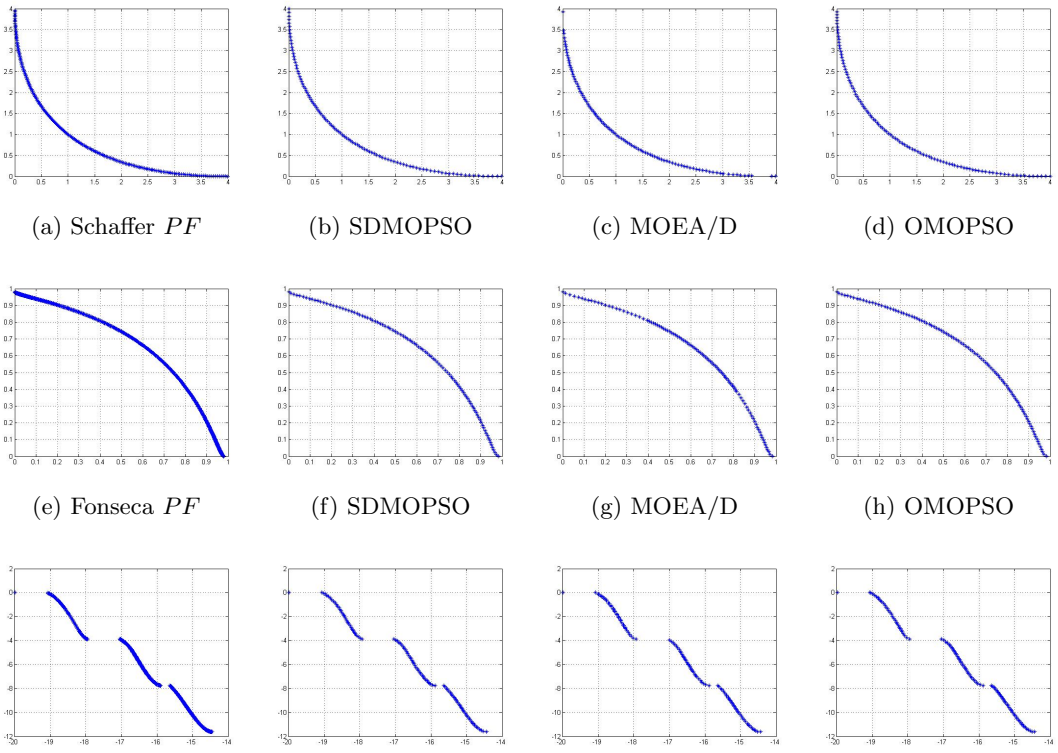
These two indicators are used to compare $PF_{approximated}$ with PF_{true}; their values are used here to quantitatively evaluate the performance of SDMOPSO in comparison with that of MOEA/D and OMOPSO.

Table 5.2 shows the average over 30 runs of the results obtained after applying GD and R-metrics measures, the last row presents the p-value resulted of applying Wilcoxon sign rank statistical test between the SDMOPSO and the other two methods. This test was selected as recommended in (Desmar (2006)) . Fig.3.1 and Fig.3.2 depict PF_{true} and $PF_{approximated}$ for the three algorithms under investigation.

3.2.4 Discussion

SDMOPSO divides the MOP into scalar aggregation problems which are solved simultaneously using PSO. The information exchange method proposed herein helps avoiding local optima without a need for applying any genetic operator and utilises the local information more effectively by facilitating simultaneous optimisation of all scalar problems. In order to maintain the diversity of the final solutions, SDMOPSO uses a crowding archive.

The previous use of PSO within the MOEA/D framework (Peng & Zhang (2008)) has several limitations. Every particle updates its position using its personal best and global best information without considering the neighborhood best, which can be

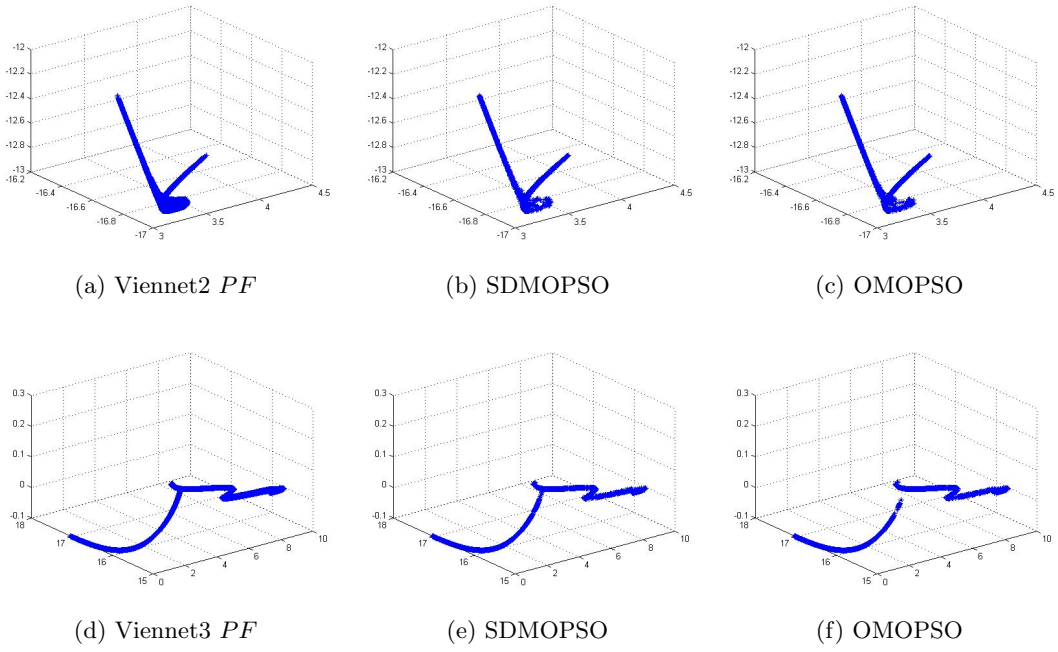


(a) Schaffer PF (b) SDMOPSO (c) MOEA/D (d) OMOPSO

(e) Fonseca PF (f) SDMOPSO (g) MOEA/D (h) OMOPSO

(i) Kursawe PF (j) SDMOPSO (k) MOEA/D (l) OMOPSO

Figure 3.1: (a, e, i) are the PF_{true} and the rest are the approximated ones



(a) Viennet2 PF (b) SDMOPSO (c) OMOPSO

(d) Viennet3 PF (e) SDMOPSO (f) OMOPSO

Figure 3.2: (a, d) are PF_{true} and the rest are the approximated ones

Table 3.1: Indicators values for the three methods applied on nine test problems: the values are presented as [GD,R-metrics]

Problem	**SDMOPSO**	**MOEA/D**	**OMOPSO**
Schaffer	[0.0165,0.00212]	[0.0242,0.0002]	[0.0164,0.0029]
Fonseca	[0.0038,2.94E-04]	[0.004,3.00E-05]	[0.0037,5.48E-04]
Kursawe	[0.0335,1.22E-03]	[0.0343,8.91E-04]	[0.0323,9.43E-04]
Viennet2	[0.0067,2.31E-10]	[0.049,3.38E-07]	[0.0062,7.06E-10]
Viennet3	[0.0096,8.65E-07]	[3.3616,5.76E-03]	[0.0107,5.51E-07]
ZDT1	[0.0044,0.004]	[0.0055,0.0044]	[0.0037,2.29E-03]
ZDT2	[0.0051,0.003]	[0.0044,0.0017]	[0.0038,0.0012]
ZDT3	[0.0043,0.003]	[0.014,0.0067]	[0.0043,0.0041]
ZDT4	[1.4319,0.3068]	[0.7714,0.2338]	[1.4329,0.3744]
ZDT6	[0.003,0.0013]	[0.0029,0.0012]	[0.0031,0.0011]
Average	[0.1519,0.0322]	[0.4271,0.0255]	[0.1517,0.0387]
Std	[0.4498,0.0965]	[1.0581,0.0732]	[0.4503,0.1179]
p-value	[-,-]	[0.1602,0.6953]	[0.4453,1]

an important asset for maintaining the diversity of the solutions and avoiding local optima. MOPSO/D uses a mutation operator, which can contribute to the complexity of the method and thus reduces the advantage of PSO over GA. MOPSO/D does not incorporate the non-dominance concept within the optimization process, which could potentially lead to premature convergence.

The results presented in this section show that OMOPSO, MOEA/D and SDMOPSO perform similarly on problems with two objectives (Schaffer, Fonseca, Kursawe, ZDT1-4 and ZDT6). When looking at three-objective problems, both SDMOPSO and OMOPSO outperform MOEA/D. However the statistical test over all datasets shows insignificant difference in the indicator values among all the methods. The advantage of the MOPSO-based methods in 3D MOPs could be explained by the fact that EA-based techniques offer advantages in problems where some structure exist in the decision space - the reproduction operators can exploit this structure very effectively. When, however, such a structure does not exist or is confounded by the interplay of several competing objectives, MOP heuristics aimed at uniform exploration of the solution space can perform better (Fleischer (2003)). The results of our experiments support this hypothesis.

For Viennet3, SDMOPSO seems to have better diversity than OMOPSO as OMOPSO does not fully cover the PF (Fig.3.2e and Fig.3.2f). For other test problems, SDMOPSO and OMOPSO perform similarly. The major algorithmic difference between SDMOPSO and OMOPSO is that OMOPSO uses mutation (Sierra & Coello Coello (2005)), whereas SDMOPSO does not apply any genetic operator. Mutation is usually regarded as turbulence that is beyond a particle's own control (Reyes-Sierra & Coello (2006)). The usage of mutation operator by PSO is generally justified because of a very high convergence speed of this method. Such convergence speed could be a disadvantage in solving MOPs, because it may lead to a false PF (Coello Coello *et al.* (2004)) due to falling into local optima. SDMOPSO, on the other hand, handles this issue by making every particle in the swarm pre-process its moves and to share this information with its neighbours. This results in a better exploitation of the local information, which alleviates the effect of premature convergence to local optima.

3.3 Clustering based Framework for Leaders Selection

Section 2.8.2 discussed the need to keep the generated solutions throughout the optimization process in an archive. A selection mechanism is necessary to select the leader(s) of the particles. Selection is a main step in evolutionary algorithms due to its strong impact on the search direction during the evolutionary process (Back (2002)). Various selection operators have been proposed for different algorithms, surveyed in (Back (2002)). Proportional selection is one of the most common selection strategies. Proportional selection assigns selection probabilities according to the relative objective value of individuals (Holland (1992)). Tournament selection chooses a set of n individuals from the population and then select the best individual from this set and copy it to the next generation; this process is iterated until the population of the next generation is fully selected. Linear ranking (Baker (1985); Grefenstette & Baker (1989)) maps individual's indices to selection probabilities using a linear function. A non-linear function may also be used, and the method is referred to as non-linear ranking selection (Michalewicz (1994)). (μ, λ) approach operates on offspring and tries to reduce it to the size μ by choosing the μ best individuals while $\mu + \lambda$ approach chooses the best μ individuals from a set that combines the original population and the offspring together (Back (2002)).

Selection implies the comparison among individuals. Most MOEA use the domination concept for comparison among individuals (Deb *et al.* (2002); Ghosh (2004); Zitzler & Thiele (1999)). GAs usually apply the selection operator on the current population, where individuals are selected and the non-dominated ones are qualified to survive and replace the parents in the next generation. MOPSOs, on the other hand, use an archive of leaders. The leaders are the set of the non-dominated individuals found so far. All leaders are qualified to guide the search. The selection operator is then applied to choose one leader from the archive (Al Moubayed *et al.* (2010); Coello Coello *et al.* (2004); Sierra & Coello Coello (2005)). The leader selection used are mostly dependent on the domination relation to determine the best individuals, where only the non-dominated solutions in the population are responsible of moving and evolving the remaining solutions in the space through the optimization process. This assumes that the non-dominated solutions are the closest solutions found so far to the Pareto front. This can be a strong assumption, especially with sparse datasets, which might

mislead the optimization process into local optima. In addition, selection operators are argued to have a tendency to reduce the population variance (Raghuwanshi & Kakde (2004)).

The majority of MOEA aim at finding non-dominated solutions as close as possible to the PF and to maintain the diversity of these solutions in the objective space without taking into account the diversity of these solutions in the solution space. This might result in discarding potentially important regions in the solution space from the optimization process.

Wang *et al.* (2009) used a uniform and orthogonal design for generating the initial population, ensuring thereby an evenly distributed and fair coverage of the solution space at least at the start of the optimization process. They also proposed a clustering method for selecting non-dominated solutions.

The leaders selection technique proposed in this section, however, goes further and defines a framework for leaders selection based on clustering the individuals in the current population in the solution space as well as in the objective space and then defining a corresponding relation between the clusters in both spaces. We propose a new technique for creating a set of solutions (archive) that covers most of the potentially good regions in both the solution space and the objective space; the leaders will then be selected from this archive.

The clustering based leaders selection (CLS) framework uses density based spatial clustering in order to create the clusters in both spaces. Density based clustering does not require a predefined number of clusters and build clusters of arbitrary shapes, which is advantageous to our framework in order to guarantee its general application to any optimisation problem. The individual is represented in the solution space by its decision variable vector and in the objective space by its objective values. Each individual is assigned into two clusters one in the objective space and the other in the solution space. Should two individuals fall in the same cluster in the solution space, it does not necessarily mean they belong to the same cluster in the objective space and vice versa.

Most real world problems have large number of decision variables. This raise a serious challenge for clustering in a multi-dimensional space with low number of samples. In this thesis this issue is solved using the principal component analysis (PCA). PCA

is applied in order to reduce the dimensionality in spaces of dimensions higher than three.

The CLS framework: Clustering based leaders selection operates on population of individuals. It uses Density Based Spatial Clustering (discussed in Section 3.3.2) to assign each individual into one solution space cluster and one objective space cluster. This clustering technique is chosen because of its ability to discover a dynamic number of clusters with arbitrary shapes and variable number of individuals in each cluster. This approach has a wide applicability and adaptivity with very few parameters to tune. The clustering takes place after ensuring that the individuals' solution and objective spaces are of dimensionality lower than four. For spaces with higher dimensions, PCA is employed to reduce the dimensions (discussed in the following section).

3.3.1 Principal Component Analysis

PCA is a mathematical procedure that applies an orthogonal projection to convert data of possibly correlated variables into a set of values of uncorrelated variables called principal components. The number of principal components is usually less than the number of original variables. The resultant first principal component has the highest variance, and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to the preceding components. In principle, PCA involves evaluating the mean $\overline{x}$ and the covariance matrix S of the data set and then finding the M eigenvectors of S corresponding to the M largest eigenvalues (Bishop *et al.* (2006)).

PCA is applied within the CLS framework when the number of variables in the search space is larger than 3. This is important as the clustering performance is usually sensitive to the dimensionality of the data.

3.3.2 Density Based Spatial Clustering

Through the optimization process the individuals of the population move gradually towards specific regions, which represent the Pareto front in the objective space and Pareto optimal set in the solution space. The population then is grouped into multiple number of clusters, where high density regions (groups of individuals) are considered as clusters and low density regions are considered as noise (Grira *et al.* (2005)).

Spatial clustering aims at finding similar spatial attributes between objects in the space. Dense regions that are near to each other are merged to form one cluster (Cohn *et al.* (2003)). Density based spatial clustering (DBSC) (Ester *et al.* (1996)) is a spatial clustering technique with few parameters to tune. The number of clusters generated by DBSC is dynamic, i.e. the number of clusters is not predefined but is determined by the locations of the objects in the space (Basu *et al.* (2002)), resulting in clusters with arbitrary shapes.

DBSC algorithm starts by randomly selecting a point n from the space. A neighborhood of n contains all the points in the space which exist in the circle surrounding n with a predefined radius ϵ, $dist(n, m) \geq \epsilon$. The neighborhood $N_\epsilon(n)$ is only created if it contains a minimum number of points $MinPts$; otherwise the point n is considered a noise point (outlier) and no cluster is considered. If the neighborhood contains $MinPts$ points or more $|N_\epsilon(n)| \geq MinPts$, a cluster is created. This process is repeated for every point in the defined cluster in order to check if there are more points can be placed in the cluster in order to expand it. Expanding one cluster is done by adding all the reachable points by the cluster's neighborhood. When the cluster cannot be expanded further, another point is selected from the space and the same process is repeated. The process terminates when each points belong to one cluster or labelled as a noise point. The distance between two points is calculated using a distance function $dist(x, y)$, where x, y are two points from the data space. $MinPts$, ϵ and $dist(x, y)$ are the only parameters to be defined in order to create the clusters (Ester *et al.* (1996)). Following is an outline of the DBSC algorithm:

Algorithm 3.2 DBSC

1: define a spatial dataset DS, $MinPts$, ϵ and $dist(x, y)$
2: **for** each point is not assigned to a cluster and is not labeled as noise n **do**
3: Find the neighborhood of n, $N(n)$
4: Place all points of the neighborhood in one cluster
5: **for** each point $\in N(n)$ **do**
6: Go to 3
7: Expand the Original Cluster if possible
8: **end for**
9: **end for**
10: Return the set of clusters found

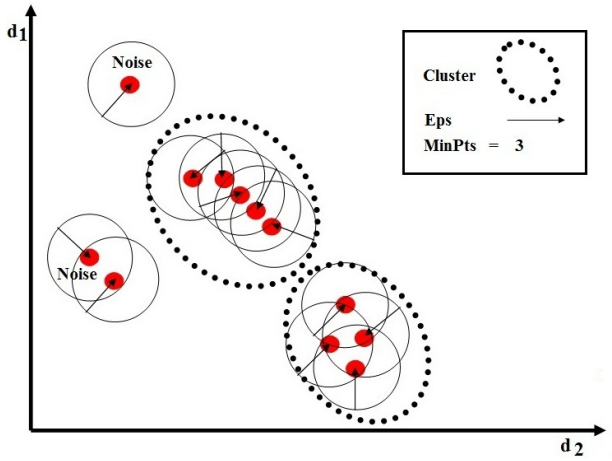


Figure 3.3: Example of clusters produced by Density Based Spatial Clustering

Fig. 3.3 demonstrates the final result of applying DBSC on a synthesized dataset.

3.3.3 The Algorithm

CLS aims at covering all the potentially good regions in both the objective and the solution spaces and maintaining a good level of diversity in both of them. This is done by clustering the individuals in the objective and solution spaces and incorporating the diversity information using ϵ dominance.

A dimensionality reduction technique needs to be used in order to reduce the number of the objectives and variables of each individual and to enhance the clustering quality. PCA is applied when needed as explained in Section 3.3.1. After PCA, the DBSC is applied to implement clustering.

The variable vector of each individual defines its location in the solution space while the objective vector defines the individual's location in the objective space. DBSC assigns all individuals into two sets of clusters using the individuals' images in both spaces.

At the end of the clustering phase, two sets of clusters are created; one in each space. Each individual has two images, one in the solution space (i.e. the variables vector) and one in the objective space (i.e. the objective values). After clustering in both spaces, each individual is assigned to a cluster in each space depending on its image in that space. This defines an indirect mapping between the two spaces. Should two individuals belong to the same cluster in one space, it does necessarily mean they

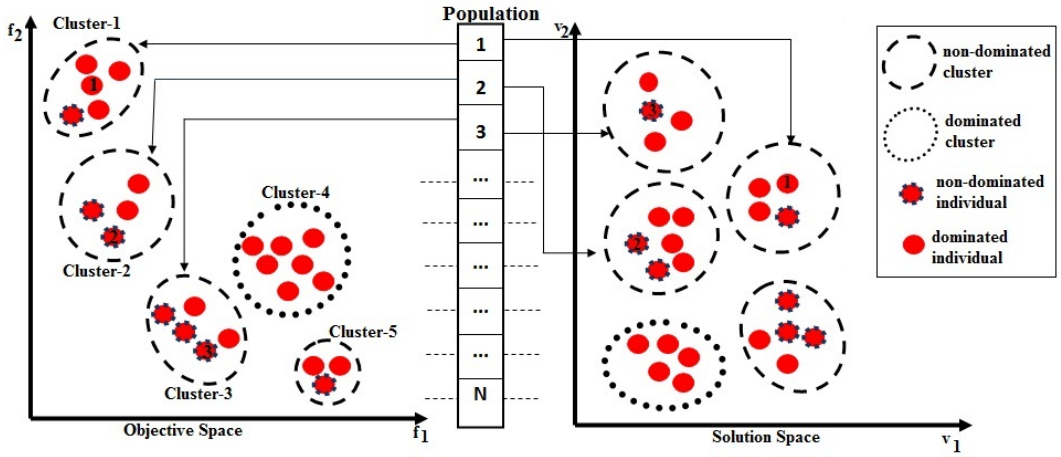


Figure 3.4: Example of the algorithm at work while mapping clusters.

belong to same cluster in the other space. Each individual is marked by an index in the population and each cluster in either space has a unique index. These added information facilitate mapping the individual into the clusters. Fig. 3.4 demonstrates this indirect relation.

A non-dominated cluster is a cluster in the objective space that contains at least one non-dominated individual. CLS exploits the relation between the non-dominated clusters in the objective space and the related clusters in the solution space. Following is a formal representation of the algorithm.

Let O be the set of clusters in the objective space and V the set of clusters in the solution space. A cluster $o \in O$ is a non-dominated cluster if and only if $\exists a \in o$ and $a \in PF$. The set of non-dominated clusters is then called $O^{'}$. Then we can define the function:

$$\phi(c) = \{v : v \in V and\ \exists X_a \in v : F_a \in c\} \tag{3.3}$$

where $c \in O$ and X_a is the image of an individual a in the solution space and F_a is the image of a in the objective space. For a set of clusters C in the objective space, the function $\Phi(C)$ is defined as:

$$\Phi(C) = \{\phi(c) : c \in C\} \tag{3.4}$$

The selection function Ψ is defined as follows:

$$\Psi : A \rightarrow \{0, 1\} \tag{3.5}$$

where A is the current population, and

$$\Psi(a) = \begin{cases} 1 & : F_a \in PF \\ 1 & : F_a \notin PF, F_a \in o, o \in O^{'}, \phi(o) \notin \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $\Gamma \subseteq V$ is the set of clusters selected so far. $\Gamma = \{c \ \in V : \exists \ a, X_a \in \ c, \Psi(a) = 1\}$.

Algorithm 3.3 CLS

```
1: if |X_a| ≥ 4 then
2:    for i=1 to N do
3:       PCA(X_{a_i})
4:    end for
5: end if
6: if |F_a| ≥ 4 then
7:    for i=1 to N do
8:       PCA(F_{a_i})
9:    end for
10: end if
11: [O,V]=Apply-DBSC
12: if |O|=0 and |V|=0 then
13:    Continue-Algorithm
14: else
15:    for all o ∈ O' do
16:       for all F_a ∈ o do
17:          if Ψ(a) = 1 then
18:             add a to leaders-archive
19:             add Λ(a) to set of selected clusters Γ
20:          end if
21:       end for
22:    end for
23: end if
```

Algorithm 3.3 outlines the algorithm to build the leaders' archive, where Continue-Algorithm skips the building of the archive and continues the evolutionary algorithm

which might happens in early stages of the optimization process where the individuals are sparse, and $\Lambda(a) \in V : X_a \in \Lambda(a)$.

The leaders archive is characterized with a limited size where it maintains the diversity in both the solution space and the objective space. ϵ dominance is used here to retain the diversity when the archive reaches its maximum size, taking into account that at least one individual from each non-dominated cluster in the objective space as well as one from the related clusters in the solution space must be maintained in the archive. Maintaining the clusters' representatives would not exceed the maximum size of the archive as the size of the archive is set to the population size and the maximum number of clusters possible is smaller than half the population size (because each cluster contains at least 2 solutions).

DBSC is used to implement the clustering step. DBSC as explained in the previous section, assigns the adjacent solutions recursively to the same cluster. When solving MOP with continuous PF the algorithm converges, at the end of the optimization process, to a state where the solutions in the approximated PF are assigned to one cluster. At the beginning several clusters might be found and gradually through the process the solutions will converge and fall into one non-dominated cluster. When solving MOP with disconnected PF, the algorithm converges and the number of the non-dominated clusters generated will be approximately similar to the number of the disconnected parts of the PF.

CLS can be integrated easily with any MOEA. For MOEA that employs selection operator on the whole population (e.g. GA), each individual selected from the population is mated with another individual selected from the leaders archive. While for MOEAs that select the leaders from an archive (e.g. MOPSO) the archive is replaced by the leaders archive proposed in this work.

3.3.4 Experiments and Results

To verify our method, CLS is integrated with NSGAII to form a new method called NSGAII based on Clustering (NSGAII/C). This is done by customizing NSGAII where one parent to mate is selected from the population but the second parent is selected from the leaders archive created by CLS. CLS takes place at the beginning of each iteration till the process meets a stopping condition.

NSGAII/C is tested on several standard test problems defined in the test suite (Coello Coello *et al.* (2007)). The selected test problems are ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, Viennet2, and Viennet3. These were chosen as they cover diverse MOPs with convex, concave, connected and disconnected PFs. ZDT1-ZDT4 and ZDT6 are two-objective problem with 30 decision variables. Viennet2 and Viennet3 are three-objective problems with 2 decision variables. NSGAII/C uses PCA to reduce the solution space's dimensionality when solving ZDT1-4 and ZDT6 while PCA is absent when solving Viennet2 and Viennet3. Results obtained using NSGAII/C are then compared to these obtained using the original NSGAII.

NSGAII uses a Simulated Binary Crossover (SBX) with probability equal to 0.9, and a polynomial mutation with probability equal to the inverse of the number of decision variables (i.e. 1/30 and 1/2), the mutation and the crossover distribution indexes are equal to 20 (Deb *et al.* (2002)).

NSGAII/C uses the same parameters as NSGAII; also it uses clustering implemented by DBSC. Before applying DBSC, the variables and objectives are normalized. ϵ is set to 0.1, $MinPts$ is set to 3, and Euclidean distance function, $dist(x_i, y_j)$, is used.

The approximation of the Pareto fronts produced by each algorithm is the combination of PFs obtained after each of 30 experimental runs. For the bi-objective problems each algorithm uses 300 individuals per generation. For the three-objective problems the corresponding values of 600 individuals were used. All compared algorithms adopt real encoding and perform the same number of objective evaluations of 60000.

Figures 3.5-3.11 depict the two PF approximations produced by NSGAII and NSGAII/C for the seven MOPs under investigation, where parts of the figure is zoomed in a separate square to show the details of the approximated PFs for easier comparison.

To validate our approach, three indicators are used for measuring the convergence and diversity of the solutions: I_{IGD}, I_{GD}, and the Set Coverage (I_{SC}).

$I_{SC}(A, B)$ calculates the percentage of solutions in B that are dominated by at least one solution in A, where A and B are two approximation of the PF produced by a two specific method (Zitzler *et al.* (2003b)).

$$I_{SC}(A, B) = \frac{|b \in B, \exists a \in A : a \succ b|}{|B|} \tag{3.7}$$

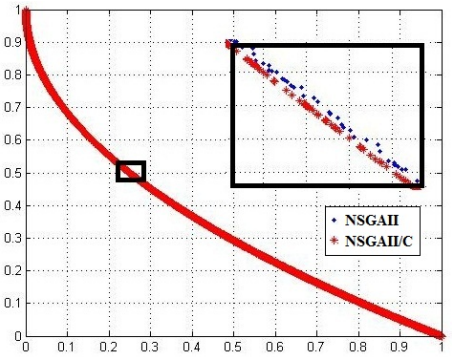


Figure 3.5: The two PF approximations for ZDT1.

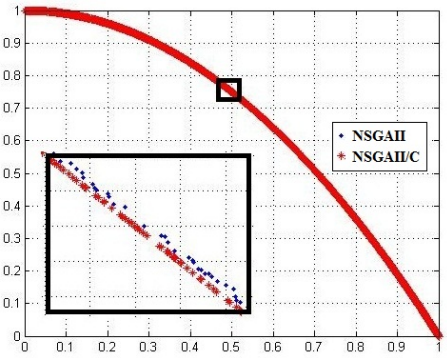


Figure 3.6: The two PF approximations for ZDT2.

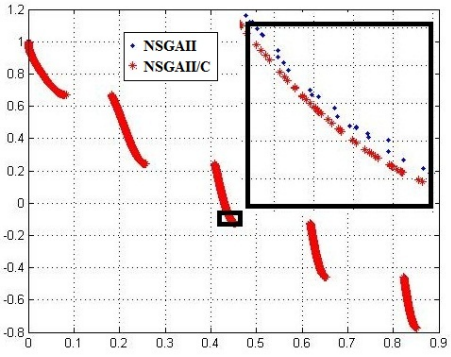


Figure 3.7: The two PF approximations for ZDT3.

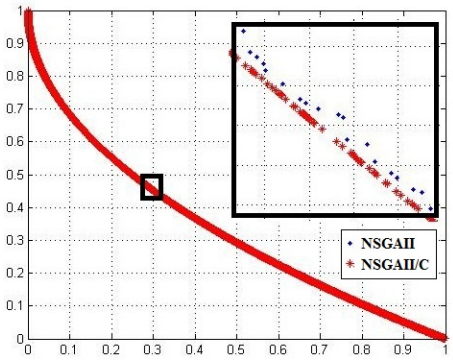


Figure 3.8: The two PF approximations for ZDT4.

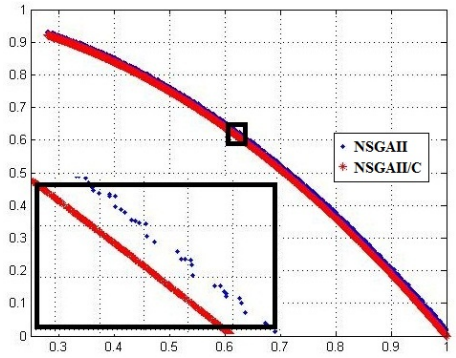


Figure 3.9: The two PF approximations for ZDT6.

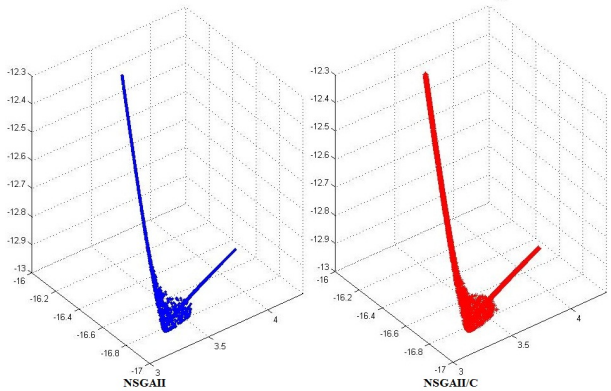


Figure 3.10: The two PF approximations for Viennet2.

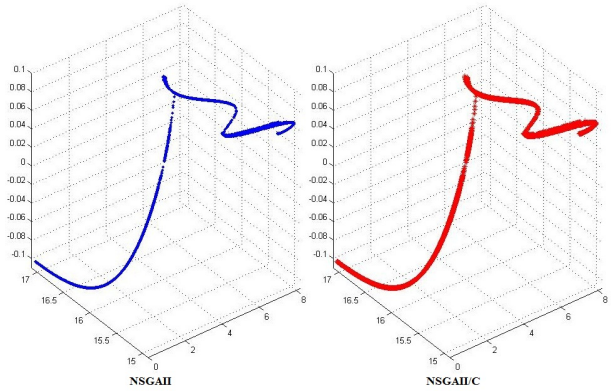


Figure 3.11: The two PF approximations for Viennet3.

$I_{SC}(A, B)$ does not necessarily equal $1 - I_{SC}(A, B)$. $I_{SC}(A, B)$ equals 1 when all the solutions in B are dominated by some solutions in A, while $I_{SC}(A, B) = 0$ when no solution in B is dominated by any solution in A.

The first two measures are unary indicators as they compare the obtained solutions with the Pareto front, while the third one is a binary indicator as it compares solutions obtained by two optimization methods.

Table 3.2 and Table 3.3 show the results obtained after applying I_{IGD} and I_{GD} measures. Fig. 3.12 shows box plots to compare the Euclidean distance values obtained while calculating I_{IGD}, using Eq. 2.17, between the two methods. There does not seem to be any significant differences on all the problems except for ZDT6 which can be explained by a better coverage of NSGAII/C of the PF compared to NSGAII. Fig. 3.13 demonstrates the results obtained by applying I_{SC} measure.

Table 3.2: Inverted Generational Distance results for the two algorithms

Problem	NSGAII	NSGAII/C
ZDT1	3.04E-05	1.14E-05
ZDT2	3.50E-05	1.19E-05
ZDT3	3.76E-05	1.75E-05
ZDT4	5.62E-05	1.16E-05
ZDT6	4.38E-04	1.50E-05
Viennet2	4.74E-05	4.81E-05
Viennet3	1.66E-05	1.73E-05

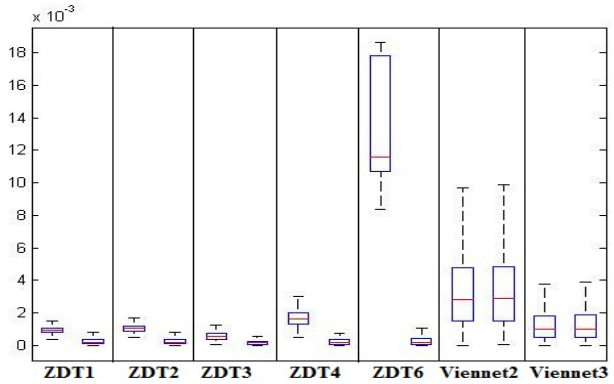


Figure 3.12: Box plots to demonstrate the average distance between each solution of the actual Pareto front and its closest solution in the approximated PF. NSGAII results are at the left of each sub plot, while NSGAII/C results are at the right side.

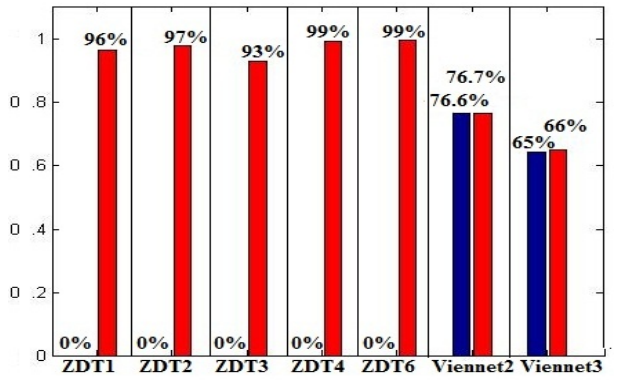


Figure 3.13: Coverage set results for the two algorithms. For each problem 2 bars are drawn, the left bar represent the percentage of solutions produced by NSGAII that dominate these produced by NSGAII/C, the right bar represents the opposite percentage.

Table 3.3: Average Generational Distance results for the two algorithms

Problem	NSGAII	NSGAII/C
ZDT1	4.26E-05	2.59E-05
ZDT2	2.88E-05	9.18E-06
ZDT3	4.78E-05	3.86E-05
ZDT4	1.26E-03	2.65E-05
ZDT6	5.47E-04	1.07E-04
Viennet2	3.45E-05	3.27E-05
Viennet3	9.87E-06	1.00E-05

3.3.5 Discussion

In this section, a new framework for leaders selection was introduced within a multi-objective evolutionary optimization process. The method is based on simultaneous clustering in the solution and objective space and then mapping the two cluster sets to create a leader archive that potentially approximate better Pareto front. CLS uses DBSC as a priori knowledge of the number of clusters in the solution or the objective space are usually unavailable. DBSC cannot cope with sparse data in high dimensional space and small number of samples, PCA is utilized for dimensionality reduction within the CLS framework to enhance the quality of clustering.

The proposed algorithm, is tested on 5 two-dimensional MOPs and 2 three-dimensional MOPs. The PF approximations shown in Fig 3.5- 3.9 demonstrate the dominance of solutions obtained by NSGAII/C over the solutions obtained by NSGAII. The PF approximation produced by NSGAII/C is also closer to the actual PF. Fig.3.10 and Fig.3.11 show a very similar solutions produced by both algorithms where no strong domination exists. These results are confirmed by different analytical measures. Table 3.3 presents the generational distance measure, which shows that NSGAII/C is closer to the actual PF than NSGAII for all 2-D MOPs, whilst there is no difference for 3-D MOPs. Table 3.2 presents the spread of the solutions obtained and shows better spread for NSGAII/C over NSGAII. Fig. 3.13 demonstrates the set coverage measure of the two methods, and shows NSGAII/C solutions to highly dominate the majority of solutions produced by NSGAII.

CLS is a general framework, different clustering algorithms may be used instead of

DBSC, PCA can be substituted by any other dimensionality reduction technique, and the mapping definition can also be replaced by any other mapping function(e.g. one that incorporates prior knowledge about the MOP).

Clusters in the objective space may also be ranked and a decision maker can be involved in directing the optimization processes toward regions according to this prior knowledge. Eliminating the worst ranked clusters and re-sampling new solutions in regions of interest is another approach that worth further investigation.

A drawback of the clustering approach is that it can be computationally expensive. However, the idea of mapping both objective and solution spaces is very attractive and potentially very useful. The next section presents an algorithm that takes a simpler approach for this mapping.

3.4 D^2MOPSO: MOPSO based on Decomposition and Dominance

Designing effective measures for diversification of solutions to a MOP and for their uniform distribution along the Pareto optimal front is a challenging research problem (Reyes-Sierra & Coello (2006)). multi-objective metaheuristics can be classified into four categories: decomposition-based (scalar), criterion-based, dominance-based, and indicator-based approaches (discussed in detail by El-Ghazali (2009)). It would be interesting therefore to ascertain whether/how these approaches can be combined or enhanced to achieve a better preservation of solution diversity, and as a consequence, a closer approximation of the Pareto optimal front. Hybridising different search approaches is reported in (Zhou *et al.* (2011)).

D^2MOPSO utilizes a hybrid approach of dominance (e.g., Sierra & Coello Coello (2005)) and decomposition (e.g., Zhang & Li (2007)). This approach achieves fast convergence to the true Pareto Front without resorting to the use of genetic operators (e.g., mutation). Also, a better exploitation of the information discovered during the search enables the suggested multi-objective PSO approach to be applied to problems that necessitate complex system optimisation. D^2MOPSO introduces a bounded leaders' archive based on the crowding distance in both objective and solution spaces to store the non-dominated particles. The leaders are then selected from the archive using the aggregation value as the selection criterion.

The main difference between SDMOPSO and D^2MOPSO is in how they incorporate the decomposition concept. SDMOPSO uses decomposition to define the neighbourhoods and only exchange information locally, while D^2MOPSO uses decomposition for leader selection only and does not define a neighbourhood but utilizes global information instead. In addition D^2MOPSO uses the newly developed bounded leaders' archive.

3.4.1 Archiving based on Crowding Distance in Objective and Solution Spaces

Dominance-based approaches to multi-objective optimisation use the concept of dominance and Pareto optimality to guide the search process. The majority of dominance-based MOPSOs use a fixed-size leaders' archive to store trade-off solutions found through the optimisation process (Coello Coello *et al.* (2007)). Thus, the selected leaders influence significantly the optimisation process; maintaining the archive and selecting the leaders therefore are main challenges for MOPSO.

MOPSO aims at minimizing the distance between the solutions in the archive and the true PF, whilst maximizing the diversity of these solutions in the objective space. Several density estimators are employed to tackle these challenges. Some commonly used techniques are discussed in Section 2.8.2 and also listed in (El-Ghazali (2009)).

Most archiving techniques maintain the quantity and diversity of the solutions in the objective space without taking into account the diversity of these solutions in the solution space, which might result in discarding potentially important regions there. The previous section tackled this issue using an approach based on clustering both objective and solution spaces. The major drawback of this approach is its computational complexity. The archiving technique suggested in this section provides a more efficient alternative that uses a density estimator in both the solution and the objective spaces.

Each particle has two crowding distance coordinates one in each space. Therefore, the crowding distance is a two-dimensional vector where the first dimension characterizes crowding in the objective space, and the second in the solution space. We use crowding distance (kernel density estimator, Section 2.8.2.1) defined as follows:

$$CD(\overrightarrow{p_i}) = \Big(\sum_{j=1}^{AS} \parallel \overrightarrow{p_i}, \overrightarrow{p_j} \parallel_{\Omega}, \sum_{j=1}^{AS} \parallel F(\overrightarrow{p_i}), F(\overrightarrow{p_j}) \parallel_{\Delta} \Big) \tag{3.8}$$

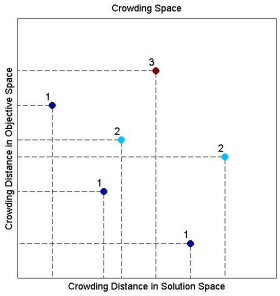


Figure 3.14: Dominance-based ranking for the non-dominated solutions of the leaders' archive using the crowding distance values in both solution and objective spaces. X-axis is the crowding distance in the solution space, Y-axis is the crowding distance in the objective space. The numbers next to each particle represents its rank. In this example the particles ranked with 3 are the best.

where AS is the size of the archive, $\overrightarrow{p_i}$ is the particle i's decision variable vector. $CD(\overrightarrow{p_i})$ is a vector of the crowding distances in the solution (Ω) and objective (Δ) spaces.

The crowding distance is only calculated when the maximum archive size is exceeded, and a replacement of some particles is needed. The elimination process starts by crowding the particles in both spaces. The elimination then considers the particles' two crowding distances in order to decide whether the particle to be removed or substituted.

A domination relationship and dominance-based ranking are applied to the created crowding space. The particle with the worst rank is then replaced, with one selected randomly in the case of a tie. This is used in many MOEAs to sort the solutions in the objective space (Zitzler *et al.* (2003a)). Fig. 3.14 demonstrates an example of the dominance-depth ranking used. The mutually non-dominated solutions of the leaders' archive are ranked in the crowding space using their crowding values.

Algorithm 3.4 outlines the proposed archiving algorithm, where the operator $r(a)$ assigns a ranking value *rank* to the set a, CD is defined in Eq. 3.8, and Φ is the empty set.

Algorithm 3.4 Dominance-depth ranking in the crowding space

if Size(leadersArchive) $\geq$ MaxSize **then**
  for p_i in leadersArchive **do**
    $CD(p_i)(Eq.3.8)$
  end for
  $temporaryArchive = leadersArchive$
  $rank = 0$
  while $temporaryArchive \neq \Phi$ **do**
    generate set a, containing the nondominated particles in $temporaryArchive$
    assign $rank$ to each individual in a
    $rank = rank + 1$
    $temporayArchive = temporaryArchive \setminus a$
  end while
  replace the particle with the worst rank.
end if

3.4.2 The algorithm

Decomposition assists the optimisation process to find potential solutions that are evenly distributed along the PF (Li & Zhang (2009)). By associating each particle with a distinct aggregation problem (i.e. λ value), the direction of exploration activity of each particle is focused on a specific region in the objective space and is aimed at reducing the distance to the reference point.

Substituting entirely the dominance approach with decomposition in MOPSO (i.e. using the aggregation value instead of dominance as the leaders' selection criterion) might lead to premature convergence as each particle is strictly directed to one destination. At some point during the optimisation process, the particles would be unable to update their positions and personal best memory as the local best and neighborhood information become static. In addition, solving a MOP with complicated PF raises a serious challenge as some λ vectors direct the corresponding particles to unattainable areas. In such cases, part of the swarm would be exploring undesirable regions in the objective space for a considerable number of evaluations. Fig 3.15 demonstrates this problem where only eight out of twenty particles are directed towards the true PF. One may suggest adjusting the initialization of λ vectors to cover only attainable regions. This solution, however, only works if the true PF is known a-priori, which is not the

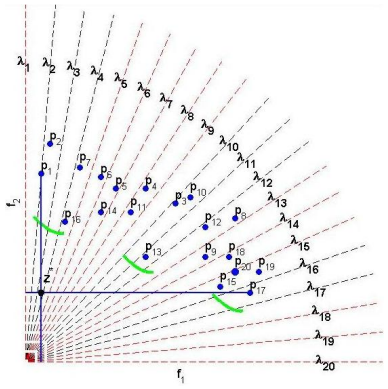


Figure 3.15: Swarm of 20 particles in a sample objective space. When only decomposition is used 8 particles are directed to promising regions in the space, the remaining 12 are directed to unpromising ones, i.e. 60% of the swarm is wasting the search effort.

case for most, if not all, real-life problems.

Another limitation of decomposition relates to how it operates in high-dimensional objective spaces. It struggles to produce a sufficient number of non-dominated solutions that cover the entire PF as the space to be covered by the swarm/population using λ vectors grows exponentially with the number of dimensions. This requires the decomposition-based approaches to use a large swarm/population in order to offer a good PF coverage, increasing therefore the number of necessary function evaluations, which can be a disadvantage for real-life problems with expensive or difficult to obtain evaluations.

To overcome all these drawbacks within MOPSO framework, D^2MOPSO integrates both dominance and decomposition. The bounded leaders' archive, Section 3.4.1, uses dominance to store only non-dominated particles. The personal best values are updated, and the leaders are selected using the decomposition's aggregation function.

D^2MOPSO uses PBI (Section 2.6.1) to transform the optimisation objective defined by Eq. 2.4 into N scalar optimisation problems, where N is the swarm size. By changing the weights and using the reference point defined above, Pareto optimal solutions may be approximated.

The following steps summarize D^2MOPSO:

Initialization: D^2MOPSO starts by initializing the swarm with N particles and

N λ vectors. Every particle is assigned a unique λ vector that gives the best aggregated fitness value (e.g., minimum in case of minimization problem) for the initialized particle. The initial value of the particle's memory $pbest_i$ is its own information ($\overrightarrow{pbest_i} = \overrightarrow{x_i}$) as it lacks any exploration experience at the beginning of the search process. The initial velocity of the particle is set to zero ($\overrightarrow{v_{i_0}} = 0$). The leaders' archive is set to a fixed size, and is initialized by the non-dominated particles in the swarm. The reference point $\overrightarrow{z^*}$ is the vector in the objective space with the best objective values found so far.

Evolution: During this phase D^2MOPSO goes through a pre-set number of iterations. At iteration (t), the particle determines the next move by calculating the new velocity and new position using Eq. 3.9 and Eq. 2.2, which involve $\overrightarrow{pbest}$ and the information about a global leader selected from the leaders' archive.

$$\begin{aligned} \overrightarrow{v_i}(t+1) &= W * \overrightarrow{v_i}(t) + C_1.r_1.(\overrightarrow{x_{pbest_i}} - \overrightarrow{x_i}(t)) \\ &+ C_2.r_2.(\overrightarrow{x_{lbest_i}} - \overrightarrow{x_i}(t)) \end{aligned} \tag{3.9}$$

where $\overrightarrow{pbest_i}$ is the personal best performance of $particle_i$, $\overrightarrow{lbest_i}$ is a leader selected from the archive, $r_1, r_2 \in [0, 1]$ are uniformly distributed random variables, $W \in [0.1, 0.5]$ is the inertia weight, and $C_1 = C_2 = 2.0$ are the learning factors. These parameters are defined following other recent MOPSOs (Al Moubayed *et al.* (2010); Martínez & Coello Coello (2011); Peng & Zhang (2008); Sierra & Coello Coello (2005)).

In order to ensure the decision variables fall into the predefined boundaries in the solution space, after each update their values are checked as follows:

$$(\overrightarrow{x_i}^d, \overrightarrow{v_i}^d) = \begin{cases} (\overrightarrow{min_d}, -\overrightarrow{v_i}^d) & if\ \overrightarrow{x_i}^d < \overrightarrow{min_d} \\ (\overrightarrow{max_d}, -\overrightarrow{v_i}^d) & if\ \overrightarrow{x_i}^d > \overrightarrow{max_d} \end{cases} \tag{3.10}$$

where i is the particle index, d is the index of the decision variable within the decision variables vector. $\overrightarrow{min_d}$ and $\overrightarrow{max_d}$ are the lower and upper boundaries of decision variable d respectively.

During leader selection (Algorithm 3.5 , where $\overrightarrow{lbest_i}$ is the selected leader for the corresponding particle i) each particle selects the leader that gives the best aggregation value using the particle's λ and the aggregation function in Eq. 2.10.

Algorithm 3.5 Leaders' Selection

1: **for** $i = 2$ to N **do**
2: $\overrightarrow{lbest_i} = \overrightarrow{lbest_1}$ % intialize $\overrightarrow{lbest_i}$ with the first local best.
3: **for** $j = 2$ to Size(leaders' archive) **do**
4: **if** $g(\overrightarrow{lbest_i}|\lambda_i, \overrightarrow{z^*}) > g(\overrightarrow{lbest_j}|\lambda_i, \overrightarrow{z^*})$ **then**
5: $\overrightarrow{lbest_i} = \overrightarrow{lbest_j}$
6: **end if**
7: **end for**
8: Select $lbest_i$ as leader for particle i
9: **end for**

After the particle updates its position and velocity, it has to update its $\overrightarrow{pbest_i}$ as well. $\overrightarrow{pbest_i}$ is replaced only if the new aggregation value is better:

$$\begin{aligned} if \quad & g(\overrightarrow{pbest_i}|\lambda_i, \overrightarrow{z^*}) > g(\overrightarrow{x_i}|\lambda_i, \overrightarrow{z^*}) \\ \text{then} \quad & \overrightarrow{pbest_i} = \overrightarrow{x_i} \end{aligned} \tag{3.11}$$

The leaders' archive is then updated with any new non-dominated particles subject to the crowding restriction explained in Section 3.4.1. The reference point is updated when a better objective value is found. When a particle updates its position, the new position is checked against $\overrightarrow{z^*}$ and update it if necessary.

$$if \quad \overrightarrow{z_j^*} < \overrightarrow{f_j(x_i)} \text{ then } \overrightarrow{z_j^*} = \overrightarrow{f_j(x_i)} : j \in [1, .., m] \tag{3.12}$$

Finally, the external archive, which contains all the non-dominated solutions found during the optimization process, is updated to contain the new non-dominated particles. The use of the external archive is optional as it is not involved in the evolution process, but it is recommended as it may contain better coverage and distributed solutions than the leaders' archive.

Termination: The algorithm terminates when the maximum number of iterations is reached. The content of the external archive is used to approximate the PF. If the external archive is not used, then the leaders' archive is considered.

Algorithm 3.6 lists a pseudo-code for D^2MOPSO, where $CheckBoundaries$ validates the decision variables and adjust them when necessary.

Algorithm 3.6 D^2MOPSO

1: Initialize the swarm with N particles and N λ vectors
2: **for** $i = 1$ to N **do**
3: assign the particle i to the λ vector that gives the best aggregation value
4: initialize velocities $\overrightarrow{V} = \{\overrightarrow{v_1}, \ldots, \overrightarrow{v_N}\}$ and $\overrightarrow{pbest_i}$
5: Initialize leaders' archive, external archive and $\overrightarrow{z^*}$
6: **end for**
7: **for** $t = 1$ to $MaxIterations$ **do**
8: **for** $j = 1$ to N **do**
9: Select $\overrightarrow{lbest_j}$ (Algorithm 3.5)
10: update Velocity, $\overrightarrow{v_j}(t+1)$ (Eq. 3.9)
11: update position, $\overrightarrow{x_j(t+1)}$ (Eq. 2.2)
12: CheckBoundaries($\overrightarrow{x_j}(t+1)$) (Eq. 3.10)
13: evaluate the new position (The corresponding problem fitness function)
14: update $\overrightarrow{pbest_j}$ (Eq. 3.11)
15: update leaders archive (Algorithm 3.7 which uses Algorithm 3.4)
16: update $\overrightarrow{z^*}$ (Eq. 3.12)
17: update external archive
18: **end for**
19: **end for**
20: Return the final result in the external archive

D^2MOPSO can solve both constrained and unconstrained continuous MOPs. An additional step is required when creating and updating the leaders' archive to accommodate constrained problems. The constraints are evaluated for each particle so that the leaders' archive update process is biased towards particles which do not violate the constraints (or breach the constraints to a lesser degree).

Algorithm 3.7 outlines the update of the leaders' archive with a new particle $\overrightarrow{S}$, where $Size$ is the size of leaders' archive, $breachConst$ checks if the particle has violated the constraints, $constraints$ evaluates the constraints; $valid(\overrightarrow{S})$ is correct if $\overrightarrow{S}$ has caused the removal of at least one particle from the archive or if it was not dominated by any other particle.

3.4.3 Novelty of D^2MOPSO

Dominance and decomposition are commonly used approaches in multi-objective evolutionary algorithms (Coello Coello *et al.* (2007); Deb *et al.* (2002); Li & Zhang (2009); Sierra & Coello Coello (2005)), but, to the best of our knowledge, they have mostly been used separately. Nasir *et al.* (2011) introduced the concept of a fuzzy dominance and only used decomposition when one solution fails to dominate the other in terms of fuzzy dominance level. D^2MOPSO is designed to take advantage of both concepts so that decomposition is used to select the leaders from a dominance-based archive. D^2MOPSO maintains the algorithmic simplicity of MOPSO by not utilizing any genetic or sampling operators. D^2MOPSO also uses a novel archiving technique that maintains diversity in both the objective and the solution spaces. Table 3.4 compares among five state-of-the-art decomposition-based MOEA.

3.4.4 Selected Test Problem

To test D^2MOPSO, I included the same problems used to test SDMOPSO with additional 17 other problems. This is due to D^2MOPSO being the more mature approach for decomposition and MOPSO, so it has to be rigorously tested.

D^2MOPSO is tested on 27 (5 constrained and 22 unconstrained) standard MOPs. The selected test problems cover diverse MOPs with convex, concave, connected and disconnected PFs, with two and three optimisation objectives. These problems were frequently used to verify the performance of several algorithms in the field of multi-objective optimisation (Al Moubayed *et al.* (2010, 2011); Coello Coello *et al.* (2007);

Deb *et al.* (2002); Li & Zhang (2009); Martínez & Coello Coello (2011); Nebro *et al.* (2008); Sierra & Coello Coello (2005)).

The following unconstrained bi-objective problems are selected: Shaffer (Deb & Agrawal (1994)), Fonseca (Fonseca & Fleming (1998)), Kursawe (Kursawe (1991)) in addition to the bi-objective version of WFG toolkit (WFG1-8 and WFG9) proposed in (Huband *et al.* (2005)). For three-objective problems, the following MOPs are used: Viennet2 and Viennet3 (Vlennet *et al.* (1996)), in addition to the DTLZ family (DTLZ1-6 and DTLZ7) proposed in (Deb *et al.* (2005)), which cover scalable MOPs with the number of decision variables of 7, 12, 12, 12, 12, 12, and 22 respectively.

To cover constrained bi-objective MOPs: three bi-constraints problems(Srinivas (Srinivas & Deb (1994b)), Constr_Ex (Deb *et al.* (2002)) and Tanaka (Tanaka *et al.* (1995))) are used in addition to the six- and eleven-constraint problems Osyczka2 (Osyczka & Kundu (1995)) and Golinski (Kurpati *et al.* (2002)) respectively. A three-objectives three-constraint problem (Viennet4 (Vlennet *et al.* (1996))) is also examined.

3.4.5 Experimental Setup

D^2MOPSO is compared to MOEA/D (Zhang *et al.* (2009)), dMOPSO (Martínez & Coello Coello (2011)) and OMOPSO (Sierra & Coello Coello (2005)) [1].

Thirty independent runs are performed for each test problem. For the bi-objective problems, 300 iterations per run and 150 particles per generation are used for all algorithms. For the three-objective problems, 600 iterations and 600 individuals are used. All algorithms under comparison adopt real encoding, perform the same number of objective function evaluations and use the same aggregation function with $\theta = 5$.

MOEA/D uses the differential evolution crossover (DE) (probability = 1.0 and differential weight = 0.5), polynomial mutation (probability = 1/number of decision variables), the mutation distribution index is equal to 20, and the neighbourhood size is set to 30.

dMOPSO sets the age threshold to 2; C_1, C_2 are assigned random values in the range [1.2, 2.0]. It uses a global set of size N, where N is the swarm size (the number of λ vectors): $N = 150$ for bi-objective problems, and $N = 600$ for three-objective ones.

[1] jMetal Framework Durillo *et al.* (2006) is used to implement MOEA/D and OMOPSO. dMOPSO implementation was provided by the authors.

OMOPSO uses turbulence probability of 0.5. C_1, C_2 were set to random values in the range [1.5, 2.0], ϵ-crowding archive with ϵ=0.0075 and leaders' archive of size N.

Both OMOPSO and dMOPSO set r_1, r_2 to random values in $[0, 1]$, and w to a random value in [0.1, 0.5]. [1]

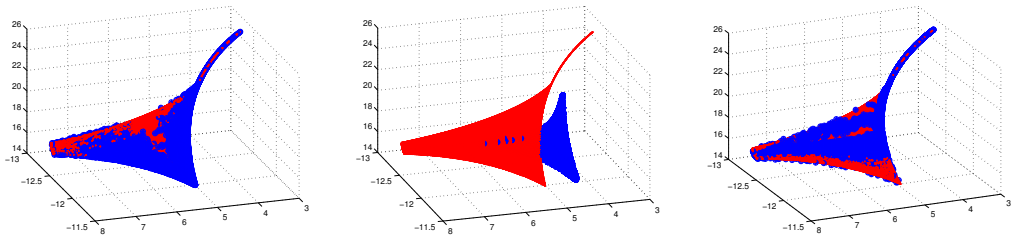
D^2MOPSO uses the parameters explained in the previous section with AS equals to 100 for the bi-objective problems and to 300 for the three-objective problems.

3.4.6 Performance Metrics

To validate our approach, three indicators: I_{IGD}, I_ϵ, and hypervolume, which estimate the convergence and diversity of the solutions, are used.

Table 3.4: A comparison among the decomposition-based MOEA under study

	MOEA/D	MOPSO/D	SDMOPSO	dMOPSO	D^2MOPSO
Decomposition	x	x	x	x	x
Dominance	-	-	x	-	x
Mutation	x	x	-	-	-
Memory reinit.	-	-	-	x	-
nbest	x	x	x	-	-
lbest	-	-	-	x	x
Leaders' archive	-	x	x	x	x



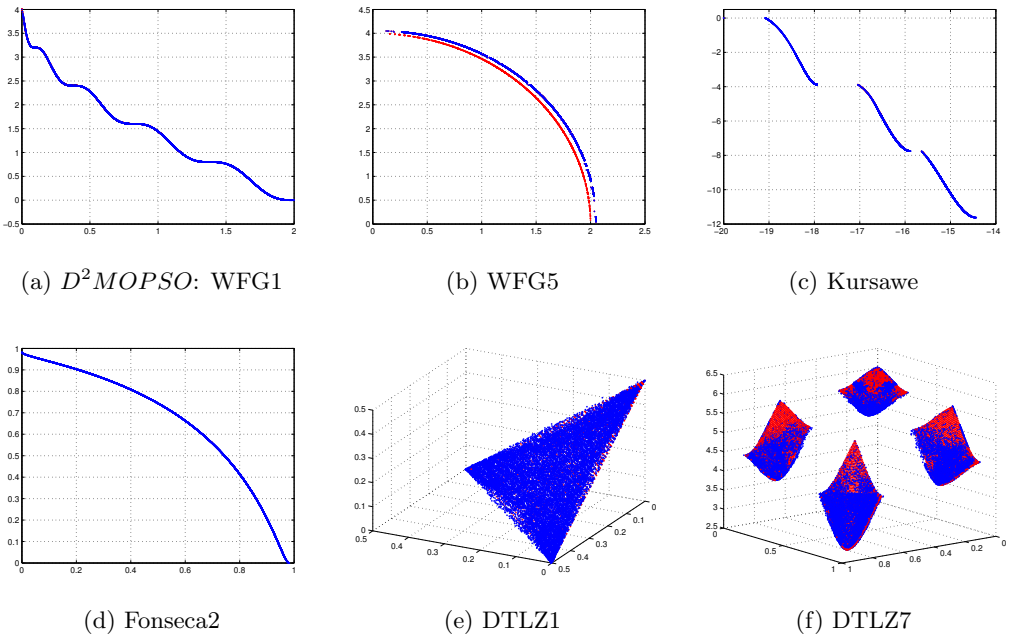
(a) Viennet4: D^2MOPSO (b) MOEA/D (c) OMOPSO

Figure 3.16: Plot of the non-dominated solutions with the lowest IGD values in 30 runs of D^2MOPSO, MOEA/D and OMOPSO for solving Viennet4.

[1] The values are chosen according to recommendations by the algorithms' authors.

Algorithm 3.7 Leaders' Archive Update

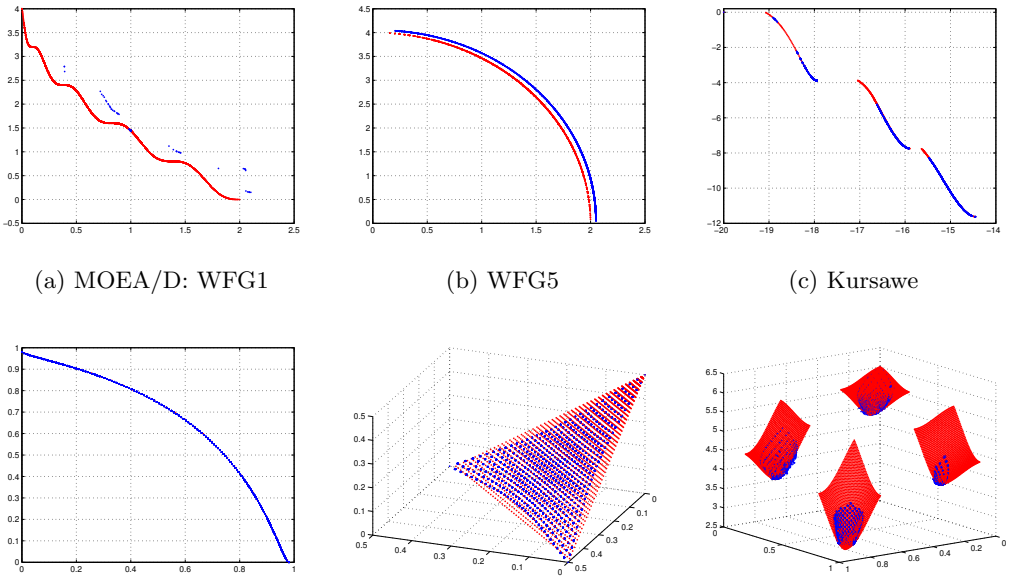
```
1: for each $\overrightarrow{S}$ in the swarm do
2:   for $i = 1$ to $SizeOfArchive$ do
3:     if breachConst($\overrightarrow{S}$) & breachConst($\overrightarrow{p_i}$) then
4:       if constraints($\overrightarrow{S}$) > constraints($\overrightarrow{p_i}$) then
5:         remove $\overrightarrow{p_i}$
6:       else
7:         if constraints($\overrightarrow{S}$) < constraints($\overrightarrow{p_i}$) then
8:           break
9:         else
10:          if constraints($\overrightarrow{S}$) > constraints($\overrightarrow{p_i}$) then
11:            remove $\overrightarrow{p_i}$
12:          else
13:            break
14:          end if
15:        end if
16:      end if
17:    else
18:      if !breachConst($\overrightarrow{S}$) & breachConst($\overrightarrow{p_i}$) then
19:        remove $\overrightarrow{p_i}$
20:      else
21:        break
22:      end if
23:    end if
24:  end for
25:  if valid($\overrightarrow{S}$) then
26:    add $\overrightarrow{S}$ to archive (Algorithm 3.4)
27:  end if
28: end for
```



(a) D^2MOPSO: WFG1 (b) WFG5 (c) Kursawe

(d) Fonseca2 (e) DTLZ1 (f) DTLZ7

Figure 3.17: Plot of the non-dominated solutions with the lowest IGD values in 30 runs of D^2MOPSO.



(a) MOEA/D: WFG1 (b) WFG5 (c) Kursawe

(d) Fonseca2 (e) DTLZ1 (f) DTLZ7

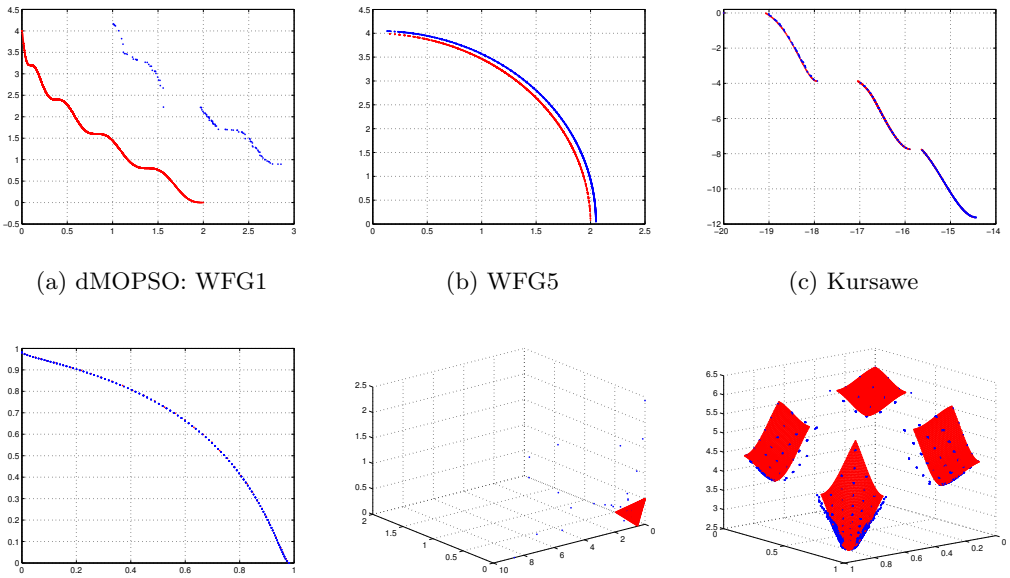
Figure 3.18: Plot of the non-dominated solutions with the lowest IGD values in 30 runs of $MOEA/D$.

Table 3.15 summarizes the main features of the performance measures used in this section. In order to calculate accurate measures and produce informative plots the objective values are normalized by the true PF, i.e. the minimum and maximum of each objective value of the true PF are used to normalise the objective values of the approximated PF.

3.4.7 Results and Discussion

3.4.8 Numeric Comparison

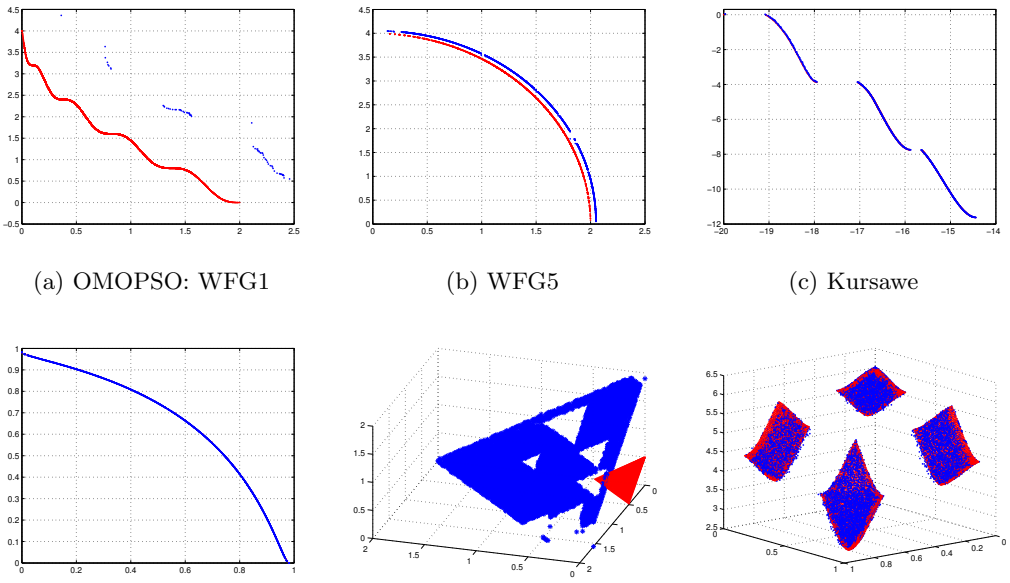
Tables 3.6, 3.7, and 3.8 contain the results of applying I_{hv}, I_{IGD} and I_ϵ respectively to the bi-objective problems, whereas Tables 3.9, 3.10, and 3.11 and Tables 3.12, 3.13, and 3.14 show the results for the three objective and constrained problems respectively. Each raw, of these tables, presents results from one problem solved by the four methods discussed before (D^2MOPSO, MOEA/D, dMOPSO, and OMOPSO) except for Tables 3.12, 3.13, and 3.14 which include results from D^2MOPSO, MOEA/D and OMOPSO



(a) dMOPSO: WFG1 (b) WFG5 (c) Kursawe

(d) Fonseca2 (e) DTLZ1 (f) DTLZ7

Figure 3.19: Plot of the non-dominated solutions with the lowest IGD values in 30 runs of $dMOPSO$.



(a) OMOPSO: WFG1 (b) WFG5 (c) Kursawe

(d) Fonseca2 (e) DTLZ1 (f) DTLZ7

Figure 3.20: Plot of the non-dominated solutions with the lowest IGD values in 30 runs of $OMOPSO$.

only.[1] The results of each problem contain three pieces of information: Med., the median value of the indicator over 30 runs; Irq., the inter quartile ranges of the indicator value over 30 runs; p., the p-value of a Wilcoxon signed-rank test applied to 30 runs of D^2MOPSO and the corresponding algorithm. A non-parametric statistical test is applied as the values are not guaranteed to follow the Gaussian normal distribution (Shapiro-Wilk normality test shows that some values do follow a Gaussian distribution but others do not).

3.4.9 Visual Comparison

To visually demonstrate the performance of the different algorithms seven problems were selected: Four bi-objective (Schaffer, Fonesca2, WFG1, and WFG5); two three-objective (DTLZ1, and DTLZ7); and a constrained problem (Viennet4). These problems are selected to demonstrate the output of D^2MOPSO in both cases where it outperforms and under performs (although slightly) the other methods. The approximated Pareto fronts found by D^2MOPSO (PF_{approx} in blue with PF_{true} in red) are plotted in Fig. 3.17. The results from MOEA/D, dMOPSO, and OMOPSO experiments are illustrated in Fig. 3.18, Fig. 3.19 and Fig. 3.20 respectively.

Although different methods might perform similarly in terms of finding the approximated Pareto front, the number of iterations each algorithm requires to reach this PF may vary. To visually check the convergence of the different methods when solving various problems, the convergence of the four algorithms on the previously selected subgroup of problems is presented. Fig. 3.21 shows the change of IGD per iteration for each method on the seven selected problems. Fig. 3.22 depicts similar plots for the change in the hyper-volume indicator, whereas Fig. 3.23 plots the changes of IGD and hyper-volume for Viennet4.

The Kruskal-Wallis test, which is a nonparametric version of the classical one-way ANOVA and an extension of the Wilcoxon rank sum test to more than two groups, is applied to the unconstrained problems and yielding a value of $p = 0.0092 < 0.05$ (among the four methods), and $p = 0.0066 < 0.05$ when applied on all the problems [2].

[1] dMOPSO has not been applied to the constrained problems because it is specially designed for non-constrained continuous problems, as stated by the authors, so the comparison would not be fair.

[2] dMOPSO is excluded as it does not solve constrained problems

There are some anomalies in the presented tables that should be noted. In Table 3.10, the values of hyper-volume for D^2MOPSO, dMOPSO, and OMOPSO applied to problem $DTLZ3$ are all zero. This is due to the failure of the algorithms to produce a reasonable approximation of the PF. This results in an invalid rank sum test, which is indicated as $--$ in the table. In Table 3.13, MOEA/D has not succeeded to approximate a reasonable PF for $Osyczka2$ resulting in a zero hyper-volume. Finally, Tanaka has a hyper-volume of 1 for MOEA/D (Table 3.13) and a negative ϵ value (Table 3.14), which is impossible because it means the approximated PF dominates the true PF. This can be explained by the fact that MOEA/D could not find any solution that satisfies the problem constraints as it converges to an infeasible solution. For $DTLZ3$, the only method able to approximate the PF is MOEA/D.

3.4.10 Analysis of Computational Complexity

D^2MOPSO combines the advantages of both decomposition (used by MOEA/D) and dominance (adopted in OMOPSO). By doing so, it capitalizes on the benefits of both techniques. In order for D^2MOPSO to be a viable alternative for the state-of-the-art methods, it should have a similar (or better) computational complexity. In this section we compare the computational complexity of D^2MOPSO to that of MOEA/D, MOPSO/D, SDMOPSO, dMOPSO, and OMOPSO.

MOEA/D updates its population using a set of T neighbors. The newly produced solutions replace one or more individuals in the neighborhood based on the aggregation values. Therefore, for a population of size N the complexity is of the order $O(NT) \sim O(N)$. When MOEA/D uses an archive of size $K \geq N$, then the complexity becomes $O(KN + NT) \sim O(KN)$ as each individual will be compared to all the particles in the archive. Similarly, MOPSO/D and SDMOPSO have the complexity of $O(N^2 + NT) \sim O(N^2)$ as $K = N$. The global best set, of size N, in dMOPSO is updated at each iteration using a newly formed set of size $2N$ (as it results from the merge of the global best set with the swarm); hence the computational complexity is $O(2N^2) \sim O(N^2)$ as the aggregation value for each individual must be evaluated against the possible $N\lambda$ vectors. OMOPSO uses the leaders' archive of size N, therefore it requires an algorithm of complexity $O(N^2)$ to be updated. In addition, it uses an $\epsilon-$dominance archive with a size depending on ϵ and the range of objectives. However, assumption

Table 3.5: A comparison of computational complexity

	MOEA/D	MOPSO/D	SDMOPSO	dMOPSO	OMOPSO	D^2MOPSO
No Arch.	O(N)	-	-	O(N^2)	-	O(N)
Arch.	O(KN)	O(N^2)	O(N^2)	O(KN)	O(KN)	O(KN)

can be made that it is of size $K > N$ making the total computational complexity of OMOPSO $O(KN + N^2) \sim O(KN)$.

D^2MOPSO uses the leaders' archive (of size $L \leq N$) which is updated on each iteration. In order to select the global leader for each particle, all solutions in the leaders' archive are checked for the best aggregation value. The complexity would then be $O(2LN) \sim O(N)$ iff $L \ll N$. When an external archive (of size $K > N$) is used, the complexity becomes $O(KN+2LN) \sim O(KN)$. The external archive is only used when the method is expected to generate a very large number of non-dominated solutions, as shown in Table 3.5.

We can conclude from this analysis that D^2MOPSO has similar computational complexity to the other state-of-the-art algorithms.

3.4.11 Discussion

D^2MOPSO is presented as a novel multi-objective particle swarm optimisation algorithm that combines decomposition and dominance. The decomposition simplifies the optimisation problem by transforming it to a set of single-objective problems, whereas dominance facilitates the leaders' archiving process. Decomposition is used to update the personal information and to select the global leaders.

A new archiving technique is also presented, which considers the diversity in both the search and objective spaces. By doing so, the archive helps covering promising regions in both spaces. Crowding distance is used to implement the new archive in this thesis, but it can be substituted by any of the other techniques explained in Section 3.4.1.

An extensive experimentation is carried out covering the different types of PFs. To quantify the performance of D^2MOPSO, three distinct quality measures are used to compare its performance with three state-of-the-art algorithms: a) MOEA/D, a genetic algorithm based decomposition algorithm. b) dMOPSO, a decomposition-based

MOPSO. c) OMOPSO, a dominance-based MOPSO. The results are supported by several statistical tests that count for direct and multiple comparison conditions. For unconstrained bi-dimensional problems, D^2MOPSO outperforms the other methods (except for $WFG8$) with respect to I_{IGD}, I_{hv} and I_ϵ. For unconstrained three-dimensional problems, D^2MOPSO performs better in terms of I_{IGD}, I_{hv}, and I_ϵ in all problems except for $DTLZ1$, and $DTLZ3$. For constrained problems, D^2MOPSO outperforms the other algorithms in terms of I_{IGD}. According to I_{hv}, D^2MOPSO under-performs in only one problem: $ConstrEx$. With respect to I_ϵ, D^2MOPSO yields similar results - outperforming in the case of $Osyczka2$, and $Srinivas$.

In general, D^2MOPSO is demonstrated to be highly competitive to the other algorithms with the advantage of no requirement of parameter tuning and a comparable computational overhead (Section 3.4.10).

3.5 Conclusion

The chapter started by presenting a new algorithm that integrates decomposition into MOPSO, SDMOPSO. Despite its demonstrated relative success, SDMOPSO is prune to falling into local optima because it relies fully on the local neighbourhood information and does not exploit the global information in the swarm.

A novel framework (CLS) for archiving was presented that maps indirectly the search and objective space via clustering both space and mapping the particles among the two sets of clusters. This has shown to be very beneficial not only for MOPSO but for the general evolutionary multi-objective optimizers. However, it suffers from the high complexity of clustering from one side and the difficulty of clustering high-dimensional data on the other side.

D^2MOPSO builds on the experience of the previous two approaches and addresses their limitations. The proposed algorithm integrates decomposition into MOPSO and incorporates both local and global information. An archiving technique is used which is simple and effective based on a mapping between search and objective spaces using a crowding distance measure: ϵ-dominance.

All the techniques are extensively studied and analysed on standard problems. A detailed analysis of the performance of SDMOPSO and D^2MOPSO is also presented. However, the real test of any optimisation method is on real-life applications. Next

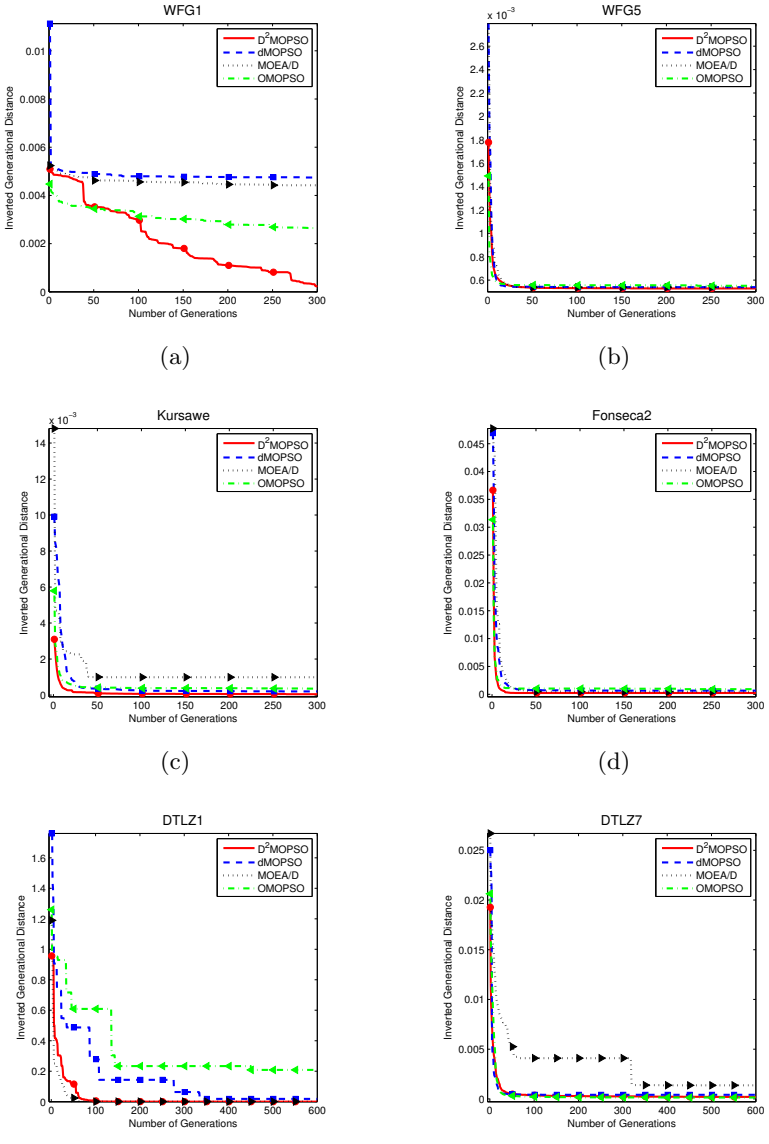


Figure 3.21: The evaluation of IGD for the four algorithms.

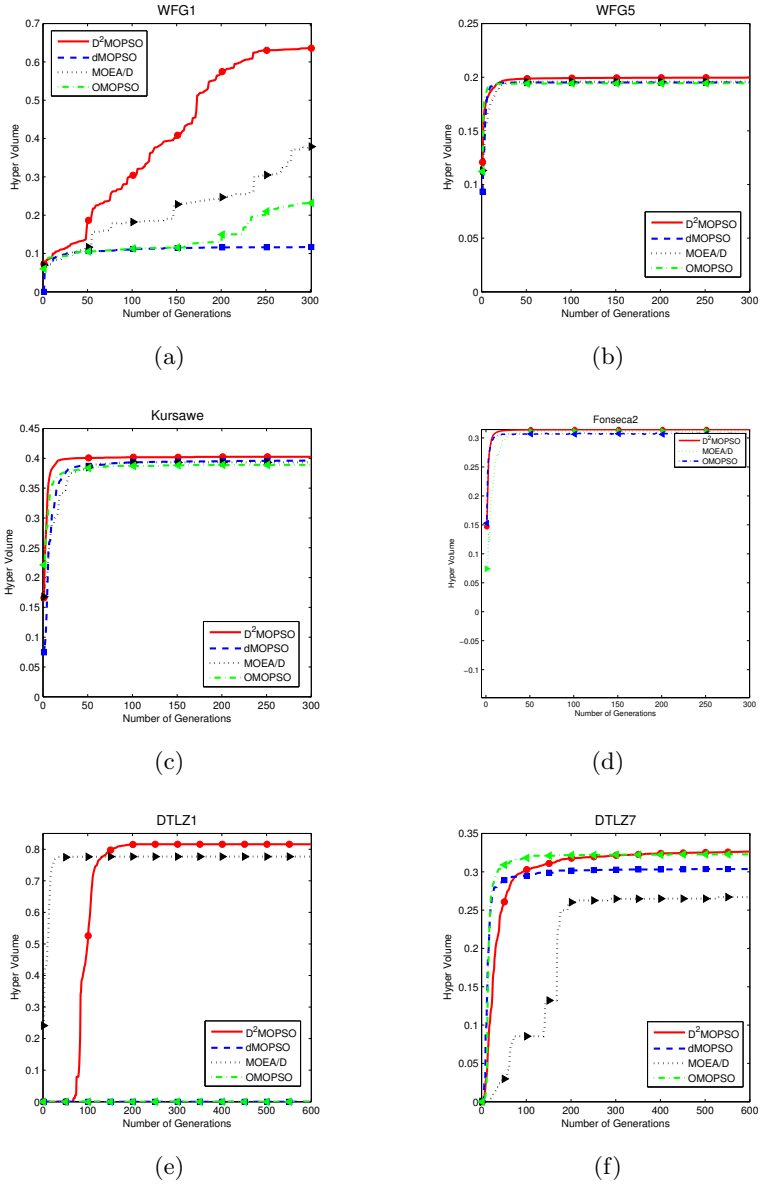


Figure 3.22: The evaluation of Hyper Volume for the four algorithms.

Table 3.6: Results of I_{IGD} on unconstrained bi-objective test problems

Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
Fonseca2	Med.	2.41e-004	5.03e-004	6.49e-004	1.20e-003
	Irq.	1.38e-005	1.89e-006	5.55e-006	1.28e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
Kursawe	Med.	6.74e-005	1.30e-003	2.02e-004	3.78e-004
	Irq.	1.76e-005	1.51e-005	8.75e-006	1.98e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
Schaffer	Med.	9.88e-005	1.27e-002	6.26e-003	1.81e-004
	Irq.	1.89e-005	6.73e-003	2.16e-006	1.22e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG1	Med.	1.45e-004	1.86e-003	4.73e-003	3.77e-003
	Irq.	2.96e-004	3.65e-004	4.75e-005	8.92e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG2	Med.	1.82e-005	1.16e-003	7.94e-004	1.13e-004
	Irq.	9.42e-006	3.32e-005	9.78e-005	2.42e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG3	Med.	6.84e-004	6.84e-004	1.52e-003	6.84e-004
	Irq.	1.42e-007	2.51e-008	1.11e-006	7.58e-008
	p	–	4.20e-010	3.02e-011	3.02e-011
WFG4	Med.	4.87e-005	1.95e-004	2.85e-004	2.71e-004
	Irq.	1.98e-005	4.55e-005	3.91e-005	6.67e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG5	Med.	5.31e-004	5.39e-004	5.39e-004	5.70e-004
	Irq.	1.48e-006	2.05e-007	2.23e-006	1.16e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG6	Med.	1.53e-005	8.55e-005	1.86e-004	1.98e-004
	Irq.	1.14e-006	6.44e-007	2.32e-005	3.65e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG7	Med.	1.48e-005	9.24e-005	1.79e-004	1.60e-004
	Irq.	1.01e-006	3.30e-007	1.45e-005	1.95e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG8	Med.	1.03e-003	8.70e-004	6.80e-004	1.04e-003
	Irq.	1.37e-004	1.50e-004	1.65e-004	1.23e-005
	p	–	2.88e-006	8.89e-010	3.03e-002
WFG9	Med.	6.26e-005	1.16e-004	1.85e-004	2.22e-004
	Irq.	9.63e-006	2.52e-005	8.82e-006	3.04e-005
	p	–	3.02e-011	3.02e-011	3.02e-011

Table 3.7: Results of I_{hv} on unconstrained bi-objective test problems

Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
Fonseca2	Med.	3.14e-001	3.12e-001	3.09e-001	3.07e-001
	Irq.	1.93e-005	4.01e-007	1.08e-004	5.22e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
Kursawe	Med.	4.04e-001	3.92e-001	3.96e-001	3.90e-001
	Irq.	4.91e-004	3.44e-004	7.25e-004	9.11e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
Schaffer	Med.	8.33e-001	7.09e-001	8.22e-001	8.32e-001
	Irq.	2.94e-005	9.82e-002	6.75e-006	7.99e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG1	Med.	6.31e-001	3.81e-001	1.19e-001	1.57e-001
	Irq.	2.71e-002	5.41e-002	2.56e-003	5.57e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG2	Med.	5.65e-001	5.53e-001	5.55e-001	5.61e-001
	Irq.	1.64e-004	2.32e-003	1.25e-003	8.64e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG3	Med.	4.44e-001	4.42e-001	2.77e-001	4.42e-001
	Irq.	5.39e-005	6.79e-006	2.32e-004	1.65e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG4	Med.	2.20e-001	2.10e-001	2.01e-001	2.07e-001
	Irq.	1.20e-003	3.41e-003	2.38e-003	1.03e-003
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG5	Med.	1.99e-001	1.96e-001	1.95e-001	1.93e-001
	Irq.	4.50e-005	1.80e-005	8.42e-005	6.89e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG6	Med.	2.13e-001	2.11e-001	2.01e-001	2.07e-001
	Irq.	8.21e-005	1.44e-005	1.52e-003	6.16e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG7	Med.	2.14e-001	2.11e-001	2.01e-001	2.07e-001
	Irq.	6.64e-005	5.73e-006	1.47e-003	7.03e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG8	Med.	1.48e-001	1.52e-001	1.65e-001	1.46e-001
	Irq.	2.67e-003	1.44e-003	7.46e-003	1.07e-003
	p	–	6.53e-008	2.23e-009	3.52e-007
WFG9	Med.	2.41e-001	2.39e-001	2.31e-001	2.32e-001
	Irq.	9.93e-004	1.99e-003	6.12e-004	9.57e-004
	p	–	4.20e-010	3.02e-011	3.02e-011

Table 3.8: Results of I_ϵ on unconstrained bi-objective test problems

Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
Fonseca2	Med.	1.88e-003	4.12e-003	6.41e-003	1.05e-002
	Irq.	1.96e-003	1.47e-005	2.77e-004	3.20e-003
	p	–	9.51e-006	8.48e-009	1.46e-010
Kursawe	Med.	6.42e-002	3.58e-001	1.18e-001	1.50e-001
	Irq.	2.40e-002	1.58e-002	1.42e-002	1.35e-002
	p	–	3.02e-011	7.39e-011	3.02e-011
Schaffer	Med.	4.69e-003	7.29e-001	9.03e-002	1.37e-002
	Irq.	1.37e-003	3.43e-001	5.50e-005	2.33e-003
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG1	Med.	8.31e-002	5.85e-001	1.13e+000	1.12e+000
	Irq.	1.22e-001	1.14e-001	4.12e-002	1.16e-001
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG2	Med.	3.71e-003	1.14e-001	9.39e-002	2.80e-002
	Irq.	3.51e-003	6.12e-001	6.68e-003	6.53e-003
	p	–	3.02e-011	3.02e-011	6.01e-008
WFG3	Med.	2.00e+000	2.00e+000	3.00e+000	2.00e+000
	Irq.	4.84e-004	7.17e-005	1.89e-004	2.14e-004
	p	–	1.07e-009	3.02e-011	1.34e-005
WFG4	Med.	1.45e-002	5.75e-002	6.73e-002	5.67e-002
	Irq.	7.28e-003	2.14e-002	1.05e-002	1.09e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG5	Med.	5.20e-002	6.96e-002	7.20e-002	9.00e-002
	Irq.	2.53e-004	3.58e-004	4.95e-004	5.80e-003
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG6	Med.	4.05e-003	1.79e-002	5.41e-002	4.22e-002
	Irq.	8.72e-004	1.27e-003	1.14e-002	1.13e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG7	Med.	3.63e-003	2.09e-002	4.31e-002	4.57e-002
	Irq.	3.62e-004	1.08e-003	3.68e-003	1.07e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
WFG8	Med.	5.08e-001	3.93e-001	5.06e-001	5.31e-001
	Irq.	1.11e-002	2.01e-001	8.85e-002	1.71e-002
	p	–	4.73e-001	7.62e-001	3.09e-006
WFG9	Med.	1.28e-002	3.50e-002	3.93e-002	4.99e-002
	Irq.	1.40e-003	1.22e-002	2.52e-003	8.76e-003
	p	–	3.02e-011	3.02e-011	3.02e-011

Table 3.9: Results of I_{IGD} on unconstrained three-objective test problems

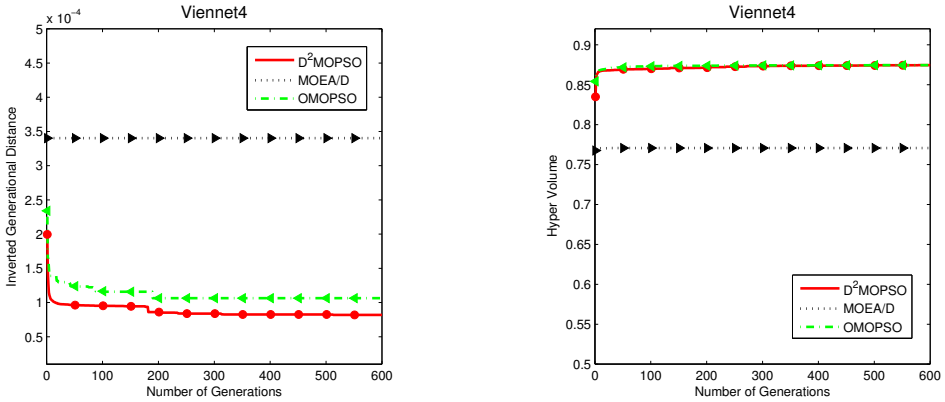
Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
DTLZ1	Med.	4.72e-002	4.75e-004	4.72e-002	1.88e-001
	Irq.	6.65e-002	1.20e-006	6.65e-002	1.34e-001
	p	–	3.02e-011	1.00e+000	2.03e-007
DTLZ2	Med.	4.19e-005	1.09e-004	1.18e-004	9.25e-005
	Irq.	3.61e-007	2.94e-008	8.17e-007	7.23e-006
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ3	Med.	3.54e-001	3.87e-004	6.14e-001	1.76e+000
	Irq.	3.78e-001	7.17e-007	5.07e-001	8.46e-001
	p	–	3.02e-011	4.43e-003	9.92e-011
DTLZ4	Med.	2.09e-004	3.88e-004	4.39e-004	2.71e-004
	Irq.	1.82e-006	1.03e-006	5.32e-006	5.52e-006
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ5	Med.	1.08e-005	1.80e-004	1.06e-004	1.68e-004
	Irq.	9.91e-006	9.63e-008	6.55e-006	5.49e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ6	Med.	2.90e-005	1.81e-004	1.80e-004	1.72e-004
	Irq.	1.20e-005	9.01e-009	9.28e-008	3.83e-005
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ7	Med.	1.95e-004	1.37e-003	4.11e-004	1.47e-004
	Irq.	1.75e-005	1.52e-005	6.35e-007	3.46e-006
	p	–	3.02e-011	3.02e-011	3.02e-011
Viennet2	Med.	6.91e-005	2.23e-003	1.56e-003	1.08e-003
	Irq.	1.33e-005	1.24e-006	7.02e-006	4.29e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
Viennet3	Med.	2.02e-003	4.98e-003	4.12e-003	6.85e-004
	Irq.	2.26e-003	1.39e-006	2.86e-006	5.75e-004
	p	–	3.02e-011	3.02e-011	6.53e-007

Table 3.10: Results of I_{hv} on unconstrained three-objective test problems

Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
DTLZ1	Med.	8.16e-001	7.76e-001	0.00e+000	0.00e+000
	Irq.	9.96e-003	3.10e-004	0.00e+000	0.00e+000
	p	–	7.88e-012	1.00e+000	5.58e-003
DTLZ2	Med.	4.63e-001	4.53e-001	4.42e-001	4.61e-001
	Irq.	1.70e-004	1.09e-005	7.52e-004	2.46e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ3	Med.	0.00e+000	4.49e-001	0.00e+000	0.00e+000
	Irq.	0.00e+000	4.06e-005	0.00e+000	0.00e+000
	p	–	1.21e-012	–	–
DTLZ4	Med.	4.61e-001	4.49e-001	4.38e-001	4.59e-001
	Irq.	1.57e-004	3.03e-005	8.09e-004	3.99e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ5	Med.	9.56e-002	8.78e-002	9.11e-002	9.13e-002
	Irq.	8.36e-005	6.03e-006	3.08e-004	7.40e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ6	Med.	9.46e-002	8.78e-002	8.78e-002	9.08e-002
	Irq.	1.91e-004	1.32e-007	7.17e-006	5.46e-004
	p	–	3.02e-011	3.02e-011	3.02e-011
DTLZ7	Med.	3.27e-001	2.64e-001	3.04e-001	3.21e-001
	Irq.	6.88e-004	1.49e-003	4.13e-004	2.14e-003
	p	–	3.02e-011	3.02e-011	3.02e-011
Viennet2	Med.	9.31e-001	8.45e-001	9.03e-001	8.81e-001
	Irq.	1.24e-004	1.38e-004	3.47e-004	1.41e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
Viennet3	Med.	8.40e-001	8.18e-001	8.31e-001	8.09e-001
	Irq.	2.95e-004	4.02e-005	4.13e-005	9.01e-003
	p	–	3.02e-011	3.02e-011	3.02e-011

Table 3.11: Results of I_ϵ on unconstrained three-objective test problems

Problem		D^2MOPSO	**MOEA/D**	**dMOPSO**	**OMOPSO**
DTLZ1	Med.	1.18e+000	3.28e-002	1.18e+000	3.81e+000
	Irq.	1.27e+000	4.14e-004	1.27e+000	2.09e+000
	p	–	3.02e-011	1.00e+000	3.52e-007
DTLZ2	Med.	1.85e-002	3.31e-002	3.75e-002	1.96e-002
	Irq.	1.99e-003	8.01e-004	1.65e-003	2.52e-003
	p	–	3.02e-011	3.02e-011	1.68e-004
DTLZ3	Med.	1.48e+001	4.07e-002	2.84e+001	8.91e+001
	Irq.	1.45e+001	1.55e-003	2.90e+001	4.24e+001
	p	–	3.02e-011	3.77e-004	3.02e-011
DTLZ4	Med.	2.73e-002	4.10e-002	4.48e-002	2.43e-002
	Irq.	2.27e-003	2.06e-003	1.40e-003	1.89e-003
	p	–	3.02e-011	3.02e-011	3.83e-006
DTLZ5	Med.	2.85e-003	1.56e-002	1.25e-002	1.08e-002
	Irq.	3.91e-003	2.12e-005	1.11e-003	3.09e-003
	p	–	3.02e-011	2.67e-009	1.56e-008
DTLZ6	Med.	7.54e-003	1.56e-002	1.56e-002	1.14e-002
	Irq.	9.46e-003	5.03e-009	2.60e-005	2.56e-003
	p	–	1.11e-006	1.11e-006	1.63e-002
DTLZ7	Med.	5.20e-002	1.46e-001	7.31e-002	4.02e-002
	Irq.	1.00e-002	3.66e-003	1.18e-003	1.33e-002
	p	–	3.02e-011	5.57e-010	7.70e-004
Viennet2	Med.	5.26e-003	6.03e-002	3.52e-002	4.83e-002
	Irq.	7.28e-004	1.62e-004	4.58e-004	1.99e-002
	p	–	3.02e-011	3.02e-011	3.02e-011
Viennet3	Med.	2.66e-002	1.06e-001	5.22e-002	1.39e-001
	Irq.	7.39e-003	1.68e-004	1.40e-004	4.38e-002
	p	–	3.02e-011	3.02e-011	3.02e-011



(a) IGD (b) Hyper Volume

Figure 3.23: The evaluation of the four algorithms for Viennet4.

Table 3.12: Results of I_{IGD} on constrained test problems

Problem		D^2MOPSO	**MOEA/D**	**OMOPSO**
ConstrEx	Med.	2.42e-003	1.02e-002	2.92e-004
	Irq.	1.04e-003	1.76e-007	2.40e-005
	p	–	3.02e-011	3.02e-011
Golinski	Med.	9.65e-003	2.65e-002	9.65e-003
	Irq.	9.90e-003	3.74e-008	3.61e-003
	p	–	3.02e-011	9.82e-001
Osyczka2	Med.	3.98e-003	2.57e-001	4.49e-003
	Irq.	7.56e-004	2.57e-003	5.63e-003
	p	–	3.02e-011	7.48e-002
Srinivas	Med.	1.05e-005	1.42e-004	1.11e-005
	Irq.	3.47e-006	1.14e-007	5.52e-006
	p	–	3.02e-011	3.04e-001
Tanaka	Med.	3.36e-004	4.71e-002	3.95e-004
	Irq.	8.25e-005	0.00e+000	5.27e-005
	p	–	1.21e-012	6.55e-004
Viennet4	Med.	1.74e-004	8.72e-004	1.44e-004
	Irq.	3.07e-005	4.44e-006	5.16e-005
	p	–	3.02e-011	1.76e-003

Table 3.13: Results of I_{hv} on constrained test problems

Problem		D^2MOPSO	**MOEA/D**	**OMOPSO**
ConstrEx	Med.	7.12e-001	9.02e-001	7.74e-001
	Irq.	2.49e-002	2.82e-005	5.02e-004
	p	–	3.02e-011	3.02e-011
Golinski	Med.	9.68e-001	9.96e-001	9.62e-001
	Irq.	1.45e-003	0.00e+000	1.72e-003
	p	–	5.22e-012	3.02e-011
Osyczka2	Med.	6.34e-001	0.00e+000	7.09e-001
	Irq.	3.78e-002	0.00e+000	9.66e-003
	p	–	1.21e-012	3.02e-011
Srinivas	Med.	5.45e-001	5.36e-001	5.45e-001
	Irq.	1.66e-004	1.64e-005	7.42e-005
	p	–	3.02e-011	2.23e-001
Tanaka	Med.	3.04e-001	1.00e+000	3.00e-001
	Irq.	4.21e-004	0.00e+000	2.45e-003
	p	–	1.21e-012	3.02e-011
Viennet4	Med.	8.70e-001	7.64e-001	8.74e-001
	Irq.	5.45e-004	6.90e-004	5.09e-004
	p	–	3.02e-011	2.99e-011

Table 3.14: Results of I_ϵ on Constrained test problems

Problem		D^2MOPSO	**MOEA/D**	**OMOPSO**
ConstrEx	Med.	1.14e-001	2.20e-002	1.51e-002
	Irq.	5.32e-002	2.09e-005	3.05e-003
	p	–	3.02e-011	3.02e-011
Golinski	Med.	7.24e+000	2.58e+000	3.78e+001
	Irq.	3.09e+000	0.00e+000	1.05e+001
	p	–	5.22e-012	3.02e-011
Osyczka2	Med.	1.56e+001	9.69e+001	2.58e+001
	Irq.	3.93e+000	7.02e-001	1.41e+001
	p	–	3.02e-011	3.83e-005
Srinivas	Med.	8.74e-001	2.51e+000	1.28e+000
	Irq.	8.73e-001	3.25e-002	4.30e-001
	p	–	4.98e-011	5.83e-003
Tanaka	Med.	1.53e-002	-4.29e-002	1.35e-002
	Irq.	4.59e-003	0.00e+000	2.37e-003
	p	–	1.21e-012	1.33e-002
Viennet4	Med.	1.31e-001	3.49e-001	9.78e-002
	Irq.	2.16e-002	1.21e-003	1.63e-002
	p	–	3.02e-011	3.79e-010

Table 3.15: Main Features of the Performance Measures

	I_{IGD}	I_{hv}	I_ϵ
Goal	Hybrid	Hybrid	Diversity
Monotone	No	Strict	Mon
Parameter	Ref set	Ref point	Ref set
Min/Max	Min	Max	min

chapter shows the performance and applicability of the proposed method on two real-life applications: Channel selection for Brain-Computer Interfaces, and regulation of Cancer Chemotherapy doses. The two problems are support by novel representation and detailed analysis of the results compared to the state of the art in either problem.

Chapter 4

Applications

"When you know better you do better. "

Maya Angelou

4.1 Introduction

Many real-life optimisation problems have several conflicting objectives that necessitate finding trade-off solutions in large search spaces with disjoint regions of feasibility. It has been discovered that problems of this nature can be effectively handled by computational intelligence techniques Liang *et al.* (2008), including Genetic Algorithms Petrovski (1999); Petrovski & McCall (1999), and Particle Swarm Optimisation (PSO) A. Petrovski & McCall (2004).

The PSO algorithm, in particular, has proven to be very efficient and capable of providing competitive solutions in many application domains Wang *et al.* (2004)Jaishia & Ren (2007)Hasan *et al.* (2010), including medicine Li *et al.* (205). Multi-objective implementations of PSO have also been developed and proved their ability to provide viable solutions Reyes-Sierra & Coello (2006)Baltar & Fontane (2006). In A. Petrovski & McCall (2004) and Hassan *et al.* (2005), the authors have observed that although PSO and GA on average demonstrate the same effectiveness in terms of solution quality, PSO is more computationally efficient and uses fewer fitness function evaluations as indicated by two standard statistical tests. Moreover, in contrast to GA, PSO requires less subjective tuning of parameters, which simplifies its implementation Hassan *et al.* (2005).

The aim of this chapter is to demonstrate the applicability of the developed algorithms in real-life problems and to show its potential impact on different fields of science and engineering. Two problems are used to test $SDMOPSO$ and D^2MOPSO: channel selection for Brain-Computer Interface, and Dose optimization for cancer chemotherapy.

Following the literature, the channel selection problem is first represented using binary variables which requires $SDMOPSO$ to be modified to accommodated these variable. A new representation is then introduced which uses continuous variables. The new representation is tested using D^2MOPSO.

The other application is the finding of effective Cancer chemotherapeutic treatments that balances complicated aspects of the therapy in order to give the patient good quality of life during the treatment while effectively targeting the tumour.

4.2 Channel Selection for Brain-Computer Interfaces

Brain-Computer Interface (BCI) research aims at developing a communication mean between the human brain and the machine to provide an alternative control pathway for the user. BCI applications vary from gaming to helping severely paralyzed patients Birbaumer *et al.* (2003); Tsui *et al.* (2009); Wolpaw *et al.* (2002). The user of a BCI system can perform several well-studied mental tasks to communicate and control Neuper & Pfurtsheller (1999); Pfurtscheller & da Silva (1999); Pfurtscheller *et al.* (2005). The machine must be able to recognize these tasks from brain signals accordingly within a suitable time window for control. Motor imagery tasks are commonly used in BCI environment due to their good separability and the understanding of their neurological mechanisms Hasan & Gan (2011).

Non-invasive BCI uses Electroencephalography (EEG) signals associated with predefined mental tasks. The number of channels used by an EEG system can vary according to experiment paradigm and hardware design. It usually ranges between 10 and 256 channels. For real-life BCI application it is important to select a smaller set of channels with as little sacrifice as possible in classification accuracy.

In order to avoid a large number of channels one can choose several electrode positions that are known from neurophysiological studies. Although this approach can be very useful, it ignores the fact that different subjects respond differently and the optimal positioning of the electrodes may vary. The other way to circumvent this problem is to use a large number of channels and use a method to reduce the dimensionality of the input features or to select the best set of channels for each subject.

Common Spatial Patterns (CSP) Dornhege *et al.* (2006) is a well-known spatial filter that is widely used in BCI. CSP is useful for channel selection as it can be used to filter out the channels that provide less discriminate data. CSP requires the data from all the channels to be available online before the dimensionality is reduced. CSP depends on the estimation of the covariance matrices of multiple channel EEG data, which is usually very sensitive to noise.

The problem of channel selection is usually looked at as a search problem. The idea is to search the space of possible combinations of channels in order to find the optimal combination that produces the best classification accuracy. In Gan (2006), the author argues that feature selection is advantageous over dimensionality reduction in terms of

interpretability. Feature selection (and similarly channel selection) using several search methods has been used frequently in the literature.

In Jin *et al.* (2008) Digital Particle Swarm Optimization (DPSO) was used, where each particle contained a number of binary variables, which is equal to the number of channels, and cross validation results were used as the fitness function. In Jun & Meichun (2008) a mixture of CSP and PSO based method was used for channel selection. In Dyson *et al.* (2009); Gan *et al.* (2011) Sequential Floating Forward Search (SFFS) based methods were employed for channel/feature selection.

Most search-based solutions presented in the literature are single-objective methods. The classification accuracy is usually chosen as the only search criterion. The shortcoming of this approach is that the optimization process does not take into consideration the trade-off between the number of channels selected and the desirable classification accuracy. In theory, more channels would provide extra information that can help enhance the classification accuracy. In practice this might not be very accurate. It could be even desirable to sacrifice the accuracy in order to have fewer channels and hence a BCI system which can react to the user input within a more reasonable time window. Hasan *et al.* (2010), was first to study a multi-objective approach to this problem using MOEA/D and OMOPSO.

4.2.1 Modeling Multi-Objective Channel Selection Problem

The modeling of the channel selection problem consists of defining the objective functions to be optimized and the representation of the problem in order for the optimization method to be able to solve it. Whilst the definition of the objective function is independent of the selected optimization method, the problem representation is highly correlated to the optimization method.

The first objective function is the classification error rate defined as $E = 1 - CV$ where CV is the cross-validation result. The second objective function is the number of selected channels N. The optimal solution(s) would have minimum number of channels with lowest error rate and hence minimize E and N.

Each channel is represented by a binary variable $c \in \{0, 1\}$, where $c = 1$ if the channel was selected for classification and $c = 0$ otherwise. This is a very simplistic modelling of the problem but the commonly used one in the literature for the single objective approach. Section 4.2.4 introduces a novel alternative presentation.

The basic PSO was originally designed for real-valued problems (continuous or discrete) which necessitates customizing SDMOPSO to handle binary variables. SDMOPSO adopts the Modified Discrete Binary PSO algorithm (MBPSO) for this purpose (Shen *et al.* (2004)) . The method is applied on the channel selection problem in Brain-Computer Interfaces (Hasan *et al.* (2010)).

4.2.2 The Binary SDMOPSO

Originally, PSO was designed to handle continuous and discrete problems with real-value decision variables. The particles change their positions by updating their decision variables using Eq.2.2 and Eq.2.3. For binary problems, Kennedy & Eberhart (1997b) customized the definition of PSO (KBPSO) in order to handle binary problems where the decision variables are bits of 0s or 1s. Each bit is only allowed to change its value to 1 or 0 or keep its value without change.

The velocity update does not change, but the resulted real value must be normalised between (0 and 1) using sigmoid limiting transformation Eq.4.1.

$$S(\overrightarrow{v_{ij}}) = \frac{1}{1 + e^{-\overrightarrow{v_{ij}}}} \tag{4.1}$$

where i is the index of the particle and j is the decision variable (bit) to be updated. Then the position update uses Eq.4.2.

$$\overrightarrow{x_{ij}} = \begin{cases} 1 & \text{if rand } \leq S(\overrightarrow{v_{ij}}) \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where $rand$ is a real random value in $[0, 1]$.

In this scenario MBPSO (Shen *et al.* (2004)) an improved version of KBPSO is adopted, where the update position is replaced by the following:

$$\overrightarrow{x_{ij}}(\text{new}) = \begin{cases} \overrightarrow{x_{ij}}(\text{old }) & : 0 < S(\overrightarrow{v_{ij}}) \leq \alpha \\ \overrightarrow{pbest_{ij}} & : \alpha < S(\overrightarrow{v_{ij}}) \leq \frac{1}{2}(1+\alpha) \\ \overrightarrow{nbest_{ij}} & : \frac{1}{2}(1+\alpha) < S(\overrightarrow{v_{ij}}) \leq 1 \end{cases} \tag{4.3}$$

where $\overrightarrow{pbest_{ij}}, \overrightarrow{nbest_{ij}}$ are the personal best and neighbouring best of i and α called static probability is a random value in $[0, 1]$. Here, $nbest$ replaces the global best ($gbest$) used in (Shen *et al.* (2004)).

The choice of *nbest* instead of *gbest* is important to accommodate the neighbours topology in SDMOPSO. This binary version of SDMOPSO has the same advantages of SDMOPSO in terms of information exchange and decomposition of solution space with the extra support of binary problems.

4.2.3 Experiments and Results

4.2.3.1 Data Set

In this study the Dataset 1 for BCI Competition IV was used. The challenge is the classification of continuous EEG without trial structure (Blankertz *et al.* (2007)). The dataset is divided into training data and testing data. The calibration data are synchronous trials for 7 subjects (3 of the datasets are synthesized data). The evaluation data are soft-cued trials. For each subject 3 motor imagery tasks (right hand, left hand and foot, where the foot side was chosen by the subject) were recorded but only the most separable 2 tasks were provided. 59 channels were used to record EEG data. The aim was to test the channel selection method, the evaluation data were not used. More technical details and information about data acquisition and recording method can be found in (Blankertz *et al.* (2007)).

4.2.3.2 Feature Extraction and Classification

The original dataset was sampled at 1000Hz. Another downsampled version (at 100Hz) was used here as provided by the authors. Autoregressive features of order 6 were extracted, with a 4 samples shift window, resulting in 25 samples per second. Linear Discriminant Analysis (LDA) was used to classify the extracted features. When the number of features is more than 20, Principle Component Analysis (PCA) was used to reduce the dimensionality of the input.

Binary-SDMOPSO is compared here with results using MOEA/D and OMOPSO. Binary-SDMOPSO uses a swarm of 59 particles. These parameters were set similar to the one used in (Hasan *et al.* (2010)). The inertia weight W is set to a random value in the range [0.1, 0.5], C_1 and C_2 are set to 2, r_1 and r_2 are set to random values in [0,1].

For OMOPSO and MOEA/D the parameters were set as in (Hasan *et al.* (2010)). For OMOPSO the number of particles was set to 59, and the perturbation index (a parameter for the mutation operator used in OMOPSO) is set to 0.5. W is set to a

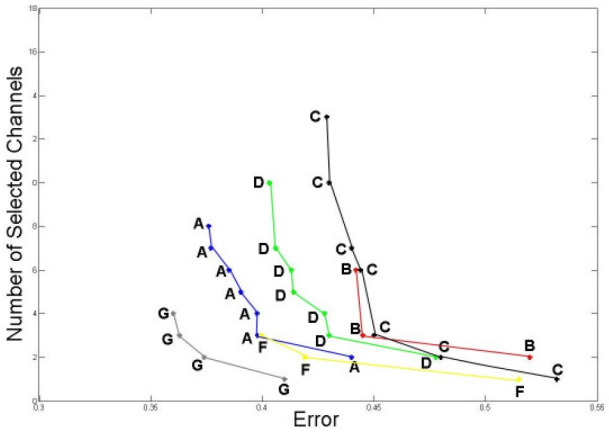


Figure 4.1: Solutions obtained by OMOPSO. The approximated Pareto front related to every subject is marked by the corresponding letter.

random value in the range [0.1, 0.5], C_1 and C_2 are set to random values in the range [1.5, 2.0]. As recommended in (Sierra & Coello Coello (2005)).

MOEA/D uses a single crossover operator to generate the offspring in addition to a mutation operator. The mutation probability equals to 0.05.

In order to save computational time, all the previous methods use a lookup table that contains all the previously tested combination of channels. This guarantees that the same combination of channels will not be tested more than once if two (or more) individuals ended up having the same solution.

Each individual/particle contains 59 binary variables, whose value can be 1 (the channel is selected) or 0 (the channel is not selected). All methods terminate after 200 iterations.

The 7 subjects were named as “a”, “b”, “c”, “d”, “e”, “f”, “g”. Subject “e” was not included in the study as the size of data is different from the other datasets.

Table 4.1 shows the solutions when using OMOPSO, with the highlighted rows being the “best” solution in terms of accuracy and number of selected channels as chosen by the experts in the field. Figure 4.1 shows the solutions provided by OMOPSO for the 6 subjects, which are the resulted Pareto front for each subject. Table 4.2 and Fig. 4.2 present the results when using MOEA/D in the same way as for OMOPSO. In a similar way Table 4.3 and Fig. 4.3 present the results for Binary-SDMOPSO. The solutions presented in the tables and figures are just the non-dominated solutions.

Table 4.1: RESULTS USING OMOPSO

Subject	Number of Selected Channels	Cross Validation Accuracy
a	2	0.56
	3	0.602
	4	0.6024
	5	0.6097
	6	0.615
	7	**0.6231**
	8	0.6239
b	2	0.48
	3	**0.555**
	6	0.558
c	1	0.468
	2	0.52
	3	0.55
	6	0.556
	7	0.56
	10	**0.57**
	13	0.571
d	2	0.5222
	3	0.57
	4	0.572
	5	0.5859
	6	0.5869
	7	**0.594**
	10	0.571
f	1	0.4848
	2	0.581
	3	**0.6007**
g	1	0.59
	2	0.626
	3	0.6371
	4	**0.64**

Table 4.2: RESULTS USING MOEA/D

Subject	Number of Selected Channels	Cross Validation Accuracy
a	1	0.5652
	2	0.6093
	3	**0.6183**
b	2	0.5726
	3	**0.5815**
	4	0.59
c	2	0.5401
	3	**0.5454**
d	1	0.5684
	3	**0.5832**
f	1	0.5544
	2	0.5845
	3	0.6087
	5	**0.6166**
g	2	0.6319
	3	0.6381
	4	**0.6404**

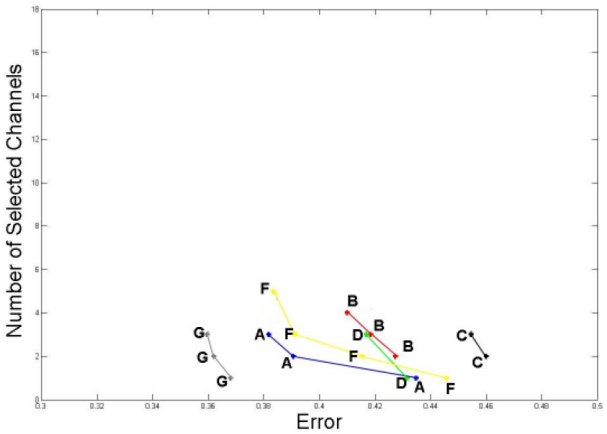


Figure 4.2: Solutions obtained by MOEA/D. The approximated Pareto front related to every subject is marked by the corresponding letter.

Table 4.3: RESULTS USING Binary-SDMOPSO

Subject	Number of Selected Channels	Cross Validation Accuracy
a	1	0.5905
	2	**0.6123**
b	1	0.5782
	2	**0.6167**
c	1	0.564
	2	0.5825
	3	**0.5977**
	7	0.6059
d	1	0.5696
	3	**0.609**
f	**2**	**0.5861**
	6	0.5939
g	1	0.5204
	2	**0.6244**

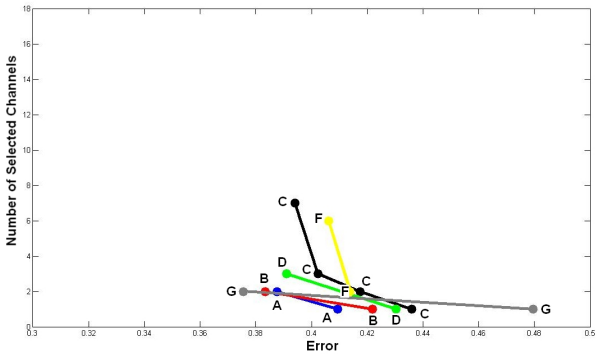


Figure 4.3: Solutions obtained by Binary-SDMOPSO. The approximated Pareto front related to every subject is marked by the corresponding letter.

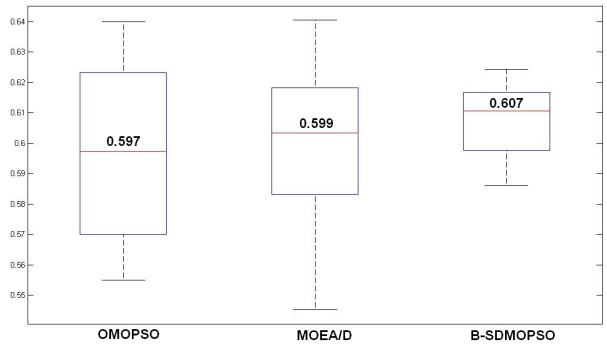


Figure 4.4: Box plot of the accuracies achieved using the three methods. The average accuracy using each of the method is also shown.

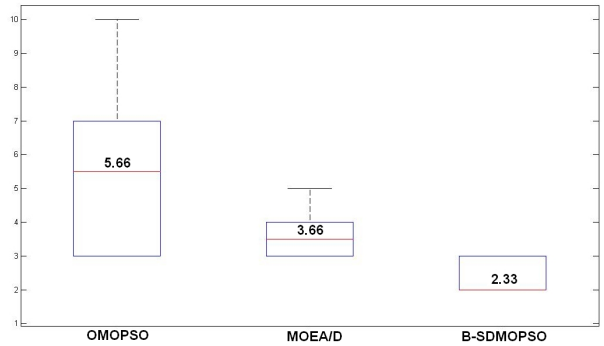


Figure 4.5: Box plot of the number of selected channels achieved using the three methods. The average number of selected channels using each of the method is also shown.

The application of Wilcoxon sign rank statistical test shows no significant difference in terms of accuracy (p=0.5625 > 0.05 between OMOPSO and Binary-SDMOPSO and p=0.6875 > 0.05 between MOEA/D). In terms of the number of selected channels, there is a significant difference between OMOPSO and Binary-SDMOPSO (p=0.0313 < 0.05) but none between MOEA/D and Binary-SDMOPSO(p=0.125 > 0.05). Figure 4.4 and Fig. 4.5 show box plots to compare among the three methods.

4.2.4 Continuous Presentation for Multi-Objective Channel Selection in Brain-Computer Interfaces

The modeling of the channel selection problem so far is similar to that in single-objective case: the number of variables is equal to the number of channels with each channel can be either selected or not. The goal of the optimization method is to minimize two objectives: the number of channels and the error rate. The drawback of this modelling is that it does not count for the spatial relations among the channels making it prone to selecting outlier channels (i.e. channels that are known to have no correlation with the performed mental tasks).

A new presentation of the channel selection problem is presented based on the projection of the real channel positions in 3D into a two dimensional space. First the maximum number of channels to select is defined by the variable C. The number of decision variables would then be $2C$ as each channel is represented by its x and y coordinates in a projected 2D space of the channels 3D locations. The variables are real variables and can take any value within the space of the EEG cap. Each channel location is surrounded by an inclusion circle with radius R. A solution is defined as a set of tuples

$$s_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_C, y_C)\} \tag{4.4}$$

Each tuple (x_i, y_i) is a point in the 2D cap space and is considered a selected channel if it falls within an inclusion circle. Should the point fall into two inclusion circles, the closest channel location is selected using Euclidean distance. Duplicated selected channels are ignored when calculating the objective values.

The first objective function is the classification error rate defined as $E = 1 - CV$ where CV is the cross-validation result. The second objective function is the number of selected channels $C' \leq C$. The optimal solution(s) would have minimum number of channels with lowest error rate and hence minimize E and C'.

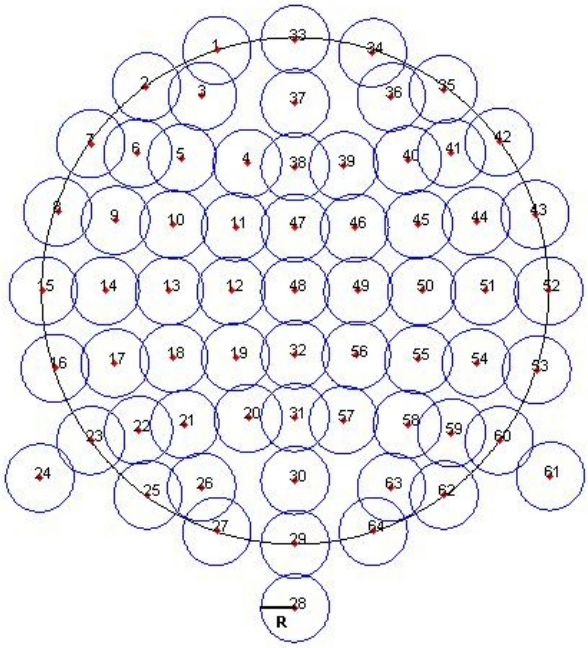


Figure 4.6: Projected Biosemi 64+2 EEG channel locations. The numbering scheme follows the standard Biosemi numbering. Inclusion circles are drawn around each channel.

D^2MOPSO is used as the multi-objective optimizer based on particle swarm optimization. For solving multi-objective channel selection problem, the particles in D^2MOPSO move continuously in the projected 2D space to locate the channels that achieve lowest error rate with fewer channels. This approach is tested on 10 participants performing right vs left motor-imagery tasks and the results are compared to Sequential Floating Forward Search (SFFS) based method.

In addition to combining dominance and decomposition, D^2MOPSO normalizes the MOP objectives. This ensures equal priorities for all objectives, thereby preventing one objective from dominating the others when the aggregation is applied. The objective values are normalized using a sigmoid limiting transformation function defined in Eq.4.5. The Sigmoid limiting transformation is chosen as it does not need any prior knowledge of the objectives' ranges.

$$S(f_i(\overrightarrow{p})) = \frac{1}{(1 + e^{-f_i(\overrightarrow{p})})} \tag{4.5}$$

The normalized value of each objective is used instead of the objective values: $S(F(\overrightarrow{p})) = (S(f_1(\overrightarrow{p}), f_2(\overrightarrow{p}), \ldots, f_m(\overrightarrow{p})))$ instead of $F(\overrightarrow{p})$.

The particle's position is a solution to the channel selection problem. The position is presented as:

$$\begin{aligned} \overrightarrow{p} &= \{x_1, y_1, x_2, y_2, \ldots, x_C, y_C\} \\ &= \{p_1, p_2, p_3, y_4, \ldots, p_{2*C-1}, p_{2*C}\} \end{aligned} \tag{4.6}$$

where C is the maximum number of selected channels, x_i, y_i are coordinates in the cap space as defined in Eq.4.4.

4.2.5 Experiments and Results

4.2.5.1 Sequential Forward Floating Search

In order to have a baseline comparison of the new method, Sequential Forward Floating Search (SFFS) approach is used. SFFS is a comprehensive search single objective algorithm. It starts by selecting the single best channel (with CV as the search criterion). The algorithm continues by combining the selected channel with all the non-selected channels one by one and as a result selects the best two channels and so on. This is

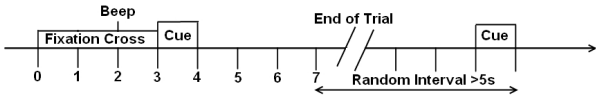


Figure 4.7: The structure of the synchronous trials

called the growing phase as opposed to the pruning phase in which a channel is removed from the selected set of channels and the criterion is checked again. If the criterion has a higher value with lower number of channels then the new channels set is adopted. The algorithm alternates between growing and pruning phases until a maximum number of channels N is selected or the maximum number of iteration is reached (Gan *et al.* (2011)).

4.2.5.2 Data Recording and Pre-processing

EEG data were recorded from 10 healthy subjects using a (64+2)-channel Biosemi system. Standard synchronous motor-imagery training was used (Pfurtscheller & Neuper (2001)) with two motor-imagery tasks: left hand, and right hand. No feedback sessions were recorded. Figure 4.7 shows the structure of the trials.

Data were originally recorded at $256Hz$ but downsampled to $25Hz$ after feature extraction. Common reference was used in this study. Butterworth bandpass filter ($1-45Hz$) is used to remove possible external interference. No artifact removal methods were applied but data were visually checked.

For each subject, data were recorded over 4 sessions with 10 minutes break in between. Every session consisted of 20 trials per class. In total 160 trials were recorded. The first 4 seconds of every trial are ignored as it does not contain any task-related information (Pfurtscheller & Neuper (2001)).

In order to get the channel locations a Biosemi 64+2 channels locations file provided by EEGLAB (http://sccn.ucsd.edu/eeglab/channellocation.html) is utilized. This provides a three-dimensional view of the channels which is projected onto a two-dimensional plane that goes through the assumed center of the brain (i.e. the origin used to define the location of the channels in the 3D space) and parallel to the XY plane.

4.2.5.3 Feature Extraction and Classification

$\mu(8 \sim 12Hz)$ and low $\beta(13 \sim 16Hz)$ rhythms are extracted from each channel by applying a bandpass FIR filter at the corresponding frequency band. The filtered data are squared and then averaged within consecutive time intervals (Kilmesch (1999)).

If a channel is selected both its μ and β features are used. Extracted features from the selected channels are combined together to form one feature vector of maximum 20 features (the maximum number of selected channels is set to 10). Linear Discriminant Analysis (LDA) was used to classify the extracted features with an averaging window of 1 second to smooth the classifier output. 4-fold cross validation is applied, based on trial by trial classification, and is used to calculate the error rate.

4.2.5.4 D^2MOPSO parameter settings

D^2MOPSO employs a swarm of 100 particles that evolve through 100 generations. The inertia weight W is set to a random value in the range $[0.1, 0.5]$, C_1 and C_2 are set to 2, r_1, r_2 are set to random values in $[0, 1]$, and α and β are set to 10%. The algorithm was run 10 times per subject to avoid any bias due to the random initialization.

The maximum number of possible selected channels is set to 10 which means the solution space, in which the swarm evolves, has 20 dimensions (see Eq.4.6). Decision variables ($p_i : i \in [1, 2C]$) are bounded by an upper and lower limit to constrain these variables within the space of the EEG cap.

4.2.6 Results

First the results using SFFS are presented as a baseline to check the validity of the results obtained using D^2MOPSO. Table 4.4 lists the results achieved on the 10 subjects. Figure 4.8 shows a histogram of the selected channels over all 10 subjects with a Gaussian fit of the distribution of the selected channels.

Figure 4.9 plots the results obtained using D^2MOPSO, where the results of each subject is plotted in distinct color with a ploynominal fit of degree 2 of the result to show the approximated Pareto Front for each subject.

To compare with SFFS in terms of classification accuracy only, Table 4.5 shows the maximum accuracy achieved for each of the 10 subjects. A two-sided t-test shows a significant enhancement ($p = 0.04 < 0.05$) of classification accuracy (taking the max

Table 4.4: Results using SFFS

Subject	Cross Validation Accuracy%	Number of Channels
Subject-1	67.05	10
Subject-2	68.17	10
Subject-3	80.17	10
Subject-4	67.20	10
Subject-5	75.29	7
Subject-6	65.16	8
Subject-7	73.71	10
Subject-8	67.69	6
Subject-9	72.22	8
Subject-10	62.37	9
Average	69.90	8.8
Std	5.34	1.47

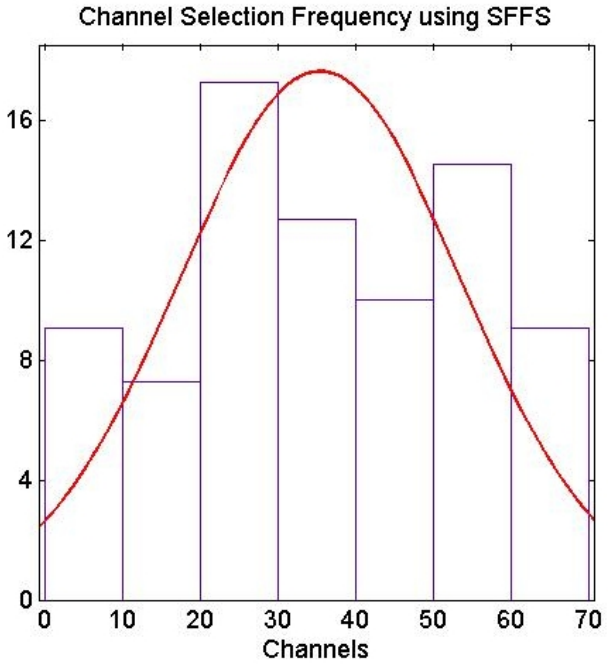


Figure 4.8: Frequency of Channels selected via SFFS

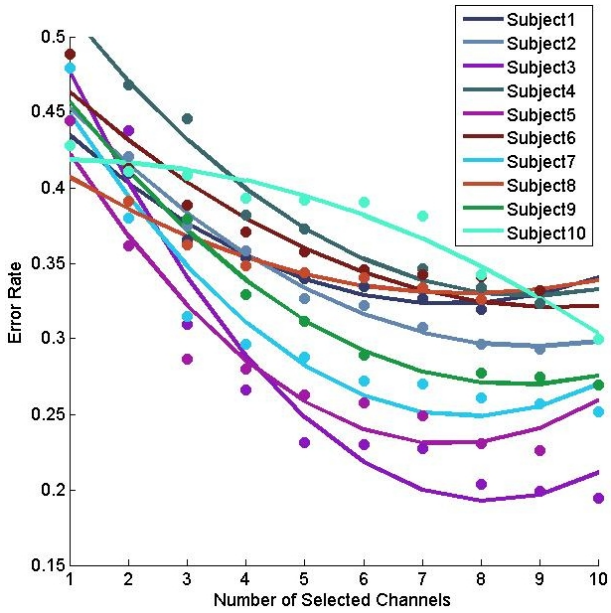


Figure 4.9: Results using D^2MOPSO. Results of each subject are plotted with a polynominal fit of degree 2 to show the approximated Pareto Front.

accuracy for D^2MOPSO) which is supported by Figure 4.10 (the majority of points are under the unity line). The number of channels is harder to compare as D^2MOPSO provides a wide range of solutions as shown in Fig. 4.9, but comparing with the highest accuracy obtained by D^2MOPSO shows insignificant difference ($p = 0.1527 > 0.05$) suggesting that D^2MOPSO is capable of achieving higher classification accuracy for the same number of channels.

Figure 4.11 illustrates the distribution of channels selected for 10 subjects and using all the solutions obtained in the 10 runs per subject (after removing any dominated solutions), showing a wider distribution than that in Fig. 4.8. This can be interpreted as D^2MOPSO is able to widen its search space and provide more diverse solutions which reflects the goal of the continuous problem presentation in the first place. Looking at Fig. 4.6 it is clear that channels in the motor cortex area have diverse channel numbers which explains why the wider distribution of channels in Fig. 4.11 achieves higher classification accuracy.

Table 4.5: Maximum Results using D^2MOPSO

Subject	**Cross Validation Accuracy%**	**Number of Channels**
Subject-1	68.03	10
Subject-2	70.70	9
Subject-3	80.11	9
Subject-4	66.78	9
Subject-5	77.39	10
Subject-6	70.03	10
Subject-7	74.81	10
Subject-8	67.39	9
Subject-9	73.07	10
Subject-10	70.03	10
Average	71.83	9.6
Std	4.43	0.52

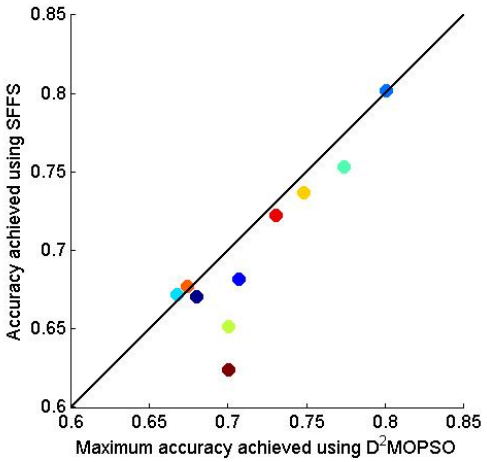


Figure 4.10: Comparison between accuracy results obtained using SFFS and D^2MOPSO

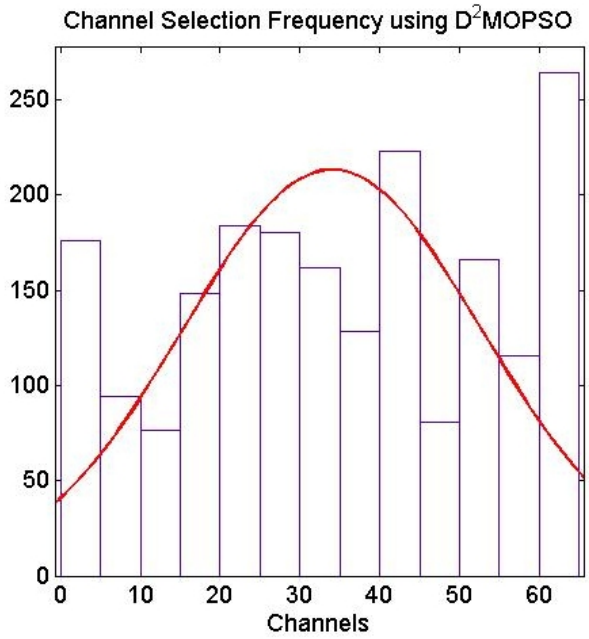


Figure 4.11: Frequency of Channels selected via D^2MOPSO

4.3 Finding Effective Cancer Chemotherapeutic Treatments:

This section presents a novel approach to optimising doses in cancer chemotherapy with respect to conflicting treatment objectives aimed at reducing the number of cancerous cells while limiting the amounts of anti-cancer drugs used. The approach is based on a customized version of $SDMOPSO$. The novelty of the algorithm is in providing particles in the swarm with information from a set of defined neighbours and leaders that assists in finding versatile chemotherapeutic treatments.

As demonstrated in the previous section $SDMOPSO$, implemented in its original and binary form, provides an effective tool for addressing multi-criteria decision-making problems that have a vast space of possible solutions, but cannot be solved by traditional optimisation methods due to either difficulties in obtaining gradients of objective functions or the presence of a large number of constraints. One example from such a class of problems is the optimisation of chemotherapeutic treatments that use multiple anti-cancer drugs - the size of the solution space increases exponentially when new drugs are being added. At the same time, the exploration of the solution space is hindered by various treatment constraints and the necessity to consider several optimisation

objectives, often of conflicting nature.

Traditional optimisation methods (gradient-based or simple heuristics) cannot find optimal solutions for this problem. Similarly, the methods of mathematical programming are impeded by the multiplicity of feasible regions in the solution space. On the other hand, it has been found in (A. Petrovski & McCall (2004); Petrovski & McCall (1999, 2001) and Tan *et al.* (2002)) that population-based computational intelligence algorithms (GA and PSO in particular) provide a good alternative to the conventional optimisation methods for a class of non-linear, multi-constrained chemotherapy design problems. However, the efficiency of conventional GA and PSO algorithms in finding good chemotherapeutic treatment schedules remains an issue, as well as their capability of handling conflicting optimisation objectives. For this reason, this section examines what potential benefits the smart multi-objective PSO with decomposition (SDMOPSO) can bring to bear on the cancer chemotherapy problem. Furthermore, given its performance in binary-coded search spaces, SDMOPSO can also be applied to a wider class of dynamic chemotherapy optimisation problems. One examples of such a problem is given by Allmendinger & Knowles (2010), where the authors are trying to identify efficacious drug combinations drawn from a non-static set of anti-cancer agents represented by a binary string.

4.3.1 Cancer Chemotherapy

Cancer chemotherapy is an inherently complex treatment modality, in which many factors are involved in determining its success or failure. In particular, chemotherapy carries a high risk due to drug toxicity, i.e. the more effective drugs tend to be more toxic (Feng & Chien (2003)). As a result, finding effective chemotherapeutic treatments is a multi-objective optimisation problem affected by such factors as the drugs used, the condition of a patient, the drugs' dosages, their form and schedule. In the next subsections we will clarify some of the related issues that are going to be used for modelling and optimisation of cancer chemotherapy.

4.3.1.1 Medical Aspects of Chemotherapy

Drugs used in cancer chemotherapy all have narrow therapeutic indices. This means that the dose levels at which these drugs significantly affect a tumor are close to those

levels at which unacceptable toxic side-effects occur. Therefore, more effective treatments result from balancing the beneficial and adverse effects of a combination of different drugs, administered at various dosages over a treatment period (Petrovski & McCall (2001)).

The beneficial effects of cancer chemotherapy correspond to treatment objectives which oncologists want to achieve by means of administering anti-cancer drugs. Nominally, the optimal treatment schedule is the one found to be the most efficacious from the set of schedules evaluated during clinical trials. For new drugs, Phase I trials evaluate the toxicity of an anti-cancer drug on a given schedule, and Phase II trials establish the efficacy of the schedule (Harrold & Parker (2009)). Several schedules may be considered, and the schedule yielding the most promising result in a statistically controlled trial is considered optimal from a clinical perspective. Novel chemotherapy schedules found to be efficacious in Phase II trials are compared to the current standard of practice against specific forms of cancer in Phase III trials. Similarly, for approved drugs, chemotherapeutic treatment is generally broken down into cycles. At the end of the first cycle the main objective is to evaluate toxicity, and at the end of the second - to determine the efficacy of the drug schedules.

The main purpose of the present paper is to investigate how the techniques of Computational Intelligence can assist in multi-objective optimisation of chemotherapy schedules that use approved, rather than novel, anti-cancer drugs. Therefore, the main treatment objective we are going to be interested in are controlling the drug toxicity and improving treatment efficacy in terms of the reduction of tumour size. The next subsection mathematically describes these objectives and associated constraints.

4.3.1.2 Problem Formulation

The adverse effects of cancer chemotherapy stem from the systemic nature of this treatment: drugs are delivered via the bloodstream and therefore affect all body tissues. Since most anti-cancer drugs are highly toxic, they inevitably cause damage to sensitive tissues elsewhere in the body. In order to limit this damage, toxicity constraints need to be placed on the amount of drug applied at any time interval, on the cumulative drug dosage over the treatment period, and on the damage caused to various sensitive tissues (Wheldon (1988)). In addition to toxicity constraints, the tumour size (i.e. the

number of cancerous cells) must be maintained below a lethal level during the whole treatment period for obvious reasons.

The goal of cancer chemotherapy therefore is to achieve the beneficial effects of treatment objectives without violating any of the constraints mentioned above. In order to solve the optimisation problem of cancer chemotherapy, we need to find a set of treatment schedules, which satisfy the toxicity and tumour size constraints yielding at the same time acceptable values of treatment objectives. This set will allow the oncologist to make a multi-criteria decision on which treatment schedule to use, given his/her preferences or certain priorities. In the remainder of this section we will define the search space of the decision vectors for the cancer chemotherapy optimisation problem, specify the constraints imposed on these vectors, and particularise the optimisation objectives.

Anti-cancer drugs are usually delivered according to a discrete dosage programme in which the doses are administered at times t_1,t_2,...,t_n (Martin & Teo (1994)). In the case of multi-drug chemotherapy, each dose is a cocktail of d drugs characterised by the concentration levels C_{ij}, where $i \in 1, n$ and $j \in 1, d$, of anti-cancer drugs in the bloodplasma. Optimisation of chemotherapeutic treatment is achieved by modification of these variables. Therefore, the solution space Ω of the chemotherapy optimisation problem is the set of control vectors $\mathbf{c} = C_{ij}$ representing drug concentration profiles. However, not all of these profiles will be feasible as chemotherapy treatment must be constrained in a number of ways. Although the constraint sets of chemotherapeutic treatment vary from drug to drug, as well as with cancer type, they have the following general form.

- Maximum instantaneous dose C_{max} for each drug acting as a single agent:

$$g_1(\mathbf{c}) = \left\{ C_{maxj} - C_{ij} : \forall i \in 1, n, \forall j \in 1, d \right\} \geq 0 \tag{4.7}$$

- Maximum cumulative C_{cum} dose for drug acting as a single agent:

$$g_2(\mathbf{c}) = \left\{ C_{cumj} - \sum_{i=1}^{n} C_{ij} : \forall j \in 1, d \right\} \geq 0 \tag{4.8}$$

- Maximum permissible size N_{max} of the tumour:

$$g_3(\mathbf{c}) = \left\{ N_{max} - N(t_i) : \forall i \in 1, n \right\} \geq 0 \tag{4.9}$$

- Restriction on the toxic side-effects of multi-drug chemotherapy:

$$g_4(\mathbf{c}) = \Big\{ C_{s-effk} - \sum_{j=1}^{d} \eta_{kj} C_{ij} : \forall i \in 1, n, \forall k \in 1, m \Big\} \geq 0 \tag{4.10}$$

The factors η_{kj} in the last constraint represent the risk of damaging the k^{th} organ or tissue (such as heart, bone marrow, lung etc.) by administering the j^{th} drug. Estimates of these factors for the drugs most commonly used in treatment of breast cancer, as well as the values of maximum instantaneous and cumulative doses, can be found in (Baker (1985)) and (Cassidy & McLeod (1995)), but are also summarised in Table 4.6 and Table 4.7 in Section 4.3.3.

The objectives of cancer chemotherapy optimisation, as was previously mentioned, relate to the efficacy of treatment schedules and their toxicity levels. In order to characterise treatment efficacy, we simulate the response of a tumour to chemotherapy using Gompertz cell-growth model with a linear cell-loss effect due to administration of anti-cancer drugs (Martin & Teo (1994)):

$$\frac{dN}{dt} = N(t).\Big[\lambda\, ln(\frac{\Theta}{N(t)}) - \sum_{j=1}^{d} K_j \sum_{i=1}^{n} C_{ij} \Big\{ H(t-t_i) - H(t-t_{(i+1)}) \Big\} \Big] \tag{4.11}$$

where $N(t)$ represents the number of tumour cells at time t ; λ, Θ are the parameters of tumour growth, $H(t)$ is the Heaviside step function; K_j are the quantities representing the efficacy of anti-cancer drugs, and C_{ij} denote the concentration levels of these drugs. One advantage of the Gompertz model from the computational optimisation point of view is that Eq. 4.11 yields an analytical solution (Martin & Teo (1994)) for $N(t)$.

Then, the efficacy of a chemotherapeutic schedule can be measured as the degree of reduction of the tumour size during the treatment interval, formulated as follows (Petrovski (1999)):

$$minimise\ F_1(\mathbf{c}) = \sum_{i=1}^{n} N(t_i) \tag{4.12}$$

subject to the state equation (Eq. 4.12) and the constraints (4.7)-(4.10).

The second optimisation objective of cancer chemotherapy related to treatment toxicity is proportional to the application time for cytotoxic drugs and the duration of treatment (Martin & Teo (1994)). These measures correspond to the total amount of toxic anti-cancer drugs in the blood plasma and can be modeled by the following objective:

$$minimise\ F_2(\mathbf{c}) = \sum_{j=1}^{d} exp(1 + K_j) \sum_{i=1}^{n} C_{ij}^2 \tag{4.13}$$

subject to the constraints (4.7)-(4.10).

4.3.2 Customized-SDMOPSO for Cancer Chemotherapy Treatment

In order to cope with a high dimensionality of the solution space for the chemotherapy optimisation problem and to avoid premature convergence, SDMOPSO requires tuning the process of updating particles' velocities so that it can effectively use the global information about the search space. This information is presented by the global non-dominated leaders found during the search through the optimisation process. To reduce the computational cost and to enhance the solutions' diversity in the objective space, a very small crowding archive is used "$Archive_{leaders}$"(size $\approx$ 10, ϵ=0.0075) to store the non-dominated solutions found so far.

The customisation of SDMOPSO to the chemotherapy optimisation problem is twofold. Firstly, the customised-SDMOPSO (c-SDMOPSO) integrates the global information into the velocity update equation - each particle in addition to its personal experience and local information uses the global leader information to discover new locations in the solution/objective space. This is done by replacing Eq. 2.3 with the following equation:

$$\begin{aligned} \overrightarrow{v_i}(t) &= W * \overrightarrow{v_i}_i(t-1) + C_1 * r_1 * (\overrightarrow{x}_{pbest_i} - \overrightarrow{x}_i(t)) \\ &\quad + C_2 * r_2 * (\overrightarrow{x}_{nbest_i} - \overrightarrow{x}_i(t)) \\ &\quad + C_3 * r_3 * (\overrightarrow{x}_{gbest_i} - \overrightarrow{x}_i(t)) \end{aligned} \tag{4.14}$$

where $gbest_i$ is a global leader for the $particle_i$, $r_1, r_2, r_3 \in [0, 1]$ are random values, $w \in [0.1, 0.5]$ is the inertia weight, and $C_1, C_2, C_3 \in [1.0, 2.0]$ take uniformly distributed

random values. $gbest_i$ is selected from the leaders' archive ($gbest_i \in Archive_{leaders}$) in such a way that its aggregation value, defined by Eq. 4.15, is the closest one to the aggregation value of $particle_i$ calculated using the λ vector associated with $particle_i$.

As the ranges of the objectives' values can differ considerably and are rarely known a priori for the majority of real life problems, a normalization technique need to be applied to all the objectives before aggregating them. This will ensure equal priorities for both objectives in our case, thereby preventing one objective from dominating the other when the aggregation is applied. The objective values are normalized using a sigmoid limiting transformation defined by Eq.4.1.

The aggregation function will then use the normalized values for each objective as follow:

$$minimize \quad g(\overrightarrow{x}|\lambda, \overrightarrow{z}^*) = \max_{1 \leq i \leq m} \{\lambda_i | S(f_i(\overrightarrow{x})) - \overrightarrow{z}_i^*|\} \tag{4.15}$$

Secondly, when converging to a local optima, the particles will not be able to diverge again because of using the same $gbest_i$, $pbest_i$ and similar $nbest_i$. The c-SDMOPSO algorithm resolves this issue by redefining the position update method - it allows each particles to change its location regardless whether it is a better position in terms of the aggregation value or not. The method still guides the particles in the objective space using the information on best global leaders and neighbours. In c-SDMOPSO the fittest position does not necessary survive, but it is always stored in the archive pool. One particle might move to a worse location in the objective space in order to be able to move to a better one in the next iteration(s).

4.3.3 Experiments and Results

The customised version of SDMOPSO (c-SDMOPSO) is compared with results obtained by NSGAII and MOEA/D (Li & Zhang (2009)).

All methods used have a population or swarm of size 300 individuals/particles and terminate after 600 iterations of the evolution cycle. Each individual/particle contains 100 real variables - a schedule of 10 drugs each of which has 10 doses - with the value in the range $[0, 15]$. The results presented in Fig. 4.12 are the set of non-dominated solutions obtained after 20 different runs of each algorithm.

As stated in Section II, the optimisation problem of cancer chemotherapy we are trying to address has two objectives - minimisation of the area under the tumour curve (Eq.4.12) and minimisation of the toxicity level in the bloodplasma after administration of anti-cancer drugs (Eq.4.13). The details of anti-cancer drugs used in our experiments are shown in Table 4.6 and Table 4.7.

Table 4.6: The side-effects of the drugs used through the treatment

Drugs	Bone marrow	Kidney	Periph.nerves	Liver	Heart
Adriamycin	3	0	0	0	2
Epirubicin	3	0	0	0	1
Taxotere / Taxol	3	0	2	0	1
Cyclophosphamide	2	0	0	0	0
Fluorouracil	0	0	0	0	0
Cisplatinum	1	3	3	0	0
Methotrexate	1	1	0	1	0
Mitomycin-C	2	0	0	1	0
Prednisolon	0	0	0	0	0
Vincristine	0	0	2	0	0

NSGAII uses a Simulated Binary Crossover (SBX) with probability equal to 0.9, and a polynomial mutation with probability equal to the inverse of the number of decision variables (i.e. 1/100); the mutation and the crossover distribution indexes are equal to 20 (Deb *et al.* (2002)).

MOEA/D uses a differential evolution (DE) crossover with probability equal to 1.0 and a differential weight of 0.5, a polynomial mutation with the same probability as that of NSGA-II (i.e. 1/100), the mutation distribution index is equal to 20, and the neighbourhood size is set to 20 (Li & Zhang (2009)).

c-SDMOPSO uses the parameters specified in Eq.4.14 with a neighbourhood size equal to 30 and the $Archive_{leaders}$ size equal to 10.

The approximation of the Pareto fronts (PF) produced by each algorithm is the combination of PFs obtained after each of 20 experimental runs (PF_{approx}).

To validate our approach, two indicators are used. The first performance indicator is the inverted generational distance (I_{IGD}). To apply this measure, all the objective

Table 4.7: Drug profiles of the anti-cancer agents used

Drugs	C_{maxj}	C_{cumj}	P_j	$K_j(*10^{-3})$
Adriamycin	75	550	1	5.605
Epirubicin	75	700	1	4.484
Taxotere / Taxol	100/130	1000/1500	0.75/0.577	7.29
Cyclophosphamide	2000	10000	0.0375	3.9235
Fluorouracil	3000	30000	0.025	2.242
Cisplatinum	120	600	0.625	4.335
Methotrexate	10000	100000	0.0075	1.6815
Mitomycin-C	15	40	5	2.242
Prednisolon	100	1000	0.75	1.121
Vincristine	2	30	200	2.242

values are scaled to be in the range of [0,1]. The second indicator is the cardinality measure $I_{cardinality}(A, B)$, which calculates the percentage of solutions in A that belongs to B, where A is an approximation of the PF produced by a specific method and B is a reference set (El-Ghazali (2009)).

$$I_{cardinality}(A, B) == \frac{|A \cap B|}{|B|} \tag{4.16}$$

In this section I_{IGD} indicator is used to compare PF_{approx} obtained by each algorithm with PF_{ref}, which is a reference set that contains the non-dominated solutions found by the three algorithms restricted to keeping 100 best solutions only. The reference set uses a crowding archive to guarantee the solutions are equally distributed along the PF in the objective space (Sierra & Coello Coello (2005)). Using the reference set as the first argument (A) in Eq.2.17 allows us to evaluate the effectiveness of the solutions by measuring the uniformity of distribution together with the distance to the reference set (see Table 4.8).

The *cardinality* indicator is based on the reference set (PF_{ref}) used by IGD, and compares the PF_{approx} obtained by each algorithm against PF_{ref} (see Table 4.9).

The I_{IGD} and $I_{cardinality}$ values are used to quantitatively evaluate the performance of the c-SDMOPSO algorithm in comparison with that of NSGAII and MOEA/D. As

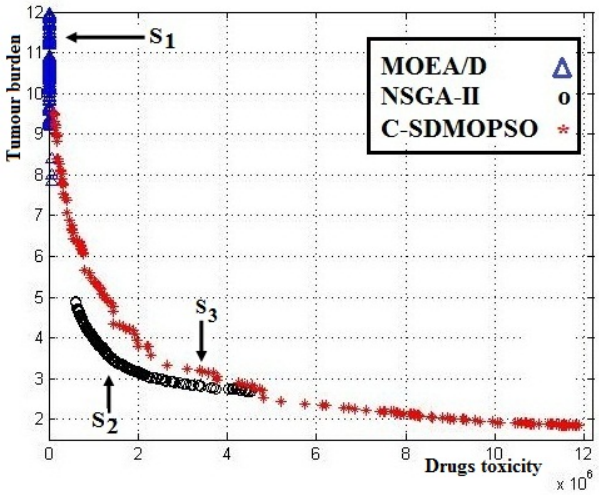


Figure 4.12: Approximated Pareto fronts

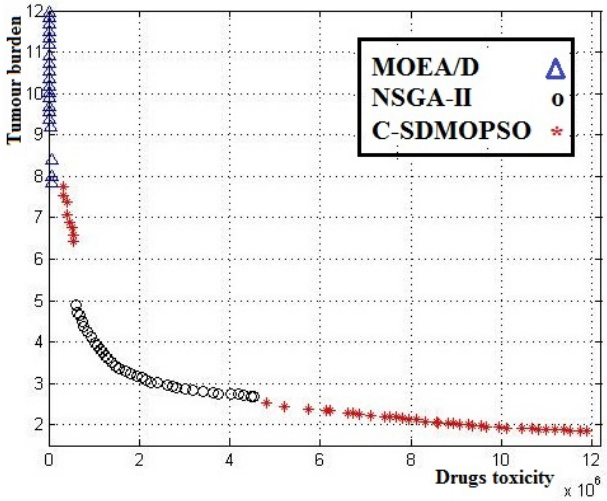


Figure 4.13: The reference set obtained by merging all the non-dominated solutions generated by the three algorithms

Table 4.8: Inverted Average Generational Distance results for the three algorithms

Measure	MOEA/D	NSGA-II	c-SDMOPSO
IGD	0.0633	0.0342	0.0062
mean	0.5121	0.2503	0.0348
max	1.1577	0.6989	0.2393
min	0.0	0.0	0.0
median	0.5139	0.2332	0.0136
Std	0.3753	0.2344	0.0519

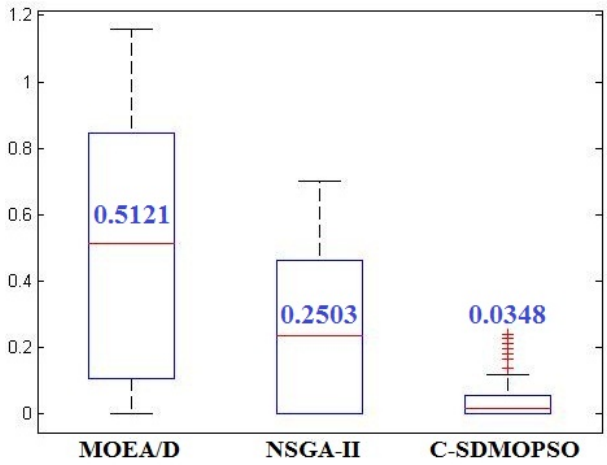


Figure 4.14: Box plot of the IGD achieved using the three methods (the central line in every box represents the median; the average IGD using each of the algorithms is shown as a numeric value)

can be seen from Table 4.8, c-SDMOPSO yields a significantly better measure of I_{IGD}, which indicates a closer approximation of the true Pareto front for the multi-objective optimisation problem under investigation. Furthermore, a better approximation is achieved by c-SDMOPSO consistently across all experimental runs as the box plot in Fig. 4.14 demonstrates.

The approximated Pareto fronts PF_{approx_i} obtained by each algorithm under investigation are shown in Fig. 4.12; these PFs are combined together where only the non-dominated solutions are kept (see Fig. 4.13) to form the reference set (PF_{ref}) used for obtaining quantitative comparative characteristics. A visual analysis of Fig. 4.12

Table 4.9: Cardinality measure results for the three algorithms

MOEA/D	NSGA-II	c-SDMOPSO
20%	36%	44%

reveals that in terms of finding the solution closest to the reference point z^* in Eq. 4.15 (in our case - the origin), NSGA-II produces the best result.

However, if we take into consideration the diversity of the solutions found, it is apparent that the c-SDMOPSO algorithm can provide the oncologists, who make decisions on the most appropriate course of treatment, with a richer choice of options. This is particularly relevant to curative treatments, represented by the trail of solutions in the bottom right corner in Fig. 4.12 - the treatment schedules represented by these solutions use considerable amounts of cytotoxic drugs and thus are primarily aimed at curing cancer.

Moreover, if we examine the composition of the Pareto reference set PF_{ref}, shown in Fig. 4.13 and summarised in Table 4.9, it becomes clear that almost half of the solutions (44%) in the PF_{ref} have been found using c-SDMOPSO, proving the effectiveness of this algorithm in the context of multi-objective chemotherapy optimisation.

Figure 4.15, Fig. 4.16 and Fig. 4.17 show the actual treatment schedules found by MOEA/D (S_1), NSGA-II (S_2), and c-SDMOPSO (S_3) respectively, which have also been highlighted in Fig.4.12. These schedules represent dose sequences of anti-cancer drugs listed in Tables 4.6 and 4.7 (the character strings along the y-axis contain the first three letters of the drug names).

As can be seen from Fig. 4.15, MOEA/D prioritises the second optimisation objective (see Eq.4.13) that tries to minimise treatment toxicity. The schedule uses small amount of drugs, and the only drug, substantial doses of which are administered, is Prednisolon that does not have toxic side-effects on any of the organs listed in Table 4.6.

Figure 4.16 shows the treatment schedule found by NSGA-II. An interesting observation that can be made from this schedule is that it uses substantial doses of seven (out of ten) drugs, but avoids administering the anti-cancer agent (Taxotere/Taxol) with severe toxic effects on the majority of organs that are taken into account (see Table 4.6).

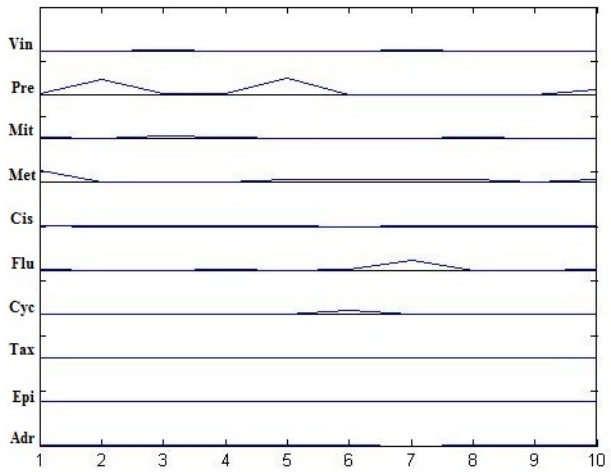


Figure 4.15: Drug doses for one of MOEA/D solutions

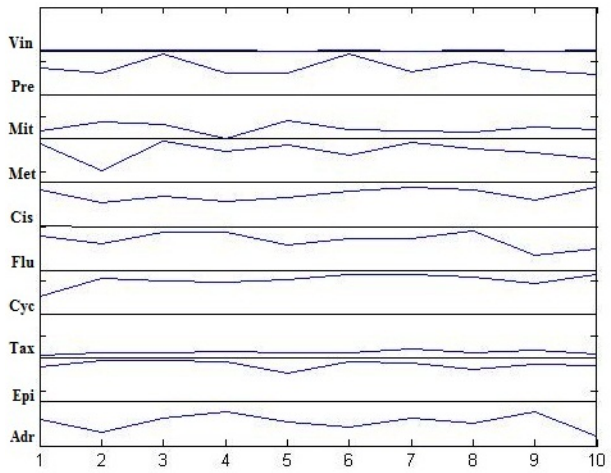


Figure 4.16: Drug doses for one of NSGA-II solutions

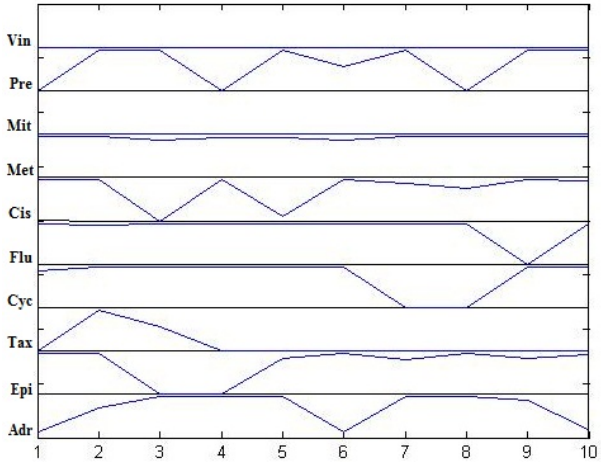


Figure 4.17: Drug doses for one of c-SDMOPSO solutions

An interesting compromise is shown in Fig. 4.17, where one of the treatment schedules found by c-SDMOPSO is illustrated. The schedule uses a smaller overall amount of drugs, but initiates the treatment with two doses of Taxotere/Taxol, followed by the administration of less toxic drugs.

4.4 Conclusion and Discussion

This section demonstrated the potential impact of the developed algorithms in real-life applications. Despite the occasional need of modifying the algorithms to better suit some problems the general framework of the algorithms have shown to be robust and competitive to the state-of-the-art.

The first problem, channel selection for BCI, was first solved using a traditional binary representation which required a binary version of $SDMOPSO$. Binary-SDMOPSO employs the velocity equation defined in Section 3.2. Binary-SDMOPSO replaces Eq. 2.3 with Eq. 4.1 and Eq. 4.3 in order to update the particle positions. Eq. 4.1 is a limiting transformation function that normalizes the velocity value. Eq. 4.3 updates the corresponding bit of the particle position directly by cloning bit from *pbest*, *nbest* or keep it without change according to the static probability variable α.

In addition to that, Binary-SDMOPSO takes advantage of the simplicity of SDMOPSO, it does not use any evolutionary operators but instead it handles the early convergences problem by making every particle in the swarm to pre-process its moves

and to share this information with its neighbours. This results in a better exploitation of the local information, which alleviates the effect of premature convergence to local optima.

The results presented chapter show that Binary-SDMOPSO is competitive to other state-of-the-art multi-objective optimization methods on the problem of channel selection for BCI. Although there was no significant difference in the classification accuracy a significant difference in the number of selected channels was demonstrated between OMOPSO and Binary-SDMOPSO with no significant difference with MOEA/D.

The results of Binary-SDMOPSO support the previous findings in (Hasan *et al.* (2010)) on the usefulness of multi-objective view of the channel selection problem in BCI. In general, MOEA/D and Binary-SDMOPSO were less prone to outliers (channels that can not be credited to motor-imagery movements) than OMOPSO. This is reflected by the smaller number of channels selected and their distribution.

A continuous presentation of the channel selection problem in Brain-Computer Interfaces was also introduced within the framework of D^2MOPSO to offer a more natural way of solving the problem aiming at higher classification accuracy with lower number of selected channels. The usage of this presentation is not limited to D^2MOPSO but can easily be adopted in any other multi-objective (or even single objective) optimizer capable of solving continuous problems (e.g. GA).

The continuous presentation has an advantage of better exploring the search space by exploiting the spatial relationships among channels rather than looking at the channel selection problem as a discrete problem where channels are considered spatially independent, which is demonstrated in Figure 4.11 and Figure 4.8.

In a typical scenario, a particle (assumed to be looking for one channel for simplicity) will start from a random location within the 2D cap space and then navigate through the channels to find the one channel that achieves the highest accuracy. When more channels are needed the particle will use this navigability mechanism to maintain the spatial relationship among channels resulting in a much homogeneous set of channels for classification.

The results presented in Fig. 4.9 show predicted Pareto fronts (i.e. error rate goes down with the increase of number of channels but only to a certain limit where the error rate can increase again) for most subjects with some exceptions (e.g. Subject10 which showed an increase in error rate with the increase in the number of channels).

The effectiveness of the multi-objective approach is minimizing the number of channels that can be used with insignificant sacrifice in accuracy is not discussed in the continuous presentation study in detail as this was previously studied and demonstrated in (Hasan *et al.* (2010); Moubayed *et al.* (2010)). However, Figure 4.9 details the different solutions obtained for each subject and it shows for most subjects similar results to these found in (Hasan *et al.* (2010); Moubayed *et al.* (2010)).

The results presented here are based on synchronous BCI design (i.e. the timing is controlled by the system) for the continuous presentation but self-paced (i.e. the timing is controlled by the participant). The Synchronous problem is usually easier to study and analyse. For this reason it was chosen for the continuous presentation study to eliminate any implicit bias caused by the self-paced paradigm.

The chapter then moves to describe a customised version of Smart Multi-Objective Particle Swarm Optimisation using Decomposition (c-SDMOPSO) and shows how it can be applied to the multi-objective problem of cancer chemotherapy optimisation. The novel features of c-SDMOPSO relate to the way how this algorithm integrates the global information, the personal particle experience, and the local neighbourhood information in the process of updating the velocities and positions of swarm particles so that they are directed to potentially better regions in the solution space. The global information is represented by the location of the swarm leaders, where the leaders selected are non-dominated particles equally distributed in the objective space. Also, the c-SDMOPSO algorithm normalises the values for different optimisation objectives before their aggregation in order to provide equal objective priorities. In order to avoid premature convergence, the particles in c-SDMOPSO update their positions regardless of their aggregation values.

To demonstrate the effectiveness of c-SDMOPSO applied to the optimisation of chemotherapeutic treatments, its performance is compared with that of two other Computational Intelligence algorithms used for multi-objective optimisation and in multi-criteria decision-making, namely NSGA-II and MOEA/D. The results obtained during the comparative study demonstrate that c-SDMOPSO finds solutions to chemotherapy optimisation problem of similar quality, but in larger quantities. One practical implication of this outcome is the possibility of providing a more advanced decision support to the oncologists involved in seeking the most suitable treatment strategies.

Further work in this problem domain could explore multi-objective optimisation of combined cytostatic and cytotoxic cancer chemotherapy as suggested in (Villasana *et al.* (2010)). This would necessitate the use of more complex mathematical models describing drug actions that take into account pharmacokinetic/pharmacodynamic characteristics of anti-cancer agents and the occurrence of drug resistance in tumour cells.

Another direction of research relates to the ability of Computational Intelligence algorithms to incorporate user preferences (in terms of optimisation objectives and control factors) that can change over time, and to prune the solutions found so that the decision-maker is provided with sufficiently dissimilar (Liang *et al.* (2008)) options applicable to different treatment scenarios.

The application of MOEA on real-life problems raises a serious concern regarding the evaluation of the solutions generated by the optimizers. Different quality assessment measures usually results in conflicting conclusions with some using an indicator approach and others a probabilistic approach. The next chapter discusses the limitations of the state-of-the-art in performance assessment for MOEA and proposes a mutual-information based alternative which defines an indicator that utilizes probability density functions. The defined indicator is more robust and reliable comparing to the state of the art as the next chapter will show.

Chapter 5

Mutual Information for Performance Assessment of Multi Objective Optimisers

"The problems are solved, not by giving new information, but by arranging what we have known since long. "

Ludwig Wittgenstein

5.1 Introduction

So far the quality assessment of the solutions produced by the methods developed in this thesis is based on established quality measures. Assessing the quality of multi-objective solutions, however, and comparing the performance of different multi-objective optimisers is still not very well understood. Current trends either model the outcome of the optimiser as a probability density function in the objective space, or defines an indicator that quantify the overall performance of the optimiser. In this chapter an approach based on the concept of mutual information is proposed. The approach models the probability density function of the optimisers' output and use that to define an indicator, namely the amount of shared information among the compared Pareto fronts. This new approach is tested on several datasets within a unified framework to compare it with other quality measures. The strength of the new approach is not only in better assessment of performance but also the interpretability of the results it provides.

In the infancy of this field of research visual comparison was used as the norm to qualify the performance of the optimizers. The field, however, has grown out of this simple approach and a separate field has developed to assess the performance of multi-objective optimizers.

As with single objective optimisation, two factors are important when assessing a multi-objective optimiser: the quality of the found solutions, and the time spent to find them. However the stochastic nature of evolutionary algorithms results in the relation between time and quality not fixed, but rather represented by a probability distribution function. Hence, when discussing quality in evolutionary algorithm we need to look at it from a probabilistic point of view. In addition having a set of solutions (Pareto front) instead of a single outcome of the multi-objective optimisation process makes quantifying the quality of these solutions much harder. This is added to having multiple runs and the necessity to statistically quantify the behaviour of the optimiser over these runs increases the difficult of quality assessment (Fonseca *et al.* (2005b)).

According to a review by Fonseca *et al.* (2005b), there are two main approaches for quality assessment: a) model the outcome of the optimizer as a probability density function in the objective space. b) the indicator approach which quantifies the outcome of a run with a number with statistical analysis applied to these performance values.

Several studies emphasized the importance of design and application of quality measures and especially how the statistical tests are applied and interpreted (Knowles & Corne (2002); Okabe *et al.* (2003); Zitzler *et al.* (2003b)).

This chapter introduces a novel approach for performance assessment of multiobjective optimizers. The approach uses mutual information as a measure which combines the probabilistic approach and the indicator approach. Calculating mutual information requires probabilistic modelling the measured variables and then a mutual information based indicator is defined based on this modelling. Initially the objectives are considered independent and mutual information is calculated separately for each objective and then a weighted sum approach is used to define an indicator. However, this approach has some disadvantages which are discussed and an alternative indicator is proposed which models the Pareto fronts as images. Mutual information is then calculated between these images. To validate and test this method, eight problems are tested using three popular multi-objective optimizers and the results are tested using three popular indicators in addition to mutual information.

5.2 Measuring Quality of Multi-objective Optimizers

In principle the easiest way to compare between two multi-objective optimizers' performance is by comparing directly the resulted output of the two methods (e.g. using an indicator of quality like the ones will be discussed later). This would work if the optimizers are deterministic, i.e. running the optimizer twice will return the same results.

Multi-objective evolutionary algorithms are stochastic in nature, due to the random element in the algorithms, i.e. running the algorithm twice would most likely produce a different set of results. For this reason the optimizer should be run several times and the probability density function is then empirically estimated. Comparing two optimizers would then mean comparing their probability density functions which then implicate the issue of statistical hypothesis testing (Fonseca *et al.* (2005a)).

In the literature, there are two main approaches to assess the quality of produced PFs. The most common one is the indicator approach where a PF is mapped, using a defined function, to a real number then a standard statistical hypothesis test is applied on the indicator values. The second approach is usually referred to as the attainment

function method in which for each objective vector there is a probability p that the produced approximation set contains an objective vector that weekly dominates z. The attainment function then gives a probability estimate of z to be attained in one optimization run with a statistical test procedure to count for all the runs (Grunert da Fonseca *et al.* (2001)).

Using quality indicators is an attractive approach of quality assessment due to its simplicity. It has , however, some shortcomings:

- each indicator looks at the performance from only one perspective, e.g. spread, diversity, or dominance, which may skew the conclusions drawn.
- In the case of incomparable PFs an indicator will actually give an inaccurate result.
- For indicators that use distance functions, outliers can cause a real problem in disturbing the calculation of the indicator.
- Quality indicators do not take the statistics of the data in the objective space into account which can be vital information to properly assess the quality of the performance.

Fonseca *et al.* (2005b) discussed other important aspect that should be considered when using quality measures such as scaling and normalization and the possible combination of quality measures.

The attainment function method, on the other hand, distances itself from the indicator approach by estimating probability density functions of the attained objectives and then apply statistical tests to compare among the different PFs, which circumvents most of the issues raised by the use of quality indicators but on a high computational expenses that they cannot be used for large number of objectives.

The approach adopted models the output of the optimizer directly as an empirical probability density function and then calculates the mutual information between the approximated PF and the theoretical one. The higher the mutual information the closer the approximation to the theoretical.

5.3 Methods

In this section we begin by defining the theoretical concepts behind mutual information. This is followed by description of how these definitions can be used to assess the quality of multi-objective optimizers.

5.3.1 Mutual Information

Intuitively speaking, mutual information measures how much information are shared between two random variables X, Y. In other words how much knowledge of one variable reduces the uncertainly about the other. Formally, mutual information is defined as follows:

$$I(X,Y) = \sum_{Y}\sum_{X} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{5.1}$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively with summation replaced by an integral in the case of continuous random variables.

$I(X,Y)$ measures the distance between the joint distribution/density functions of X and Y. And since mutual information is measuring distance it is always positive, i.e. $I(X,Y) \geq 0$, and symmetric, i.e. $I(X,Y) = I(Y,X)$.

If the two variables are dependent, then $I(X,Y)$ measures the shared information between the two variables and it would be positive. If on the other hand, they are independent, then $I(X,Y) = 0$.

In practice and for numerical stability during implementation the multiplication/division of probabilities is rewritten as a subtractions of *log* probabilities:

$$I(X,Y) = \sum_{Y}\sum_{X} p(x,y)\Big(\log p(x,y) - \log p(x) - \log p(y)\Big) \tag{5.2}$$

To complete the calculation of MI, the joint and marginal probabilities should be calculated. To estimate the joint probability distribution/density function a two dimensional histogram is used (Scott (1979)). The histogram is a classical non-parametric density estimator. It estimates the probability density function of a random variable X from a set of observed values $\{x_1, \ldots, x_n\}$ by dividing the space between min_{x_i} and max_{x_i} into L bins and then counting the number of observed values x_i that fall into

each bin. Once the density of $p(x, y)$ is estimated the marginal probabilities are easily calculated as $p(x) = \sum_y p(x, y)$ and $p(y) = \sum_x p(x, y)$. This approach is similar to the approach used in (Peng *et al.* (2005)).

The mutual information as defined in the present context is not normalized, i.e. it can take any positive value. Here we use a normalized version as defined in Witten & Frank (2005):

$$U(X, Y) = 2\frac{I(X, Y)}{H(X) + H(Y)} \tag{5.3}$$

where $H(X)$, $H(Y)$ are the marginal entropies of X and Y respectively. Please refer to Witten & Frank (2005) for more on how to calculate the entropy function.

Mutual information has drawn a lot of attention mainly due to its simplicity and robustness. It has been applied in a wide range of applications, e.g. the image registration of Magnetics Resonance Imaging (MRI) (Frackowiak (2004)) and the dimensionality reduction through feature selection in classification (Peng *et al.* (2005)).

5.3.2 Measuring Quality with Mutual Information

The way mutual information is applied would differ depending on the independence assumptions among the objectives. The definition in Sec 5.3.1 is only valid for univariate random variables, so if we work on the assumption that the objectives are all independent then mutual information can be measured separately between the approximated PF and the true PF, one objective at a time and then the mutual information indicator is defined as:

$$I_{MI}(A, B) = \sum_{i=1}^{n} \alpha_i U(A_i, B_i) \tag{5.4}$$

where A is the approximated PF, B is the true PF, A_i is the values of objective i from the PF A, U is the normalized mutual information function defined in Eq.5.3, n is the number of objectives, and α is the weight vector where $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha = 1$. The goal of the weight vector is to account for the cases where not all objectives have the same priority so the objective with higher priority is given higher weight value.

The result of this indicator quantifies how much the approximated PF reduces uncertainty about the true PF. The higher I_{MI} then the better approximation A is to the true PF.

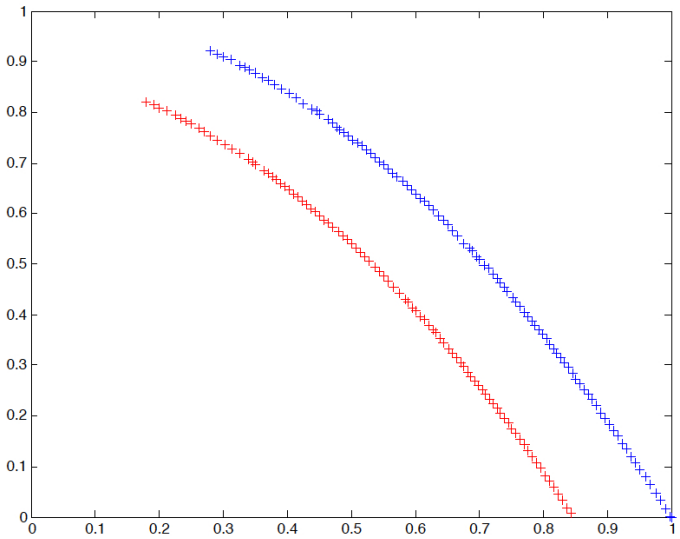


Figure 5.1: An example of two unequal PFs with the same histogram. The blue PF (A) is the true PF for ZDT1 and the red PF (B) is a shifted copy of the blue PF. I_{MI}(A,A) = I_{MI}(A,B) = 1.

On the positive side, this is a simple measure that can be applied to any MOP regardless of the shape of the Pareto front. It does not require a reference point and does not depend on one aspect of the data like spread or diversity. However, if the number of points in either A or B is low then the estimation of the probabilities will not be reliable and so would be I_{MI}, but this is not the case in the majority of MOPs. Another major drawback of this indicator is that the histogram estimation would allow two unequal PFs to produce the same I_{MI} if they have similar histograms, as demonstrated in Figure 5.1. To circumvent this problem a novel method for mutual-information based indicator is proposed in the next section.

5.3.3 Processing Pareto Fronts as Images

The main idea is to create a pixel-based image for each PF, and then calculate mutual information among these newly created images. In order for these images to be comparable they should all have the same resolution per dimension (i.e. objective), have the same origin and the same image size.

Given iA: the image of the approximated PF A and iB: the image of the true PF

B then a new indicator is defined as:

$$I_{iMI}(A, B) = U(iA, iB) \tag{5.5}$$

where U is the normalised mutual information defined in Eq. 5.3.

The highest resolution possible for the images is the minimum difference between two adjacent points of all the PF of interest which might be a very high resolution that the generated images would be extremely large to be practical for processing, so the resolution can be reduced in order to generate images of reasonable size. A careful balance must be maintained as low resolution images could lead to a large lose of information affecting the quality of the measure itself. Algorithm 5.1 summarizes the steps to create the PF images.

Algorithm 5.1 Create Pareto Front Images

```
PFs = Scale(PFs)
ratio=CalculateRatio(PFs,Resolution)
for i=1 to N do
  for j=1 to length(PFs_i) do
    pixelIndex=PFs_i(j)*ratio.
    PFImage(pixelIndex)=1;
  end for
end for
```

where PFs is the set of all Pareto fronts to be compared, PFs_i is the PF number i, and N is the total number of PFs. Scale(PFs) ensures all PFs have the same origin by subtracting the minimum objective value among all the PFs from all the PF points. CaculateRatio(PFs) calculates the ratio between the required resolution and the maximum objective value among all PFs which is necessary to project from objective space to the image pixel space (voxel space for the case of three objectives and so on). The ratio actually represents the loss of information when shifting from the objective space to the image space as, naturally, each pixel would average small region of the, usually continuous, objective space. The higher the ratio the smaller the loss of information.

Going back to the problem of unequal PFs with equal histograms. If we used the new measure as an indicator in Fig.5.1, then $I_{iMI}(A, A) = 1$ and $I_{iMI}(A, B) = 0.0018$ which clearly shows that the problem is resolved.

5.3.4 Handling Outliers:

One of the main features of any quality indicator is its ability to handle outliers in the approximated PF. Outliers can cause a bias in the quality measure calculation especially if a distance measure is used. Some measures try to reduce this effect, for example I_{IGD} is calculated by starting from the true PF and then trying to find the closest points in the approximated PF. Although this minimizes the effect of outliers, it still does not provide a fair comparison as it will give the same value for an approximated PF with or without the outliers.

Figure 5.2 demonstrates this effect . In this example we created a hypothetical true PF and approximated PF that contains some outliers. To check the robustness of the quality indicators we calculate the value of the indicator using the approximated PF with and without the outliers, Table 5.1.

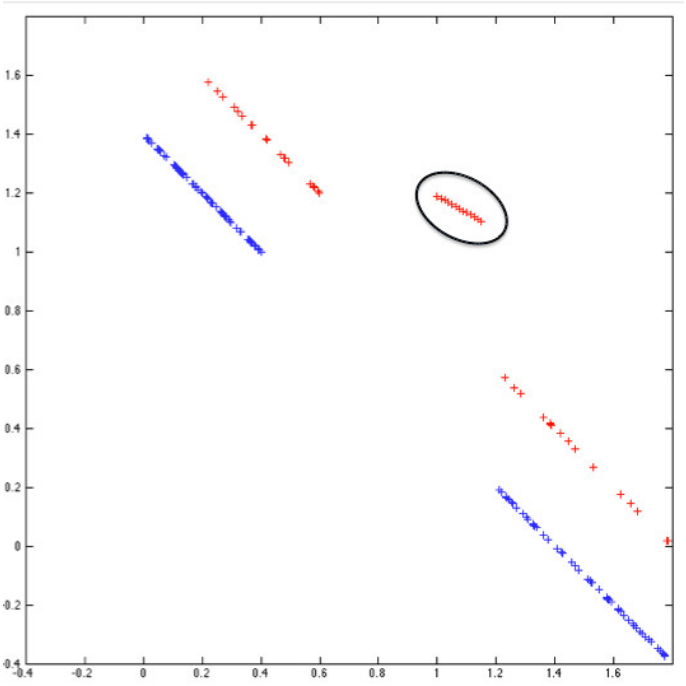


Figure 5.2: The blue dots belong to a hypothetical true PF. The red dots belong to a hypothetical approximated PF. The dots within the circle belong to the approximated PF and are considered outliers.

Table 5.1: Values of the quality indicators with and without outliers in the approximated PF.

Indicator	Without Outliers	With Outliers
I_{IGD}	0.0145	0.0145
I_{hv}	0.2284	0.2332
I_{ϵ}	0.3911	0.3911
I_{iMI}	0.0008	0.0007

5.4 Experiments

It is usually tricky to evaluate the evaluation metric. We tackle this issue indirectly by comparing the results of three other quality measures and discuss how each, including I_{iMI} indicator, handle different cases of evaluation.

5.4.1 Selected Test Problem

To test the newly developed measure 5 standard two-dimensional MOPs are used: ZDT1-ZDT4, and ZDT6 (Deb *et al.* (2005)). The selected test problems cover diverse MOPs with convex, concave, connected and disconnected PFs. These problems were frequently used to verify the performance of several algorithms in the field of multi-objective optimisation (Al Moubayed *et al.* (2011); Alvarez-Benitez *et al.* (2005); Coello Coello *et al.* (2007); Deb *et al.* (2002); Nebro *et al.* (2008); Zhang *et al.* (2009)). They were also used in Fonseca *et al.* (2005b) to compare among several common indicators.

5.4.2 Experimental Setup

The test MOPs are solved using three popular multi-objective optimisation methods:

- NSGAII Deb *et al.* (2002): uses a fast non-dominated sorting approach.
- SPEAII Zitzler *et al.* (2001): uses an external archive of the generated solutions with a clustering method to maintain diversity.
- IBEA Zitzler & Künzli (2004): computes tness values by comparing individuals on the basis of an arbitrary binary quality indicator.

These are the same methods used to test the performance assessment framework in Fonseca *et al.* (2005b). D^2MOPSO was not used here as the assessment measure is general by definition to the output of any multi-objective optimizer so I opted to use only commonly used algorithms.

For each MOP the three methods were run 30 times using a population of 300 individual and lasted for 250 generations.

The quality of the approximated PFs from the five MOPs is measured using four indicators: I_{IGD}, I_{ϵ}, I_{hv}, and I_{iMI}. For I_{iMI} calculation a resolution is set to $1000X1000$.

5.5 Results and Discussion

To demonstrate the results of the approximated PF using the three methods and tested by the four indicators, for each MOP four plots are generated each for each indicator. Each of these plots contains a box plot representation of the values of one indicator applied on the 30 runs for each of the three methods. The line in the centre of a box represents the median with the edges of the box at the 25 and 75% levels and the outer lines the 5 and 95% levels. Figures 5.3 - 5.7 show all the results for the ZDT family.

To further analyse and understand the performance of the different methods a non-parametric statistical significance test is performed, namely Wilcoxon rank sum test. The test compares between the values of the indicator of two methods at a time for each MOP. If a significant difference is found $p < 0.05$ then the difference between the median of the two sets is calculated based on this difference we can tell which of the two methods performed better as stated by the indicator. For example in the case of I_{IGD} the smaller the indicator value the better the approximation of the PF, so if $median(I_{IGD}(A, PF)) - median(I_{IGD}(B, PF)) > 0$ then method B is considered better than method A in terms of I_{IGD}. This is the same for $I_{epsilon}$ and the opposite for I_{iMI} and I_{hv}.

Tables 5.2-5.5 demonstrate these results, where $\triangle$ means that the method in the column is significantly better than the method in the raw. $\triangleleft$ indicates that the method in the raw is significantly better than the method in the column. Because the comparison is symmetric the cell underneath the diagonal are left empty and so on the diagonal as we are not comparing any method with itself.

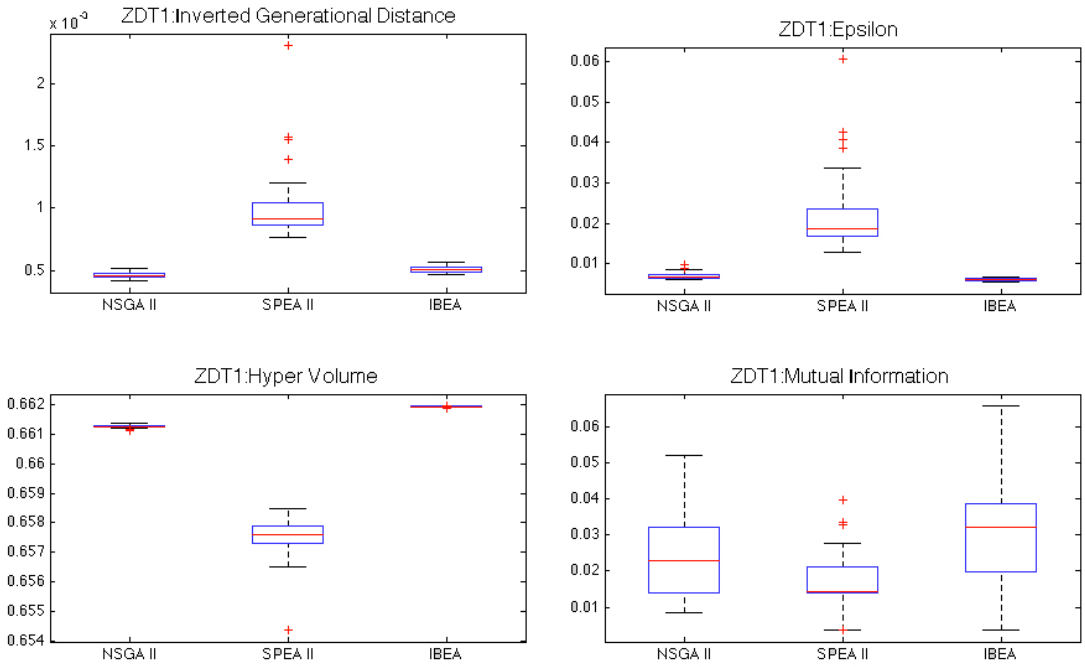


Figure 5.3: Results of ZDT1 using three algorithm: NSGAII,SPEAII, and IBEA compared using four indicators: I_{IGD}, I_ϵ, I_{hv}, and I_{iMI}.

Looking carefully at the figures and tables, the different indicators draw a rather vague, and somehow confusing, picture about the performance of the different methods. Although most of the differences among the different methods are statistically significant they do not always go in the same direction for different indicators. This is a known issue in the quality assessment of multiobjective optimizers via indicators Fonseca *et al.* (2005b). This is interpreted as different indicators provide different information regarding the approximated PF so one can chose the optimizer based on what is more relevant to the application.

The MI indicator seems to be giving a slightly different view from the rest of the indicators. For instance, it is the only indicator to show no significant difference in some cases which actually reflects more what we see from visual inspection of the approximated PFs. It also usually shows less variance among the different runs and fewer outliers (an outlier is represented by a red'+' sign in the box plots) which can be credited to its probabilistic nature.

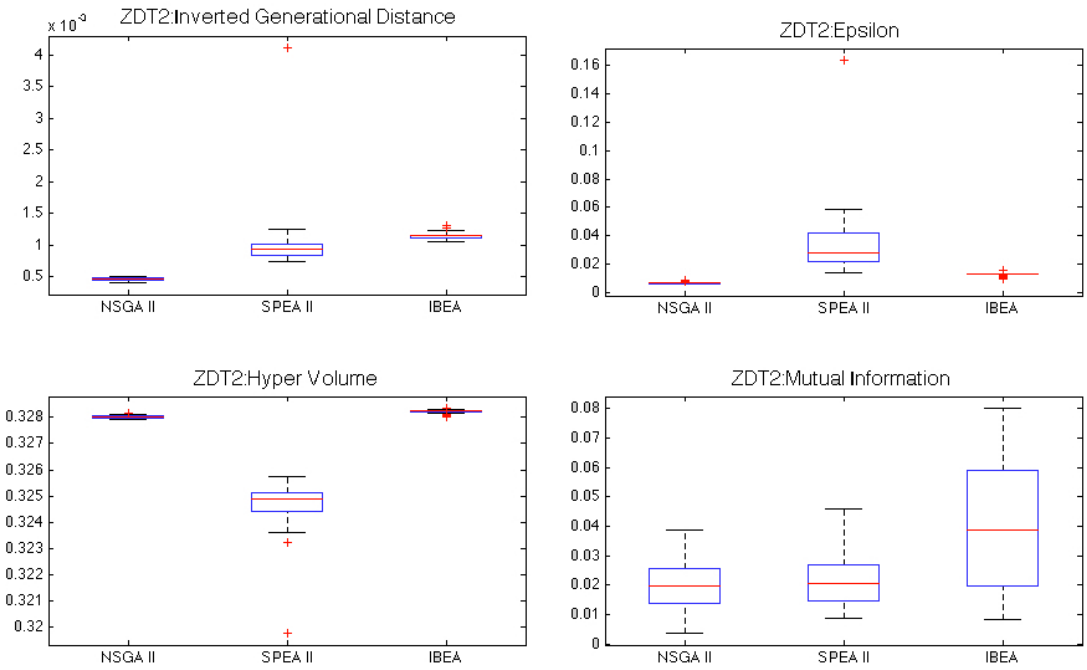


Figure 5.4: Results of ZDT2 using three algorithm: NSGAII,SPEAII, and IBEA compared using four indicators: I_{IGD}, I_ϵ, I_{hv}, and I_{iMI}.

5.6 Conclusion

In this chapter we have presented preliminary results of a novel approach to use mutual information as a quality measure of multionbjective optimisation algorithms. A new indicator, I_{iMI}, is defined that calculates the mutual information between two image representation of the true and approximated PFs. The resulted measure quantifies the reduction of uncertainty about the true Pareto front when the approximated PF is known. In other words it measures the amount of information the approximated PF tell us about the true PF. MI is always positive and equals to 1 when the approximated PF completely match the true PF and 0 when the approximated PF does not hold any information about the true PF.

By definition mutual information is only applied on univariate random variables and hence an independence assumption is imposed among the objectives. If the objectives are dependent, which would be the case in most problems, then the previous definition of mutual information is not accurate. However, by transforming the PF to an image

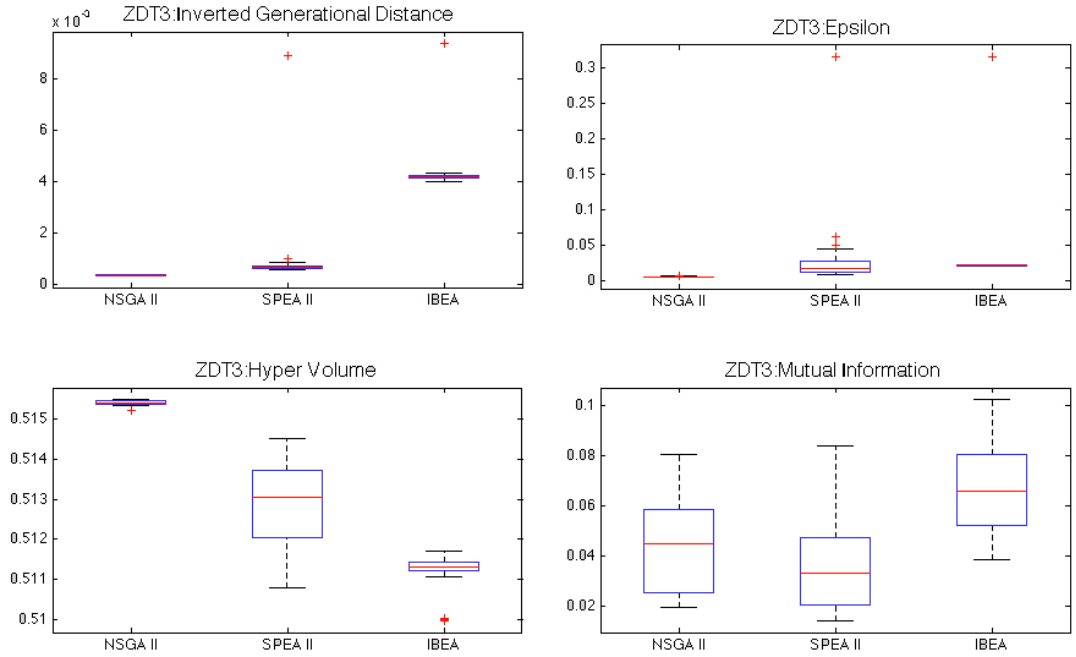


Figure 5.5: Results of ZDT3 using three algorithm: NSGAII,SPEAII, and IBEA compared using four indicators: I_{IGD}, I_{ϵ}, I_{hv}, and I_{iMI}.

all the dependency information are preserved and hence the indicator I_{iMI} can be seen as a multivariant approach.

Because the mutual information function uses estimated probability density functions/ distributions, it is less affected by outliers and can produce a much more robust results than the indicators that use distance functions.

The current results are obtained from only two dimensional data. However, the same algorithms can be applied for higher dimension PFs .

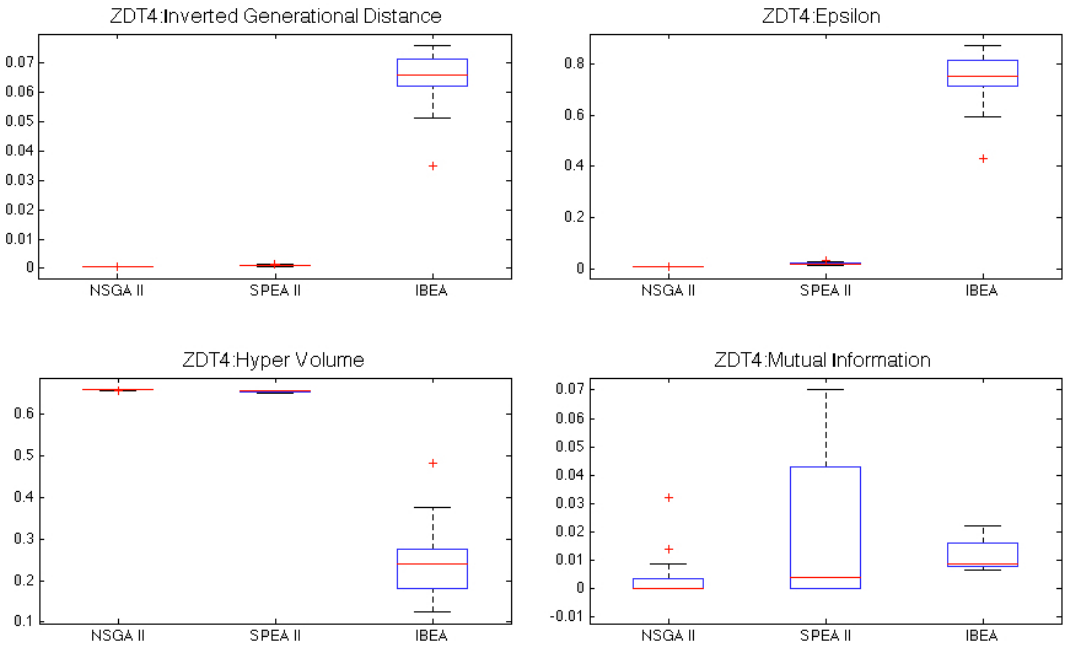


Figure 5.6: Results of ZDT4 using three algorithm: NSGAII,SPEAII, and IBEA compared using four indicators: I_{IGD}, I_ϵ, I_{hv}, and I_{iMI}.

Table 5.2: Statistical significance of the difference between the I_{IGD} values for the different algorithms applied on the 5 problems. A $\triangle$ means the method indicated by the column is significantly better than that indicated by the raw, i.e. $p < 0.05$. $\triangleleft$ mean the raw is significantly better than the column. - means there is no significant difference, i.e. $p > 0.05$. The problems are ordered as follows: ZDT1-ZDT4,ZDT6

	NSGAII	**SPEAII**	**IBEA**
NSGAII		$\triangleleft\ \triangleleft\ \triangleleft\ \triangleleft\ \triangleleft$	$\triangleleft\ \triangleleft\ \triangleleft\ \triangleleft\ \triangleleft$
SPEAII			$\triangle\ \triangleleft\ \triangle\ \triangleleft\ \triangle$
IBEA			

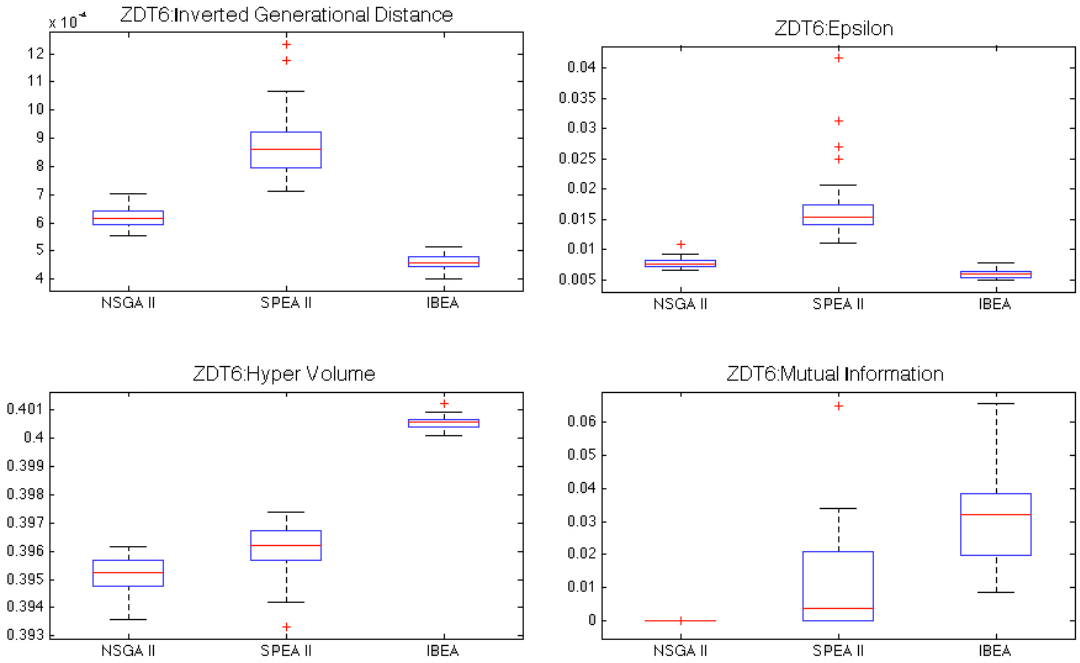


Figure 5.7: Results of ZDT6 using three algorithm: NSGAII,SPEAII, and IBEA compared using four indicators: I_{IGD}, I_ϵ, I_{hv}, and I_{iMI}.

Table 5.3: Statistical significance of the difference between the I_ϵ values for the different algorithms applied on the 5 problems. A $\triangle$ means the method indicated by the column is significantly better than that indicated by the raw, i.e. $p < 0.05$. $\triangleleft$ mean the raw is significantly better than the column. - means there is no significant difference, i.e. $p > 0.05$. The problems are ordered as follows: ZDT1-ZDT4,ZDT6

	NSGAII	**SPEAII**	**IBEA**
NSGAII		$\triangleleft \triangleleft \triangleleft \triangleleft \triangleleft$	$\triangle \triangleleft \triangleleft \triangleleft \triangle$
SPEAII			$\triangle \triangle \triangleleft \triangleleft \triangle$
IBEA			

Table 5.4: Statistical significance of the difference between the I_{hv} values for the different algorithms applied on the 8 problems. A $\triangle$ means the method indicated by the column is significantly better than that indicated by the raw, i.e. $p < 0.05$. $\triangleleft$ mean the raw is significantly better than the column. - means there is no significant difference, i.e. $p > 0.05$. The problems are ordered as follows: ZDT1-ZDT4,ZDT6.

	NSGAII	**SPEAII**	**IBEA**
NSGAII		$\triangleleft$ $\triangleleft$ $\triangleleft$ $\triangle$ $\triangleleft$	$\triangle$ $\triangle$ $\triangleleft$ $\triangleleft$ $\triangle$
SPEAII			$\triangle$ $\triangle$ $\triangleleft$ $\triangleleft$ $\triangle$
IBEA			

Table 5.5: Statistical significance of the difference between the I_{iMI} values for the different algorithms applied on the 8 problems. A $\triangle$ means the method indicated by the column is significantly better than that indicated by the raw, i.e. $p < 0.05$. $\triangleleft$ mean the raw is significantly better than the column. - means there is no significant difference, i.e. $p > 0.05$. The problems are ordered as follows: ZDT1-ZDT4,ZDT6.

	NSGAII	**SPEAII**	**IBEA**
NSGAII		$-$ $\triangle$ $-$ $-$ $-$	$-$ $\triangle$ $\triangle\triangle$ $\triangle$
SPEAII			$\triangle$ $\triangle$ $\triangle$ $-$ $\triangle$
IBEA			

Chapter 6

Conclusions

" I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right. "

Albert Einstein

The thesis provides advances to several areas of research within the field of multi-objective evolutionary optimisation in general and multi-objective particle swarm optimisation in particular. The thesis presents novel solutions to the combination of multi-objective PSO and decomposition. First decomposition is employed to define the neighbourhoods in MOPSO which resulted in the algorithm SDMOPSO. D^2MOPSO on the other hand applied decomposition for leader selection and combined that with a novel archiving technique which maps the solution and objective spaces using crowding distances in both spaces. A general framework for is also introduce which maps the solution and objective spaces using clustering. The methods are validated on several standard test suits and two real-life problems: Channel selection for Brain-Computer Interfaces and Treatment doses for Cancer Chemotherapy. Finally the thesis provide a new method for the assessment of multi-objective optimisers. Following is a detailed description of the contribution of this thesis and a perspective of the possible future work.

6.1 Summary of Contributions

The thesis contributes to the field of evolutionary multi-objective optimisation in general and multi-objective Particle swarm optimisation in particular. The main contributions can be summarized as follows:

- Multi-objective particle swarm optimisation based on decomposition: In this thesis solutions are presented to incorporate the concept of decomposition into MOPSO. First the neighbourhood relations among particles is changed in MOEA/D to accommodate PSO which in turns modifies the information flow during optimisation. This leads to better exploitation of local information among the particles. The results of SDMOPSO is highly competitive to the state of the art in both MOPSO and evolutionary multi-objective algorithms.

  The next challenge was to address the issue of archiving the solutions found during the optimisation process. A novel approach was first proposed which maps the solutions and their corresponding objective values. This is done by clustering both spaces and then find a mapping between the clusters in both spaces which is developed as a general framework for archiving. Although, this approach proved

effective it is too computationally expensive. The framework, however, is general enough for the clustering to be substituted with another measure and to be incorporated with almost any evolutionary multi-objective algorithm.

Finally, a new optimisation algorithm (D^2MOPSO) is proposed, which also incorporates decomposition as a measure to select among leaders and it also uses crowding with ϵ-dominance as a metric to replace clustering in the proposed archive. These methods are tested on standard optimisation problems and are extensively tested. The results on test suits are very competitive and show very promising results not in terms of approximated PFs alone but also in terms of computational complexity and performance. D^2MOPSO is currently considered the state of the art in MOPSO and has outperformed all the newly developed methods in the literature that combines decomposition and MOPSO.

- Real-world applications for the developed methods: The proposed algorithms were tested on two real-life problems: Channel selection for Brain-Computer Interfaces (BCIs) and the finding of effective cancer chemotherapeutic treatments. The methods are tested against the-state-of-the-art in the respective fields. The results obtained from both problems show significant improvement when using SDMOPSO and D^2MOPSO. These results were confirmed and approved by experts in the respective fields.

  In the channel selection problem for BCI I also proposed a novel problem representation which represents the problem in a continuous, rather than discrete, 2D space which suits better the nature of PSO. The results on BCI show significantly better results not in terms of accuracy alone but also the interpretability of the results and the distribution of the channels in relation to what is known from neuroscience studies.

  The solutions to chemotherapy optimisation problem of similar quality to the state of the art, but in larger quantities. One practical implication of this outcome is the possibility of providing a more advanced decision support to the oncologists involved in seeking the most suitable treatment strategies.

- Mutual Information based assessment of multi-objective optimisation methods: This is a novel assessment method for multi-objective optimisers. For the first

time the assessment method is based on estimating the probabilistic distribution of the solutions with a definition of an evaluation metric. A method is proposed which sees the Pareto front as a pixelated image which are then compared using a defined as the mutual information between the images representing Pareto fronts. The new metric is superior to other measure as it is robust to outliers and is not based on single property of the data as it looks at the distribution of the whole approximated PFs.

The work resulted in one journal paper and 8 conference papers (including 4 book chapters) in highly ranked conference in the field.

6.2 Future Work:

- The leaders selection is an extremely important step for all MOPSO methods. However, more work is still needed to understand the actual effect and study the movement of the particles accordingly.

- Storing all none-dominated particle is not only unpractical but also is not feasible for most applications. Maintaining the leaders archive under a size restriction is very important. More studies are needed to show how this can effect the optimisation process and the convergence speed for the PF formed.

- Hybrid algorithms: the thesis showed that combining techniques from, usually, independent approaches can be very beneficial. This is becoming more trendy in the field but more work is necessary to take advantage of advances in several fields of computational intelligence in general and evolutionary optimisation in particular. Hybrid MOPSOs in particular have proven their effectiveness in the field, so the combination of several MOPSOs seems like a potentially interesting area to investigate.

- The introduction of machine learning techniques in quality measure, archiving, and leaders selection have very promising future and needs to be investigated more. A potential area of application can be on the very expensive problem, i.e. when the evaluation function is very expensive or time consuming to conduct.

Machine learning approaches can help prioritising the evaluations and hence reducing the need for unnecessary evaluations.

- Methods validation: The proposed methods are validated on two real-life problems in addition to the standard test problems. However, testing on more real-life problems will help understand the performance of the methods better and hence potentially enhance the methods. This approach of evaluating new algorithms on real-life problems as well as test suits is a very important step froward for the field. As we have seen in this thesis different applications will have different requirements which will as a result affect the design of the optimisation algorithm itself to accommodate for these requirements.

- The developed archiving technique is a general one that can be applied for any evolutionary multi-objective optimiser. It is of interest to study the impact of the new archive on other state-of-the-art evolutionary algorithms other than MOPSO. It is also of a great interest to provide analytical understanding of the effect of this archiving technique on the optimisation process.

- The mutual information assessment measure must be studied in more detail especially for problems with more than two objectives. Although in theory it should perform similarly to what been presented here, experimental tests might be necessary to confirm this. Other information-based and multivariate statistical tests can be very interesting in this domain. The future of these measure should be more dependent on the distribution of the solutions and not only on measures that take one, or few, aspect(s) of the data.

6.3 In Conclusion ...

The thesis offers comprehensive solutions to several aspects of multi-objective optimisation using decomposition with the framework of PSO. The thesis proposes novel archiving techniques with two MOPSO algorithms (SDMOPSO and D^2MOPSO) which are demonstrated to outperform the state of the art on standard test suits and real-life problems. The channel selection problem for Brain-Computer Interfaces and the regulation of Cancer chemotherapy treatments are both solved using the newly developed

methods which also resulted in advances in the problem representation in either fields. Finally a novel quality measure was developed based on mutual information which overcomes shortcoming and limitations of the traditional quality measures by using the distribution of the solution to define the metric. The work resulted in one journal article and eight conference papers published in the highest ranking publication outlets in the field.

References

A. Petrovski, B.S. & McCall, J. (2004). *Optimising Cancer Chemotherapy Using Particle Swarm Optimisation and Genetic Algorithms*, vol. 3242 of *Lecture Notes in Computer Science*. Springer. 107, 126

Al Moubayed, N., Petrovski, A. & McCall, J. (2010). A novel smart particle swarm optimisation using decomposition. In *PPSN XI*, Lecture Notes in Computer Science, Springer. 60, 79, 82

Al Moubayed, N., Petrovski, A. & McCall, J. (2011). Clustering-based leaders' selection in multi-objective particle swarm optimisation. In *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, vol. 6936 of *Lecture Notes in Computer Science*, 100–107, Springer Berlin / Heidelberg. 82, 151

Allmendinger, R. & Knowles, J. (2010). Evolutionary optimization on problems subject to changes of variables. In *Parallel Problem Solving from Nature - PPSN XI, Part II*, vol. 6239 of *Lecture Notes in Computer Science*, 151–160, Springer Berlin / Heidelberg. 126

Alvarez-Benitez, J., Everson, R. & Fieldsend, J. (2005). A mopso algorithm based exclusively on pareto dominance concepts. In *Evolutionary Multi-Criterion Optimization*, 459–473, Springer. 32, 151

Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *Evolutionary Programming VII*, 601–610, Springer. 12

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA. 9

Back, T. (2002). Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In *IEEE World Congress on Computational Intelligence*, vol. 1, 57 – 62, Orlando, FL , USA. 60

Baker, J.E. (1985). Adaptive selection methods for genetic algorithms. In *Proc. of the International Conference on Genetic Algorithms and Their Applications*, 101–111, Pittsburgh, PA, USA. 60, 129

Baltar, A. & Fontane, D. (2006). A generalized multiobjective particle swarm optimization solver for spreadsheet models: application to water quality. In *the twenty sixth annual American geophysical union hydrology days*. 107

Bartz-Beielstein, T., Limbourg, P., Mehnen, J., Schmitt, K., Parsopoulos, K. & Vrahatis, M. (2003). Particle swarm optimizers for pareto optimization with enhanced archiving techniques. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 3, 1780–1787, IEEE. 39

Basu, S., Banerjee, A. & Mooney, R. (2002). Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. 63

Baumgartner, U., Magele, C. & Renhart, W. (2004). Pareto optimality and particle swarm optimization. *Magnetics, IEEE Transactions on*, **40**, 1172–1175. 37

Ben-Jacob, E. (2003). Bacterial self–organization: co–enhancement of complexification and adaptability in a dynamic environment. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, **361**, 1283–1312. 2

Birbaumer, N., Hinterberger, T., Kubler, A. & Neumann, N. (2003). The thought-translation device (ttd): Neurobehavioral mechanisms and clinical outcome. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, **11**, 120–123. 108

Bishop, C. *et al.* (2006). *Pattern recognition and machine learning*, vol. 4. springer New York. 32, 62

Blackwell, T. (2007). Particle swarm optimization in dynamic environments. *Evolutionary computation in dynamic and uncertain environments*, 29–49. 18

Blankertz, B., Dornhege, G., Krauledat, M., Mller, K. & Curio, G. (2007). The non-invasive berlin brain-computer interface: fast acquisition of effective performance in untrained subjects. *NeuroImage*, 539–550. 111

Cassidy, J. & McLeod, H. (1995). Is it possible to design a logical development plan for an anti-cancer drug. *Pharmaceutical Medicine*, **9**, 95–103. 129

Ciarlet, P. (1989). *Introduction to numerical linear algebra and optimisation*. Cambridge University Press. 9

Clerc, M. & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, **6**, 58–73. 20

Coello Coello, C. & Lechuga, M. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 2, 1051–1056, IEEE. 38

Coello Coello, C., Pulido, G. & Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, **8**, 256–279. 38, 59, 60

Coello Coello, C., Lamont, G. & Van Veldhuizen, D. (2007). *Evolutionary algorithms for solving multi-objective problems*, vol. 5. Springer. 8, 9, 20, 21, 23, 27, 30, 37, 46, 55, 68, 75, 82, 151

Cohn, D., Caruana, R. & Mccallum, A. (2003). Semi-supervised clustering with user feedback. Tech. rep., Cornell University. 63

Cohon, J. (2004). *Multiobjective programming and planning*, vol. 140. Dover Publications. 23

Das, I. & Dennis, J. (1998). Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, **8**, 631–657. 24

Deb, K. & Agrawal, R.B. (1994). Simulated binary crossover for continuous search space. *Complex Systems*, **1**, 115–148. 83

Deb, K. & Goldberg, D. (1989). An investigation of niche and species formation in genetic function optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, 42–50, Morgan Kaufmann Publishers Inc. 32

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, **6**, 182–197. 25, 33, 60, 68, 82, 83, 132, 151

Deb, K., Thiele, L., Laumanns, M. & Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization*, 105–145. 83, 151

Desmar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**, 1–30. 56

Dornhege, G., Blankertz, B., Krauledat, M. & et al (2006). Combined optimization of spatial and temporal filters for improving brain-computer interfacing. *IEEE Transactions on Biomedical Engineering*, 2274–2281. 108

Durillo, J., Nebro, A., Luna, F., Dorronsoro, B. & Alba, E. (2006). jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos. 55, 83

Dyson, M., Sepulveda, F., Gan, J. & S.J.Roberts (2009). Sequential classification of mental tasks vs. idle. In *In International IEEE EMBS Conf. on Neural Engineering*, 351–354. 109

Eberhart, R. & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. In *Evolutionary Programming VII*, 611–616, Springer. 12

Eberhart, R. & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, 84–88, IEEE. 14

Eberhart, R., Simpson, P. & Dobbins, R.W. (1996). *Computational intelligence PC tools*. Boston: Academic Press. 12

Eberhart, R., Shi, Y. & Kennedy, J. (2001). *Swarm intelligence*. Morgan Kaufmann. 14

El-Ghazali, T. (2009). *Metaheuristics: from design to implementation*. Jonh Wiley and Sons Inc., Chichester. 9, 10, 11, 56, 74, 75, 133

Engelbrecht, A. (2005). *Fundamentals of computational swarm intelligence*, vol. 1. Wiley London. 18

Engelbrecht, A. (2007). *Computational intelligence: an introduction*. wiley. 12, 15

Ester, M., Kriegel, H.P., Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings of 2nd International Conference on Knowledge Discovery and*, 226–231. 63

Feng, S.S. & Chien, S. (2003). Chemotherapeutic engineering: Application and further development of chemical engineering principles for chemotherapy of cancer and other diseases. *Chemical Engineering Science*, **1**, 4087–4114. 126

Fieldsend, J., Uk, E. & Singh, S. (2002). A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. 38

Fleischer, M. (2003). *The Measure of Pareto Optima Applications to Multi-objective Metaheuristics*, vol. 2632 of *Lecture Notes in Computer Science*. Springer. 59

Fonseca, C., da Fonseca, V. & Paquete, L. (2005a). Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In *Evolutionary Multi-Criterion Optimization*, 250–264, Springer. 45, 144

Fonseca, C., Knowles, J., Thiele, L. & Zitzler, E. (2005b). A tutorial on the performance assessment of stochastic multiobjective optimizers. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, vol. 216. 45, 46, 47, 143, 145, 151, 152, 153

Fonseca, C.M. & Fleming, P.J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization (s. forrest, ed.). In *Proceedings of the fifth international conference on genetic algorithms*, vol. 1, 416–423, Morgan Kaufmann, San Mateo, CA. 31

Fonseca, C.M. & Fleming, P.J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, **28**, 26–37. 83

Frackowiak, R.S. (2004). *Human brain function*. Academic Press. 147

Gan, J. (2006). Feature dimensionality reduction by manifold learning in brain computer interfaces. In *3rd International Workshop on Brain Computer Interfaces*, 28–29, Graz, Austria. 108

Gan, J.Q., Hasan, B.A.S. & Tsui, C. (2011). A hybrid approach to feature subset selection for brain-computer interface design. *Intelligent Data Engineering and Automated Learning-IDEAL 2011*, 279–286. 109, 120

Gentle, J. (2003). *Random number generation and Monte Carlo methods*. Springer. 11

Gershenfeld, N. (1998). *The nature of mathematical modeling*. Cambridge university press. 8

Ghosh, A. (2004). Evolutionary algorithms for multi-criterion optimization: A survey. *International Journal of Computing and Information Sciences*, **2**, 38–57. 60

Goldberg, D. & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 41–49, L. Erlbaum Associates Inc. 32

Grefenstette, J.J. & Baker, J.E. (1989). How genetic algorithms work: A critical look at implicit parallelism. In *International Conference on Genetic Algorithms*, 20–27. 60

Grira, N., Crucianu, M. & Boujemaa, N. (2005). Unsupervised and semi-supervised clustering: a brief survey. 62

Grunert da Fonseca, V., Fonseca, C. & Hall, A. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. In *Evolutionary Multi-Criterion Optimization*, 213–225, Springer. 45, 145

Harrold, J. & Parker, R. (2009). Clinically relevant cancer chemotherapy dose scheduling via mixed-integer optimization. *Computers and Chemical Engineering*, **1**, 2042–54. 127

Hasan, B.A.S. & Gan, J. (2011). Temporal modeling of eeg during self-paced hand movement and its application in onset detection. *Journal of Neural Engineering*, **8**, 056015. 108

Hasan, B.A.S., Gan, J. & Zhang, Q. (2010). Multi-objective evolutionary methods for channel selection in brain-computer interfaces: some preliminary experimental results. In *WCCI*, 3339–3344, IEEE, Barcelona, Spain. 107, 109, 110, 111, 139, 140

Hassan, R., Cohanim, B., de Weck, O. & Venter, G. (2005). A comparison of particle swarm optimization and the genetic algorithm. In *Structural Dynamics and Materials*, Texas, USA. 107

Ho, S., Yang, S., Ni, G., Lo, E. & Wong, H. (2005). A particle swarm optimization-based method for multiobjective design optimizations. *Magnetics, IEEE Transactions on*, **41**, 1756–1759. 40

Holland, J.H. (1992). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA. 60

Huband, S., Barone, L., While, L. & Hingston, P. (2005). A scalable multi-objective test problem toolkit. In *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, 280–295, Springer. 83

Huband, S., Hingston, P., Barone, L. & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, **10**, 477–506. 48

Jaishia, B. & Ren, W. (2007). Finite element model updating based on eigenvalue and strain. *Mechanical Systems and Signal Processing*, **21**, 2295–2317. 107

Janson, S. & Merkle, D. (2005). A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking. *Hybrid Metaheuristics*, 902–902. 41

Jin, J., Wang, X. & Zhang, J. (2008). Optimal selection of eeg electrodes via dpso algorithm. In *In Proc. of the 7th World Congress on Intelligent Control and Automation*. 109

Jun, L. & Meichun, L. (2008). Common spatial pattern and particle swarm optimization for channel selection in bci. In *In Proc. 3rd International Conference on Innovative Computing Information and Control. ICICIC 08*, 457–457. 109

Kadirkamanathan, V., Selvarajah, K. & Fleming, P. (2006). Stability analysis of the particle dynamics in particle swarm optimizer. *Evolutionary Computation, IEEE Transactions on*, **10**, 245–255. 18

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, 1942–1948, IEEE. 2, 12, 13, 36

Kennedy, J. & Eberhart, R. (1997a). A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, 4104–4108, IEEE. 12

Kennedy, J. & Eberhart, R.C. (1997b). A discrete binary version of the particle swarm algorithm. In *International Conference Systems, Man and Cybernetics*, 4104–4108, IEEE. 110

Kilmesch, W. (1999). Event–related band power changes and memory performance. *Handbook of Electroencephalography and Clinical Neurophysiology Revised Series*, **6**, 161–178. 121

King, I. (2002). A simple introduction to dynamic programming in macroeconomic models. 9

Knowles, J. & Corne, D. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary computation*, **8**, 149–172. 31

Knowles, J. & Corne, D. (2002). On metrics for comparing nondominated sets. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, 711–716, IEEE. 45, 56, 144

Kurpati, A., Azarm, S. & Wu, J. (2002). Constraint handling improvements for multiobjective genetic algorithms. *Structural and Multidisciplinary Optimization*, **23**, 204–213. 83

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature*, 193–197. 83

Laumanns, M., Thiele, L., Deb, K. & Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, **10**, 263–282. 32

Li, H. & Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-ii. *IEEE Trans. on Evolutionary Computation*, **13**, 284 –302. 77, 82, 83, 131, 132

Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary ComputationGECCO 2003*, 198–198, Springer. 39

Li, Y., Yao, D., Yao, J. & Chen, W. (205). A particle swarm optimization algorithm for beam angle selection in intensity-modulated radiotherapy planning. *Physics in Medicine and Biology*, **1**, 3491–3514. 107

Liang, Y., Leung, K.S. & Mok, T. (2008). Evolutionary drug scheduling models with different toxicity metabolism in cancer chemotherapy. *Applied Soft Computing*, **1**, 140–49. 107, 141

Maaranen, H., Miettinen, K. & Penttinen, A. (2007). On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, **37**, 405–436. 10

Mahfouf, M., Chen, M. & Linkens, D. (2004). Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels. In *Parallel problem solving from nature-ppsn viii*, 762–771, Springer. 41

Martin, R. & Teo, K.L. (1994). *Optimal Control of Drug Administration in Cancer Chemotherapy*. World Scientific Pub Co Inc. 128, 129, 130

Martínez, S. & Coello Coello, C. (2011). A multi-objective particle swarm optimizer based on decomposition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, ACM. 42, 79, 83

Mattson, C., Mullur, A. & Messac, A. (2004). Smart pareto filter: Obtaining a minimal representation of multiobjective design space. *Engineering Optimization*, **36**, 721–740. 24

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA. 60

Miettinen, K. (1999). *Nonlinear multiobjective optimization*, vol. 12. Springer. 23

Moore, J. & Chapman, R. (1999). Application of particle swarm to multiobjective optimization. *Department of Computer Science and Software Engineering, Auburn University*. 32

Morse, J. (1980). Reducing the size of the nondominated set: Pruning by clustering. *Computers & Operations Research*, **7**, 55–66. 32

Mostaghim, S. & Teich, J. (2003a). The role of ϵ-dominance in multi objective particle swarm optimization methods. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 3, 1764–1771, IEEE. 33

Mostaghim, S. & Teich, J. (2003b). Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, 26–33, IEEE. 38

Mostaghim, S. & Teich, J. (2004). Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, 1404–1411, IEEE. 39

Moubayed, N.A., Hasan, B.A.S., Gan, J., Petrovski, A. & McCall, J. (2010). Binary-SDMOPSO and its application in channel selection for brain-computer interfaces. In *Computational Intelligence (UKCI), 2010 UK Workshop on*, 1–6, IEEE. 140

Nasir, M., Mondal, A., Sengupta, S., Das, S. & Abraham, A. (2011). An improved multiobjective evolutionary algorithm based on decomposition with fuzzy dominance. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 765–772, IEEE. 82

Nebro, A.J., Luna, F., Alba, E., B, A.B. & Dorronsoro, B. (2008). Abyss: Adapting scatter search for multiobjective optimization. *IEEE Trans. on Evolutionary Computation*, **12**, 439–457. 83, 151

Neuper, C. & Pfurtscheller, G. (1999). Motor imagery and ERD. *In: Pfurtscheller G, Lopes da Silva FH. Handbook of Electroencephalography and Clinical Neurophysiology- Event-related Desynchronization*. 108

Okabe, T., Jin, Y. & Sendhoff, B. (2003). A critical survey of performance indices for multi-objective optimisation. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 2, 878–885, IEEE. 45, 144

Osyczka, A. & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural and Multidisciplinary Optimization*, **10**, 94–99. 83

Ozcan, E. & Mohan, C. (1998). Analysis of a simple particle swarm optimization system. *Intelligent Engineering Systems Through Artificial Neural Networks*, **8**, 253–258. 18

Parsopoulos, K. & Vrahatis, M. (2002). Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, 603–607, ACM. 37

Parsopoulos, K. & Vrahatis, M. (2008). Multi-objective particles swarm optimization approaches. *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, 20–42. 39

Peng, H., Long, F. & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **27**, 1226–1238. 147

Peng, W. & Zhang, Q. (2008). A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In *IEEE International Conference on Granular Computing*, Hangzhou. 42, 56, 79

Petrovski, A. (1999). *An Application of Genetic Algorithms to Chemotherapy Treatment*. Ph.D. thesis, Robert Gordon University, Aberdeen, United Kingdom. 107, 129

Petrovski, A. & McCall, J. (1999). Decision support system for cancer chemotherapy using genetic algorithms. In *International Conference on Computational Intelligence for Modelling, Control and Automation*, vol. 1, IOS Press. 107, 126

Petrovski, A. & McCall, J. (2001). Multi-objective optimisation of cancer chemotherapy using evolutionary algorithms. In *The First International Conference on Evolutionary Multi-Criterion Optimisation*, Zurich, Switzerland. 126, 127

Pfurtscheller, G. & da Silva, F.L. (1999). Event-related EEG/MEG synchronization and desynchronization: basic principles. *Clinical Neurophysiology*, **110**, 1842–1857. 108

Pfurtscheller, G. & Neuper, C. (2001). Motor imagery and direct brain–computer communication. *Proceedings of the IEEE*, **89**, 1123–1134. 120

Pfurtscheller, G., Neuper, C. & Birbaumer, N. (2005). *Human brain–computer interface (BCI). In A. Riehle and E. Vaadia (Eds.), Motor Cortex in Voluntary Movements. A Distributed System For Distributed Functions*. Boca Raton, FL: CRC Press. 108

Poli, R., Kennedy, J. & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, **1**, 33–57. 13, 14, 15, 18

Pulido, G. (2005). *On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization*. PhD thesis, Computer Science Section, Department of Electrical Engineering, CINVESTAV-IPN, Mexico. 29

Pulido, G. & Coello Coello, C. (2004). Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In *Genetic and Evolutionary Computation–GECCO 2004*, 225–237, Springer. 38

Raghuwanshi, M. & Kakde, O. (2004). Survey on multiobjective evolutionary and real coded genetic algorithms. In *In Proceedings of the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, vol. 11, 150–161, Cairns, Australia. 61

Raquel, C. & Naval Jr, P. (2005). An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, 257–264, ACM. 30

Ray, T. & Liew, K. (2002). A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, **34**, 141–153. 38

Reyes-Sierra, M. & Coello, C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, **2**, 287–308. 13, 21, 28, 35, 43, 59, 74, 107

Rothlauf, F. (2011). *Design of Modern Heuristics: Principles and Application*. Springer. 8, 9

Rudolph, G. (1998). On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 511–516, IEEE. 30

Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international Conference on Genetic Algorithms*, 93–100, L. Erlbaum Associates Inc. 21

Scott, D.W. (1979). On optimal and data-based histograms. *Biometrika*, **66**, 605–610. 146

Shen, Q., Jiang, J.H., Jiao, C., Shen, G. & R.Yu (2004). Modified particle swarm optimization algorithm for variable selection in mlr and pls modeling: Qsar studies of antagonism of angiotensin ii antagonists. *European Journal of Pharmaceutical Sciences*, **22**, 145–152. 110

Shi, Y. & Eberhart, R. (1998). Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, 591–600, Springer. 36

Shi, Y. & Eberhart, R. (1999). Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, IEEE. 36

Sierra, M. & Coello Coello, C. (2005). Improving pso-based multi-objective optimization using crowding, mutation and-dominance. In *Evolutionary Multi-Criterion Optimization*, 505–519, Springer. 40, 52, 55, 59, 60, 74, 79, 82, 83, 112, 133

Srinivas, N. & Deb, K. (1994a). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, **2**, 221–248. 25

Srinivas, N. & Deb, K. (1994b). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, **2**, 221–248. 83

Stacey, A., Jancic, M. & Grundy, I. (2003). Particle swarm optimization with mutation. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 2, 1425–1430, IEEE. 36, 37

Tan, K.C., Khor, E.F., Cai, J., Heng, C.M. & Lee, T.H. (2002). Automating the drug scheduling of cancer chemotherapy via evolutionary computation. *Artificial Intelligence in Medicine*, **25**, 169–185. 126

Tanaka, M., Watanabe, H., Furukawa, Y. & Tanino, T. (1995). GA-based decision support system for multicriteria optimization. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 1556–1561, IEEE. 83

Tsui, C., Gan, J. & Roberts, S. (2009). A self-paced brain–computer interface for controlling a robot simulator: An online event labelling paradigm and an extended Kalman filter based algorithm for online training. *Medical and Biological Engineering and Computing*, **47**, 257–265. 108

Tucker, A. (1957). Symposium on modern techniques for extremum problemslinear and nonlinear programming. *Operations Research*, **5**, 244–257. 23

Van Den Bergh, F. *et al.* (2002). An analysis of particle swarm optimizers. 18, 20

Van Veldhuizen, D. (1999). Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Tech. rep., DTIC Document. 46

Van Veldhuizen, D.A. & Lamont, G.B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. *Air Force Inst. Technol., Dayton, OH, Tech. Rep. TR-98-03*. 46

Villalobos-Arias, M., Pulido, G. & Coello, C. (2005). A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, 22–29, IEEE. 41

Villasana, M., Ochoa, G. & Aguilar, S. (2010). Modeling and optimization of combined cytostatic and cytotoxic cancer chemotherapy. *Artificial Intelligence in Medicine*, **50**, 163–173. 141

Vlennet, R., Fonteix, C. & Marc, I. (1996). Multicriteria optimization using a genetic algorithm for determining a Pareto set. *International Journal of Systems Science*, **27**, 255–260. 83

Wang, Y., Dang, C., Li, H., Han, L. & Wei, J. (2009). A clustering multi-objective evolutionary algorithm based on orthogonal and uniform design. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, 2927–2933, IEEE Press. 61

Wang, Z., Durst, G., Eberhart, R., Boyd, D. & Miled, Z.B. (2004). Particle swarm optimization and neural network application for qsar. *Parallel and Distributed Processing Symposium, International*, **10**, 194. 107

Wheeler, W. (1912). The ant colony as an organism. *Journal of Morphology*, **22**, 307. 2

Wheldon, T. (1988). *Mathematical Models in Cancer Research*. Taylor & Francis. 127

Witten, I.H. & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann. 147

Wolpaw, J., Birbaumer, N., McFarland, D., Pfurtscheller, G. & Vaughan, T. (2002). Brain-computer interfaces for communication and control. *Clinical neurophysiology*, **113**, 767–791. 108

Zhang, Q. & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, **11**, 712–731. 25, 42, 51, 74

Zhang, Q., Liu, W. & Li, H. (2009). The performance of a new version of moea/d on cec09 unconstrained mop test instances. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, 203–208, IEEE. 51, 54, 55, 83, 151

Zhang, Q., Liu, W., Tsang, E. & Virginas, B. (2010). Expensive multiobjective optimization by moea/d with gaussian process model. *Evolutionary Computation, IEEE Transactions on*, **14**, 456–474. 27

Zheng, Y., Ma, L., Zhang, L. & Qian, J. (2003). On the convergence analysis and parameter selection in particle swarm optimization. In *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 3, 1802–1807, IEEE. 14

Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N. & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*. 74

Zitzler, E. & Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN VIII*, 832–842, Springer. 151

Zitzler, E. & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms. a comparative case study. In *Parallel problem solving from naturePPSN V*, Lecture Notes in Computer Science, 292–301, Springer. 47

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, **3**, 257–271. 32, 60

Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L. & Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. 151

Zitzler, E., Laumanns, M. & Bleuler, S. (2003a). A tutorial on evolutionary multiobjective optimization. In *In Metaheuristics for Multiobjective Optimisation*, 3–38, Springer-Verlag. 76

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. & da Fonseca, V. (2003b). Performance assessment of multiobjective optimizers: An analysis and review. *Evolutionary Computation, IEEE Transactions on*, **7**, 117–132. 45, 68, 144

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. & da Fonseca, V.G. (2003c). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. on Evolutionary Computation*, **7**, 117–132. 47