Multi-Objective Evolutionary Algorithms

Data Structures, Convergence, and Diversity

Zur Erlangung des akademischen Grades
DOKTOR-INGENIEUR (Dr.-Ing.)
der Fakultät Elektrotechnik, Informatik und Mathematik
der Universität Paderborn
genehmigte Dissertation
von
M.Sc. Sanaz Mostaghim
aus
Teheran, Iran

Referent:	Prof. Dr.-Ing. Jürgen Teich
Erster Korreferent:	Prof. Dr. Michael Dellnitz
Zweiter Korreferent:	Prof. Dr.-Ing. Klaus Meerkötter
Tag der mündlichen Prüfung:	21.09.2004

Paderborn, den 22. November 2004

Diss. 14/205

In Loving Memory of
My Mother

Abstract

Many real-world optimization problems consist of several conflicting objectives, the solutions of which is a set of trade-offs called the Pareto-optimal set. During the last decade, Evolutionary Algorithms (EAs) have been utilized to find an approximation of the Pareto-optimal set. However, the approximation set must possess solutions with high convergence towards the Pareto-optimal set and hold a good diversity in order to demonstrate a good approximation.

The subject of this thesis is to improve the existing Multi-Objective Evolutionary Algorithms (MOEAs) and to develop new techniques in order to achieve approximated sets with high convergence and diversity in low computational time.

Reducing the computational time has been attained by incorporating various data structures and archiving techniques in the storage of the approximated solutions. A desirable convergence of solutions has been accomplished by applying a controllable search strategy to MOEAs in a hybrid MOEA.

Since 1995, the search strategy of EAs has been modified by a new optimization technique called Particle Swarm Optimization (PSO), inspired by the simulation of social behavior of bird flocking. In this thesis, a novel approach in Multi-Objective Particle Swarm Optimization (MOPSO) has been developed which leads to a better convergence and diversity of solutions than MOEAs. This has been demonstrated by several test problems and two real-world applications in antenna design and molecular force field parameterization in computational chemistry. Furthermore, a new quantitative metric for measuring the diversity of the approximated set has been developed and applied to the obtained solutions.

Zusammenfassung

Viele alltägliche Optimierungsprobleme bestehen aus gegensätzlichen Zielen, die gleichzeitig erfüllt werden müssen. Die Lösungen, die Kompromisse zwischen diesen Optimierungszielen erreichen, bilden eine so genannte Pareto-optimale Menge. In den vergangenen Jahren sind evolutionäre Algorithmen (engl. Evolutionary Algorithm (EA)) eingesetzt worden, um Annährungen zu diesen Pareto-Mengen zu finden, die auf der einen Seite so gut wie möglich in Richtung der Elemente dieser Menge konvergieren, und auf der anderen Seite einen großen Teil dieser Menge überdecken (Diversität).

Diese Arbeit beschäftigt sich mit der Verbesserung vorhandener Methoden für die Mehrziel-Evolutionären-Algorithmen (engl. Multi-objective Evolutionary Algorithm (MOEA)) und die Entwicklung neuer Methoden, um bessere Konvergenz und Diversität in geringer Rechenzeit zu erreichen.

Die Verkürzung der Rechenzeit ist durch die Nutzung geeigneter Datenstrukturen und neuer Archivierungsmethoden für die Elemente der Annährungsmengen erreicht worden. Ein neuer hybrider MOEA ist hier durch die Ergänzung bestehender MOEA mit einer kontrollierbaren Suchmethode entwickelt worden, die eine erwünschte Konvergenz liefert.

Seit 1995 wird die Suchstrategie von evolutionären Algorithmen durch neue Verfahren – Particle Swarm Optimization (PSO) genannt – verbessert. PSO liegt die Beobachtung des sozialen Verhaltens von Tieren zugrunde, welche in großen Verbänden leben und ihr Verhalten einem in der Regel am besten ausgebildeten Leittier anpassen.

Hier wird das neue Verfahren Mehrziel-PSO (engl. Multi-Objective PSO (MOPSO)) vorgestellt, welches bessere Konvergenz und Diversität von Lösungen bietet als die bisher existierenden MOEA. Die Überlegenheit von MOPSO ist durch zahlreiche Testaufgaben und zwei realistische Problemstellungen – nämlich Antennenentwurf und Parametrisierung von Kraftfeldern – gezeigt worden. Außerdem ist ein neues Maß für die Diversität der Annährungspunkte vorgeschlagen und an den betrachteten Problemen angewandt worden.

I would like to thank my supervisor Prof. Dr.-Ing. Jürgen Teich for giving me the opportunity of doing a PhD, for his support, and for teaching me how to work efficiently.
Gratitude goes also to my two co-referents Prof. Dr. Michael Dellnitz and Prof. Dr.-Ing. Klaus Meerkötter for agreeing to do this important task.
I would also like to thank Safoora Sedigh for the careful English proofreading.

Contents

List of Acronyms

AGA	Adaptive Grid Archiving
ANN	Artificial Neural Network
AR1	Rudolph and Agapie's Elitist Genetic Algorithm
CHARMM	Chemistry at Harvard Molecular Mechanics
EA	Evolutionary Algorithm
ER	Error Ratio
GD	Generational Distance
HMOEA	Hybrid MOEA
LAHC	Lebesgue Archiving Hill Climber
MM3	Molecular Mechanics version 3
MO	Multi-objective Optimization
MOEA	Multi-Objective Evolutionary Algorithm
MOP	Multi-objective Optimization Problem
MOPSO	Multi-Objective Particle Swarm Optimization
NSGAII	Non-dominated Sorting Genetic Algorithm II
PAES	Pareto-Archived Evolution Strategy
PSO	Particle Swarm Optimization
SBX	Simulated Binary Cross-over
SPEA	Strength Pareto Evolutionary Algorithm
VEGA	Vector Evaluated Genetic Algorithm

List of Symbols

n	Number of parameters
m	Number of objectives
$\vec{x}$	Decision vector
$\vec{f}(\vec{x})$	Objective vector
S	Feasible decision space
A	Archive
P	Population
T	Maximum number of generations
N	Population size
p_m	Mutation probability
p_c	Cross-over probability
$S(i)$	Strength value of individual i
$R(i)$	Raw Fitness value of individual i
w	Inertia weight
v	Velocity
p^i	Best position of individual i in the parameter space
p^g	Global best position
R_T	Turbulence factor
$\vec{c}$	Composite vector
τ	List of composite vectors
k	Successorship
$S_i(k)$	k-Set
$\vec{\sigma}$	Sigma vector
$\tilde{\sigma}$	Performance metric (median Sigma vector)
C	Performance metric (C metric)
D	Performance metric (Sigma diversity metric)
$\mathcal{B}_k$	Box covering
B	Box in parameter space
P_B	Set of test points

Chapter 1

Introduction

In our daily lives we are inevitably involved in optimization. How to get to the university in the least time is a simple optimization problem that we encounter every morning. Just looking around ourselves we can see many examples of optimization problems even with conflicting objectives and higher complexities. It is natural to want everything to be as good as possible, in other words optimal. The difficulty arises when there are conflicts between different goals and objectives. Indeed, many real-world optimization problems with multiple conflicting objectives exist in science and industry, which are of great complexity. We call them Multi-objective Optimization Problems (MOPs).
Over the past decade, lots of new ideas have been investigated and studied to solve optimization problems. Any new development in optimization which leads to a better solution of a particular problem is of considerable value to science and industry. Therefore, many utilities are available besides classical optimization techniques that use stochastic iterative search methods like Evolutionary Algorithms (EAs) and Particle Swarm Optimization (PSO).
At the beginning of this chapter, a brief introduction on MOP and existing works are outlined. Then the challenges and problems are studied and followed by the overview and objectives of this thesis. Finally, a document organization is given to sketch the remaining chapters.

1.1 Multi-objective Optimization

Multi-objective Optimization (MO) methods are used when there exist several objectives to be optimized at the same time. Consider the functions $f_1(x)$ and $f_2(x)$ in Figure 1.1 (a). We observe that the minimum of each of these functions are $f_1(0)$ and $f_2(1)$, i.e., in $x = 0$ and $x = 1$. Now, if we try to minimize both of them at the

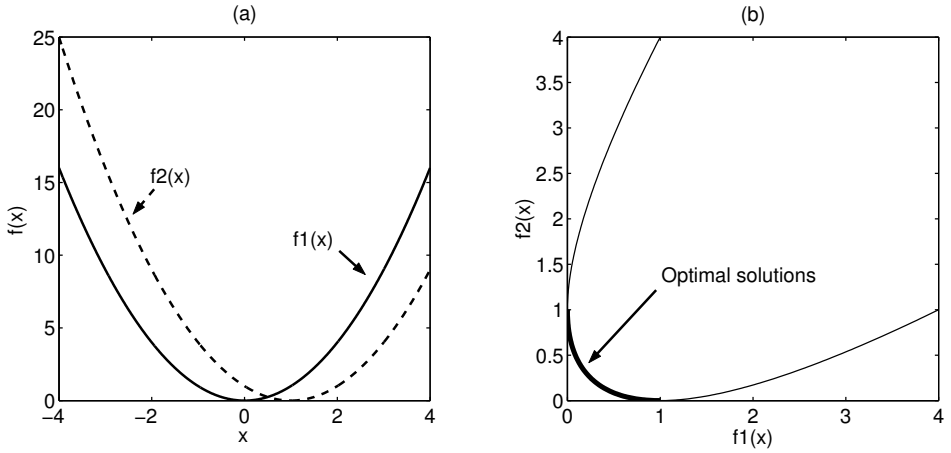


Figure 1.1: (a) 2-objective optimization problem (b) Objective space

same time, there is no single minimum. $x = 0$ is not a minimum of $f_2(x)$ and $x = 1$ is not a minimum of $f_1(x)$. In this case the x values in $[0,\ 1]$ are called the *optimal solutions*. Indeed, multiple optimal solutions in multi-objective optimization arise because of trade-offs between conflicting objectives. Therefore, without further information no solution from the set of optimals can be considered better than any other. In the following, we state the multi-objective optimization problem in its general form:

A **MOP** has several objective functions which are to be minimized or maximized at the same time:

$$\begin{aligned} \text{minimize} \quad & \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x})) \\ \text{subject to} \quad & \vec{e}(\vec{x}) = (e_1(\vec{x}), e_2(\vec{x}), \cdot, e_k(\vec{x})) \leq \vec{0} \\ & \vec{x} \in S \end{aligned} \tag{1.1}$$

involving $m(\geq 2)$ conflicting objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ that we want to minimize simultaneously. The *decision vectors*[1] $\vec{x} = (x_1, x_2, \cdots, x_n)^T$ belong to the feasible region $S \subset \mathbb{R}^n$. The feasible region is formed by constraint functions $\vec{e}(\vec{x})$. We denote the image of the feasible region by $Z \subset \mathbb{R}^m$ and call it feasible objective region. The elements of Z are called objective vectors and they consist of objective (function) values $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x}))$. Figure 1.1 (b) shows the set of optimal solutions of Figure 1.1 (a) in the *objective space*. Indeed in the objective space, each objective is assigned to one of the coordinate axis.

For clarity and simplicity of the treatment we assume that all the objective functions are to be **minimized**. If an objective function f_i is to be maximized, it is equivalent

[1] Decision vectors are also called *parameters*.

to minimize the function $-f_i$.
In the following, some general concepts and notations are presented. All the vectors are assumed to be column vectors.

Definition 1.1 (Dominance) *A decision vector $\vec{x}_1 \in S$ is said to* **dominate** *a decision vector $\vec{x}_2 \in S$ (denoted $\vec{x}_1 \prec \vec{x}_2$) if:*

1. *The decision vector $\vec{x}_1$ is not worse than $\vec{x}_2$ in all objectives, or*

   $f_i(\vec{x}_1) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m.$

2. *The decision vector $\vec{x}_1$ is strictly better than $\vec{x}_2$ in at least one objective, or $f_i(\vec{x}_1) < f_i(\vec{x}_2)$ for at least one $i = 1, \ldots, m$.*

A decision vector $\vec{x}_1 \in S$ **weakly dominates** *$\vec{x}_2 \in S$ (denoted $\vec{x}_1 \preceq \vec{x}_2$) if:*

The decision vector $\vec{x}_1$ is not worse than $\vec{x}_2$ in all objectives, or

$f_i(\vec{x}_1) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m.$

A decision vector $\vec{x}_1 \in S$ is **indifferent** *to $\vec{x}_2 \in S$ (denoted $\vec{x}_1 \sim \vec{x}_2$) if:*

$\vec{x}_1 \not\preceq \vec{x}_2 \wedge \vec{x}_2 \not\preceq \vec{x}_1.$

Figure 1.2 (a) shows an example of Definition 1.1 graphically.

Definition 1.2 (Non-dominated Set) *Among a set of solutions P, the non-dominated set of solutions P' contains those solutions that are not dominated by any member of the set P.*

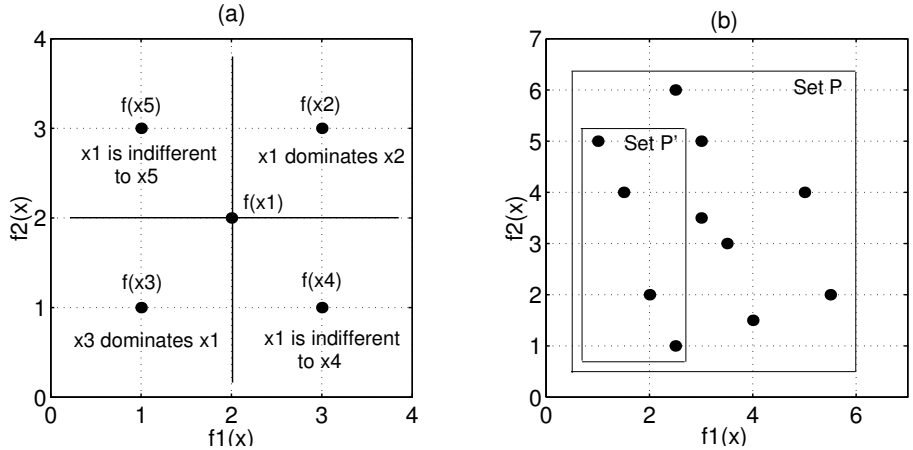


Figure 1.2: Objective space (a) an example of domination (b) P is a set of solutions and P' is the non-dominated set

Figure 1.2 (b) shows an example of the non-dominated set P', graphically. The members of P' are the non-dominated solutions of the set P.

Definition 1.3 (Weakly Non-dominance) *A decision vector $\vec{x}_1 \in S$ is a* **weakly non-dominated** *solution if there is no $\vec{x}_2 \in S$ such that $f_i(\vec{x}_2) < f_i(\vec{x}_1)\ \forall i = 1, \ldots, m$.*
Among a set of solutions P, the weakly non-dominated set of solutions P'' contains weakly non-dominated solutions of the set P.

Figure 1.3 (a) shows a set of weakly non-dominated solutions and a non-dominated set.

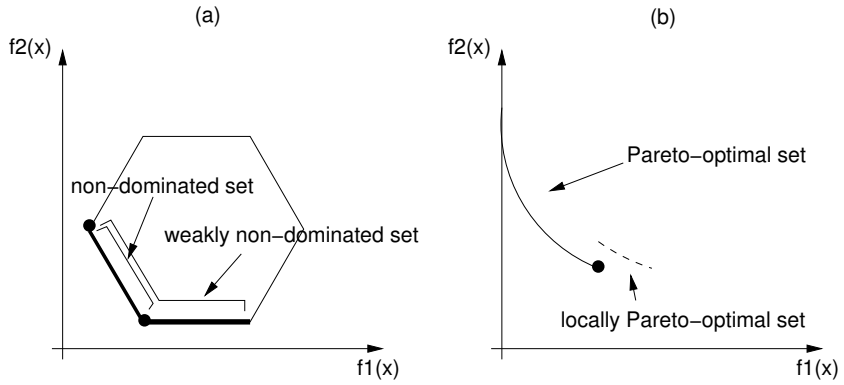


Figure 1.3: (a) Weakly non-dominated set (b) Locally Pareto-optimal set

Pareto Terminology

In MOP it is not possible to find a single solution that would be *optimal* for all the objectives simultaneously. In this case, there exist some optimal solutions where none of their objective values can be improved without deterioration of at least one of the other objective values. In other words, there is no way to improve an objective value of an optimal solution without increasing[2] other objective values. Edgeworth presented this definition in 1881 [Edg81]. However, the definition is called *Pareto optimality*, named after the French-Italian economist and sociologist Vilfredo Pareto [Par06, Mie99].

Definition 1.4 (Pareto-optimal Solution) *A decision vector $\vec{x}_1 \in S$ is called Pareto-optimal if there is no other $\vec{x}_2 \in S$ that dominates it. An objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal.*

[2]This is considered in the minimization cases.

A multi-objective optimization problem is solved mathematically, when all its Pareto-optimal solutions are found.

Definition 1.5 (Pareto-optimal Set) *The non-dominated set of the entire feasible search space S, is called the Pareto-optimal set. The Pareto-optimal set in the objective space is called Pareto-optimal front or simply Pareto-optimal front.*

An approximation of the Pareto-optimal set contains a finite number of non-dominated solutions, which are very close to the set of Pareto-optimal solutions. It is also called *quality set.*

Definition 1.6 (Locally Pareto-optimal Set) *If for any member $\vec{x}_1 \in S$ in a set P''' there is no solution $\vec{x}_2 \in S$ (in the neighborhood of $\vec{x}_1$ such that $\|\vec{x}_2 - \vec{x}_1\|_\infty \leq \epsilon$, where $\epsilon > 0$) dominating any member of the set P''', then the solutions belonging to the set P''' constitute a locally Pareto-optimal set.*

Figure 1.3 (b) shows an example of a locally Pareto-optimal set and a (global) Pareto-optimal set in the objective space.

1.2 Existing Work

One of the first ideas in solving a MOP is to convert it to a single-objective optimization problem. This has been done in a family of *weighting methods* about 40 years ago [Zad63]. The weighting method is known as a classical method. Other classical methods are also available such as; the ϵ-constraint method, min-max goal programming, sequential quadratic programming, and homotopy approach [Mie99, Deb01, Hil01]. Most of these classical methods can find Pareto-optimal solutions. However, when dealing with special problems with non-convex Pareto-optimal fronts, they cannot find all the Pareto-optimal solutions. In most of the classical methods, only one Pareto-optimal solution can be found in one simulation run and most of these algorithms require some prior knowledge such as suitable weights or ϵ values [Deb01]. Hence, scientists have searched for a method to find a set of optimal solutions in each simulation run. This was made possible during the last decade by using population-based stochastic search and optimization methods like EAs.
In EAs, since a population of solutions are processed in each iteration, the outcome is also a population of solutions. If an optimization problem has a single optimum, then all the population members can be expected to converge to that optimum solution. However, if an optimization problem has multiple optimal solutions, the Evolutionary Algorithm is used to capture multiple solutions in its final population.

Due to this particular property of multi-objective optimization problems, Evolutionary Algorithms are being studied more and more during the last decade. These methods are called Multi-Objective Evolutionary Algorithms (MOEAs).
The history of MOEAs goes back to the first MOEA, called Vector Evaluated Genetic Algorithm (VEGA), introduced by Schaffer in 1985 [Sch85]. This method is based on objective-wise selection. VEGA is the simplest possible MOEA and is a straightforward extension of a single-objective EA. It works efficiently for some generations, but in some cases suffers from the principal of not using the domination criterion in fitness evaluation. Therefore, it is not able to deliver a good spread of solutions in some cases. In 1991, Kursawe introduced his method, which uses the same idea as VEGA [Kur91]. Later, Fonseca and Fleming (MOGA) [FF93], Horn and Nafpliotis (NPGA) [HNG94], and Srinivas and Deb (NSGA) [SD94] introduced other kinds of MOEAs which use Pareto-based (domination-based) selection, niching and visual comparisons. In these methods, the difficulty of VEGA in finding a good diversity of solutions is eliminated by introducing the non-domination concept and an explicit diversity-preserving operator.
In 1999, Zitzler and Thiele added elitism as an important part in MOEA [ZT99], which can guarantee convergence. Their method, Strength Pareto Evolutionary Algorithm (SPEA) and also later methods from Knowles and Corne (PAES, PESA, 1999, 2000) [KC99], Deb et al. (NSGAII, 2000) [DAPM00] and Zitzler et al. (SPEA2, 2001) [ZLT02] made another category of MOEA based on archiving, elitism, and quantitative performance metrics. These methods are supposedly faster and better than the previous methods. This is due to the elite-preserving operator. An elite-preserving operator gives the elite solutions to be directly carried over to the next generations. Therefore, it ensures that the fitness of the best solution does not deteriorate.
Hence, the MOEA methods are arranged into Pioneers, Classic, and Elitist categories. These methods constitute the basic part of the MOEA, however there are many other recorded MOEA methods for particular problems [CVL02, Deb01].
In the search for faster population-based stochastic search methods, researchers have come to the Particle Swarm Optimization (PSO) technique [KE01]. The single-objective PSO is recorded to be faster than Evolutionary Algorithms. Therefore, the scientists are also motivated to work on a multi-objective version of PSO. The first Multi-Objective Particle Swarm Optimization (MOPSO) methods have been suggested by Coello Coello and Lechuga [CL02] and Hu and Eberhart [HE02] in 2002.

1.3 Challenges and Problems

Multi-objective optimization methods must be able to find optimal solutions with the following four properties:

- several optimal solutions as the output,
- convergence of the solutions to the Pareto-optimal front,
- diversity of the solutions,
- low computational time.

Indeed, obtaining high convergence and diversity of solutions in low computational time is always the goal of MO methods. The computational time of MO increases, especially when elitism in the form of an external population is used. Existence of the external population or the archive increases the convergence of the solutions, since the discovered non-dominated solutions of each generation are stored in it. However, since it must be updated in each generation, therefore the computational time increases. One possibility to reduce the update time of the archive is to make the updating process faster by using improved data structures. Until 2002, linear lists were used as the data structure for storing the archive members. However, they require a lot of computational time the large archive and population sizes. In 2002, Mostaghim et al. investigated Quad-trees [Hab83] as another data structure, for the first time in the context of MOEA [MTT02, MT03b]. Quad-trees need less computational time than linear lists, particularly for larger sized populations. Later in 2002, Fieldsend et al. introduced the dominated-tree data structure [FES02], which reduces the computational time of MOEAs with small populations and large numbers of generations. Indeed, both the Quad-tree and dominated-tree data structures reduce the computational time of MOEA methods. However, they are not comparable with each other, since the dominated-tree requires less computational time than linear lists, when it is used in MOEAs with small populations, large numbers of generations, and large archives, whereas, the Quad-trees require less computational time than linear lists when they are integrated in the MOEAs with large populations and small archives with low number of objectives. Here, it must be emphasized that the archive must be updated after each generation. This makes a considerable difference when we don't use these data structures in MOEAs [Hab83, SS96b, SS96a, Sch03, Sch04].
Also, the archive size plays an important role in the diversity of solutions and the computational time. Bounding the archive size not only makes the method faster, but also allows for diverse solutions [FES02, Zit99]. On the other hand, a restricted amount of solutions in the archive is desired by many decision makers. In general, storing only a bounded number of non-dominated solutions is desirable due to

the following; true non-dominated sets are infinitely large, the computational and memory overheads of maintaining the archive are reduced, and diversity of solutions can be obtained by removing the selection pressure. Truncation [ZLT02], clustering [Zit99], and Adaptive Grid Archiving (AGA) [KC03] techniques are methods for restricting the archive size while keeping diverse solutions. However, bounding the archive size requires a high computational time and any other technique which bounds the archive size by keeping diverse solutions has considerable importance.
Until now, elitism has been discussed as a satisfactory step in MOEAs for obtaining good convergence of solutions. However, convergence is not obtained only by the archive, but by the effective search strategy of the Evolutionary Algorithms, i.e., cross-over and mutation. Indeed, the *inheritance* property obtained from recombination of solutions makes the method to find good solutions during generations. The search process is controlled by recombination, fitness evaluation, and random selection. Fitness evaluation has been an important topic and is studied by many researchers [Sch85, FF93, HNG94, SD94, ZT99, KC99, DAPM00, ZLT02]. The result of the search process is that Evolutionary Algorithms are able to find some very good (converged) solutions in a low number of generations. However, covering the entire Pareto-front requires an extensive number of generations. Indeed, the search process explores the whole search space in each generation; the good solutions (with better fitness values) are selected as well, but it can result in an already explored region being explored again. This brings the idea of changing the search process in some sense to a controllable search method. Controllable exploration is proposed by using binary search methods [Hug03], and combining Evolutionary Algorithms with subdivision methods [SMDT03, BH03]. The basic idea of the subdivision techniques is to divide the search space into subspaces [DJ98, DH97].
Controllable exploration brings the new idea of using the PSO [KE01, KE95] method. In PSO, there is no selection, the solutions are assigned some velocities and move towards the optimum. A turbulence factor is also defined, similar to the mutation operator, in order to avoid local optimum. The results of single-objective PSO are very satisfactory [KE01]. However, changing the PSO to MOPSO has been investigated first in 2002 by Coello Coello and Lechuga [CL02]. Later, Fieldsend and Singh [FS02] and Mostaghim and Teich [MT03b] proposed other variants of MOPSO with the aim of obtaining better convergence and diversity of solutions. The comparisons with MOEAs show that MOPSO is a good method to optimize MOPs. However, obtaining better diversity and convergence of solutions for high number of objectives must be investigated further.
Another important issue in MO is the quantitative comparison of the performance of different algorithms. The outcome of the MOEA or MOPSO is usually an ap-

proximation of the Pareto-optimal front, which is denoted as an approximation set, leading many researchers to investigate the evaluation methods of the quality of the approximated front, e.g., [KC02, Deb01, ZTL+02, HJ98]. The approximation sets can be compared by measuring the diversity and convergence of solutions in the objective space separately. There also exist convergence and diversity metrics [FMA02, Zit99, DMM03a]. However, before comparing the diversity of solutions, it must be clear what percentage of the space is occupied by the solutions and where the solutions are. This is more important, when we increase the number of objectives. For example, it is not possible to observe the solutions of a 4-objective MOP graphically. This is the problem of most of the diversity metrics, they only compute a value of for example the hyper-volume of the region made by the approximation set. This value is useful when comparing two approximated sets.
Motivation for optimization comes from real-world optimization problems. The improvements in the MO techniques are completed step by step by encountering different problems of different levels of difficulties. Therefore, there are many challenges involved in solving real-world optimization problems.

1.4 Research Goals and Specific Objectives

The main goal of this thesis is to improve elitist MOEAs in order to achieve better convergence and diversity of solutions in low computational time. The major achievements of this thesis can be summarized as follows:
In a category of elitist MOEAs, the elite solutions of each generation are stored into an archive. The data structures and algorithms for storing and updating archives may have a great impact on the computational time, especially when optimizing continuous problems with larger population sizes. Therefore, the first objective is to study the data structures of the archive. Here, the problem of storing the Pareto-optimal solutions in the archive is addressed in such a way, which helps to attain the desired results in the least possible time. Hence, an efficient data structure is required in order to hold these solutions.

Objective 1: Possible data structures for storing non-dominated solutions in the archive in order to reduce computational time

The second objective is to investigate a Hybrid MOEA (HMOEA) in order to obtain good convergence and diversity of the solutions. This is performed by HMOEA, which is a combination of MOEA and subdivision techniques. One good property of the HMOEA is the controllable exploration of the search space, which helps to attain

good convergence of solutions and even covering the approximated Pareto-optimal front.

> Objective 2: Possible combination of MOEAs with other non-stochastic methods

The disadvantage of using HMOEAs is the high computational time needed for MOPs with high number of parameters. As the third objective, Multi-objective PSO is studied and compared with MOEAs in terms of convergence and diversity of solutions.

> Objective 3: Formulation of Multi-Objective Particle Swarm Optimization (MOPSO) techniques

Indeed, MOPSO should take less computational time than MOEAs, since there is no fitness evaluation, selection, and recombination involved. Another time-consuming part of a MOEA is the process of bounding the archive size. A bounding technique must be applied when the size of the archive exceeds a fixed size. Therefore, the fourth objective is dedicated to the investigation of a technique called ϵ-dominance for bounding the archive size.

> Objective 4: Demonstration of the influences of ϵ-dominance on the size of the archive, convergence and diversity of solutions

The fifth objective of this thesis concerns the metrics on measuring the diversity of the solutions. There are several diversity metrics, each of them having some disadvantages as well as advantages. The previous diversity metrics become more complicated when dealing with high dimensional spaces and they do not express any knowledge about the solutions on the approximated Pareto-optimal front (e.g., the percentage of the objective space occupied by the solutions). Therefore, the next objective is to investigate a variant of a diversity measure which provides us with more information about the found solutions along the approximated Pareto-optimal front.

> Objective 5: Development of a metric to measure the diversity of solutions along an approximated Pareto-optimal front

The last objective concerns the application of MOEAs and MOPSOs for solving real-world problems. This is achieved by two real-world examples in antenna design and computational chemistry.

> Objective 6: Demonstration of the ability of MOEAs and MOPSOs in solving real-world problems

1.5 Document Organization

This thesis is organized as follows. Chapter 2 reviews the basic concepts of Evolutionary Algorithms and MOEAs. In this chapter, different test functions and comparison metrics, which are referred to during the thesis, are studied. Chapter 3 is dedicated to data structures for storing non-dominate solutions in archive-based MOEAs. In Chapter 4, Hybrid MOEAs and covering the approximated Pareto-optimal front are studied. PSO and MOPSO methods are investigated in Chapter 5. In this chapter, the so-called Sigma method and the Sigma diversity metrics are introduced and covering the approximated Pareto-optimal front by MOPSO is studied. Chapter 6 is being dedicated to bounding the archive size by using a so-called ϵMOPSO method. In Chapter 7, the applications of MOPSO and MOEA methods using two real-world examples are studied, and Chapter 8 concludes the thesis.

Chapter 2

Multi-Objective Evolutionary Algorithm (MOEA)

Even though an algorithm is in fact a purely algebraic event, it seems that computer scientists find it easier to grasp an abstract phenomenon when it is grounded in a familiar metaphor, whether it is evolution, insect swarming, immune systems, pheromone following or neural dynamics.

J. Kennedy and R. Eberhart[1]

In this chapter, the basic concepts of Evolutionary Algorithms (EAs) and Multi-Objective Evolutionary Algorithms (MOEAs) are briefly reviewed. Particularly, the Strength Pareto Evolutionary Algorithm (SPEA) [ZLT02] is studied, as it plays an important role in comparing methods in this thesis. Test problems and comparison metrics are also other issues in MOEAs, which are outlined later in this chapter.
It must be emphasized, that the algorithms and the methods explained here do not reflect EAs and MOEAs for a general form. The parameters and operators can be defined in other ways, which are not presented here. Various aspects of EAs and MOEAs may be found in [CVL02, Deb01, Gol89, Bäc96, Fog95].

2.1 Evolutionary Algorithms

EAs denote a famous class of population-based iterative search methods, which are based on abstractions of the processes of Darwinism evolution. Actually, they have the potential to find a set of optimal solutions in a single simulation run. Different kinds of EA methods exist, whereas, all EAs have some basic elements in common,

[1] From the book *New Ideas in Optimization*, D. Corne et. al [CDG99].

as follows:

- EAs typically work with a population of candidate solutions at a time, rather than a single candidate solution.
- EAs incorporate a selection method based on the fitness values. The better the fitness of a candidate solution, the more often it is selected, and the more parts of it will be passed on to later generations.
- Inheritance: The new candidate solutions are obtained by recombining the selected candidate solutions.

Therefore, an EA is characterized by the fact that a number, N, of potential solutions (called *individuals* $j \in X$, where X represents the space of all possible individuals) of the optimization problem simultaneously sample the search space. This *population*, $P = \{j_1, j_2, \cdots, j_N\}$, is modified according to the natural evolutionary process: selection and recombination are executed in a loop for a fixed number of iterations. Each run of the loop is called a *generation* and P_t denotes the population at generation t. Algorithm 1 shows a typical basic algorithm for EAs. In the following, different parts of the Algorithm 1 are explained.

Input and Output

The input parameters of an EA are:

- N: number of individuals in the population (population size).
- T: maximum number of generations (to determine the stopping criterion).
- p_m and p_c: probabilities of applying mutation and cross-over operators, respectively.

The output of the algorithm is the optimal solution, which is stored in A ($|A| = 1$).

Population Initialization

The population P at generation t, P_t, consists of N individuals, j_i, $i = 1, \cdots, N$, which are initialized at random. Depending on the problem to be solved, various codings for individuals exist, e.g., the individuals are represented by bit strings, vectors of integers or reals, trees, and graphs. Bitstring individuals are also known as *Chromosomes*. The population itself is typically arranged as a linear list, which must be updated at the end of each generation t.

Fitness Evaluation

The quality of an individual, j_i, is measured by a fitness function, $F(j_i)$. The fitness value of each individual is calculated by the fitness function, and is not necessarily equal to the objective value. The fitness function is a measure of the quality of the candidate solution. This step plays the most important role for selection in the next step. The *constraints* can also be handled in this step by using *penalty* functions to penalize potential solutions that are infeasible. Penalty functions degrade the fitness values of infeasible population members, but still allow them to remain in the population and influence the final solution.

Algorithm 1 : Basic Evolutionary Algorithm

Input: N, T, p_c, p_m
Output: A

1. Initialize population: $P_t = \emptyset$, $t = 0$
for $i = 1$ to N **do**
  Find a random $j_i \in X$
  $P_t = P_t \cup j_i$
end for
2. Fitness Evaluation:
for $i = 1$ to N **do**
  Calculate $F(j_i)$
end for
3. Selection: $P^{'} = \emptyset$ ($P^{'}$: mating pool)
for $i = 1$ to N **do**
  Select one individual j, according to the selection method
  $P^{'} = P^{'} \cup j$
end for
4. Recombination:
cross-over: Choose two individuals randomly from $P^{'}$. Apply cross-over operator on them with a probability of p_c. Replace the (new) individuals in $P^{'}$.
mutation: Choose one individual randomly from $P^{'}$. Apply mutation operator on it with a probability of p_m. Replace it in $P^{'}$.
5. Termination: $P_{t+1} = P^{'}$, $t = t + 1$
if $t > T$ **then**
  $A = P_t$, STOP
else
  goto Step 2
end if

Selection

The selection operator is intended to improve the average quality of the population by giving individuals of higher quality a higher probability of survival. Selection thereby focuses the search on promising regions in the search space. There are different kinds of selection operators, e.g., *Tournament* and *Roulette Wheel* Selection [Gol89, Deb01]. In *Tournament Selection*, k individuals are chosen at random and the best one is selected, whereas in *Roulette Wheel Selection*, each population member is assigned a proportion of the Roulette Wheel equal to the ratio of its fitness to the sum of the entire population's fitness. Depending on the problems and their restrictions, certain characteristics can be applied on the selection process. If the selection operator uses much *selection pressure*, i.e., it emphasizes the population's best solution by assigning many copies of it, the algorithm loses the diversity of its solutions very quickly. On the other hand, if the selection pressure is very low, the method will behave like a random search. The speed with which the best solution in a population is emphasized can be determined by the characteristic time of a selection operator. *Take over time* is defined as the time required for the best solution in the population to occupy all of the population slots by repetitive application of the selection operator alone. The handling of the constraints can also be carried out in the selection step [Deb01]. This method compares two selected individuals, if both of them are feasible, the individual with "better" fitness value is selected. In the case that one of them is infeasible, the feasible individual is selected and when both of them are infeasible, the individual with smaller overall constraint violation is chosen.

Recombination

Recombination changes the genetic material in the population, either by cross-over or by mutation, in order to explore new points in the search space. The choice of coding also determines the recombination operator.

- **Cross-over:** There are various kinds of cross-over operators, depending on the coding of individuals, the problem, the search space, and many other parameters. For example, the cross-over operators for string (bitstring or integer value) individuals are: single point, n point, uniform, and edge, whereas, for real value individuals the cross-over operators are: linear, blend, simulated binary cross-over operators [Spe98, Deb01]. Indeed, the main idea in utilizing cross-over is searching the search space locally. This is particularly achieved for bitstring coding of individuals. The cross-over operator is usually applied on two randomly selected individuals with a probability of p_c [Spe98].

  - Single point cross-over: This is achieved by randomly choosing a crossing point along the string. All the string elements (bits) following the crossing point are exchanged.
  - n point cross-over: This operator functions similarly to the single point cross-over operator, except that $2n$ cross-over points are randomly chosen. The segments of the string between the $2n$ points are exchanged.
  - Uniform cross-over: This operator is the extreme version of the previous operators. It randomly chooses different bits from each of the parents, with equal probabilities.
  - Simulated binary cross-over (SBX): This operator is introduced by Deb et al. [Deb01] for real vector individuals. Indeed, SBX simulates the single point cross-over on real vectors.

- **Mutation:** Mutation operators also depend on the coding of the individuals [Spe98]. These operators are applied on the individuals of a population with a probability of p_m. For bitstring individuals, it changes a bit from 0 to 1 and vice versa. There are different mutation operators for different kinds of individuals. The aim of mutation is to apply a sudden large change on the bitstring individuals for performing global searching.

Termination

An EA can terminate when a termination criterion is fulfilled. Termination criteria can be defined by a maximum number of generations (e.g., T in Algorithm 1) or when there has been no change in the population for a given number of generations.

2.2 MOEA Methods

One of the goals of multi-objective optimization is to approximate the set of Pareto-optimal solutions. Sets of solutions can be easily found in one simulation run by EAs. Indeed, EAs explore the search space by several individuals in the population. Therefore, they can simultaneously explore the search space and under certain additional assumptions can converge to a set of optimal solutions. In other words, in single-objective optimization the whole population converges to a single minimum, whereas in multi-objective optimization they can theoretically converge to the set of Pareto-optimal solutions [Rud98, RA00]. This advantage has attracted many researchers to solve MOPs using Multi-objective Evolutionary Algorithms (MOEAs).

The structure of MOEA is very similar to EA, with some differences, e.g., in fitness evaluation. There are many MOEA methods, which are divided into different categories (see Section 1.2). One group of MOEA methods are *elitist* MOEAs. There, the best solutions of each generation are passed to the next generation either by adding them directly to the next population or by storing them in an external population called *archive*. In fact, the elites cannot be lost during generations and their presence enhances the probability of creating better offsprings. It has been proved [Rud98] that a class of EAs converges to the global optimal solution of some functions in the presence of elitism (see Section 2.3). Elitist MOEAs are investigated in algorithms like Non-dominated Sorting Genetic Algorithm (NSGAII) [DAPM00], Pareto-Archived Evolution Strategy (PAES) [KC99], Rudolph and Agapie's Elitist (AR) [RA00], and Strength Pareto Evolutionary Algorithms (SPEA2) [ZLT02] (refer to [Deb01, CVL02] for more details). However, a large number of publications are generated each year with respect to the interest in MOEA. In the years 2001 and 2003, the first and second international conferences on MOEA were held in Zurich, Switzerland and Faro, Portugal, respectively. These conferences were dedicated to MOEAs; theory, design, analysis, and applications. Also, in other conferences there are special sessions dedicated to MOEAs, where international researchers discuss different issues in MOEAs. These points illustrate the intense interest in MOEAs for solving academic and real-world MOPs[2].
In the following, some of these recently proposed algorithms are briefly reviewed.

Rudolph and Agapie's Elitist Genetic Algorithm (AR1)

Rudolph and Agapie suggested AR1 as an elitist MOEA [RA00]. In AR1, μ parents are used to create λ ($\lambda \geq \mu$) offsprings using genetic operators. So in each generation, there are two populations, the parent population P_t and the offspring population Q_t. The algorithm works in three phases.

1. Non-dominated solutions of Q_t are identified, deleted from Q_t, and placed into an elite population A_t.

2. Each solution q of A_t is compared with each solution of the parent population P_t. If q dominates a solution in P_t, that solution is deleted from P_t and q is moved into a set P'_t. On the other hand, if q is indifferent to the solutions in P_t, it is removed from A_t and out into another set, Q'_t.

[2]One of the most complete listings of publications in the MOEA field is maintained in a database at the web address http://www.lania.mx/ ccoello/EMOO/EMOOconferences.html by Dr. Carlos Coello Coello. Currently, this database contains over 1000 MOEA citations.

3. In the third phase, all the above sets are arranged in a special order of performance. First, P_t and P'_t are combined. If together they do not fill up the whole population, solutions are taken from the sets in the following order: Q'_t, A_t and Q_t.

This algorithm guarantees convergence to the true Pareto-optimal front. However, a disadvantage is that it doesn't guarantee a good distribution of diverse solutions on the Pareto-optimal front [Deb01].

Elitist Non-dominated Sorting Genetic Algorithm (NSGAII)

In this algorithm [DAPM00] introduced by Deb et al., the offspring population Q_t of size N is created from the parent population P_t of size N. First, these two populations are combined to form a population R_t of size $2N$. Then, a non-dominated sorting procedure is used to classify the entire population R_t as follows:

1. Non-dominated individuals are calculated from R_t and are called non-dominated solutions of front 1. Then, these are temporarily disregarded from R_t and the non-dominated solutions of the remaining elements of R_t are then determined and called non-dominated solutions of front 2. This procedure is continued until all the members of R_t are classified into a non-dominated front. This is called *sorting* process.

2. After the sorting process, the new population is filled with solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front. Since R_t has $2N$ solutions and the new population must have N solutions, just N best solutions appear in the new population (N solutions from the best fronts). In the case of inadequate available space in the new population to accommodate all solutions of a non-dominated set, a crowding strategy is used to identify solutions which reside in less crowded areas (in the objective space).

In this algorithm, the non-dominated solutions of each generation are added to the next population. Therefore, they will not be lost during generations. A new variant of NSGA2 is called CNSGA2 [DMM03a], which uses a clustering technique instead of a crowding strategy in the second step.

Pareto-Archived Evolution Strategy (PAES)

In this algorithm [KC99] introduced by Knowles and Corne, the offspring population Q_t of size N is created from the parent population P_t of size N. Then each offspring

is compared with its parents. If it dominates a parent, it is accepted as a parent in the next generation. If one of the parents dominates it, the offspring is discarded and a new offspring must be created. In the case that both are indifferent, the offspring will be compared with an archive of current best solutions. If it dominates any member in the archive, it will be a parent in the next generation. But if it does not dominate any member in the archive, both of its parents and the offspring are compared for their nearness (distance) to the members of the archive. If the offspring resides in a least crowded region (in objective space), it is accepted as a parent and will be added to the archive. In the case that the offspring and the parents have the same nearness (distance) to the archive, one of them is selected at random.
In this algorithm, the non-dominated solutions of each generation are stored in an external population (archive).

2.2.1 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 [ZLT02] is the second version of the SPEA [Zit99] method and in contrast to its predecessor, it incorporates a fine-grained fitness assignment strategy and a density estimation technique that are explained in the following. This method uses Pareto-dominance based selection [Gol89] and elitism where the policy is to always include the $\overline{N}$ best individuals of the current generation into the next generation in order not to lose them during exploration. These $\overline{N}$ best individuals are stored in an archive A_t. In order to maintain a high diversity within the population, it uses an enhanced truncation method. Altogether, SPEA2 has shown to provide superior results compared to existing approaches (see, e.g., [ZLT02] for many problems of interest). The algorithm of SPEA2 is listed in Algorithm 2.

Fitness Assignment

In SPEA2 [ZLT02], both dominating and dominated solutions are taken into account for calculating the fitness value of each individual (to avoid the situation that individuals dominated by the same archive members get identical fitness values). In detail, a strength value $S(j)$ is assigned to each individual j in the archive A_t and the population P_t, representing the number of solutions it dominates:

$$S(j) = \left| \left\{ \widetilde{j} \mid \widetilde{j} \in P_t \cup A_t \wedge j \prec \widetilde{j} \right\} \right| \tag{2.1}$$

On the basis of the strength S, the raw fitness $R(j)$ of an individual j is calculated:

$$R(j) = \sum_{\widetilde{j} \in P_t \cup A_t, \widetilde{j} \prec j} S(\widetilde{j}) \tag{2.2}$$

For example, Figure 2.1 shows the raw fitness values for a given population and a minimization problem with two objectives f_1 and f_2.

Algorithm 2 : SPEA2 Algorithm

Input: N, $\overline{N}$, T
Output: A

1. Initialization: Initialize P_t, $A_t = \emptyset$, $t = 0$
2. Fitness Assignment: Calculate fitness values of individuals in P_t and A_t
3. Update: Copy all non-dominated individuals in P_t and A_t to A_{t+1}
if $|A_{t+1}| > \overline{N}$ **then**
  reduce A_{t+1} by means of the truncation operator
else if $|A_{t+1}| < \overline{N}$ **then**
  fill A_{t+1} with dominated individuals in A_t and P_t
end if
4. Termination:
if $t > T$ or another stopping criterion is satisfied **then**
  $A = A_{t+1}$, STOP
end if
5. Selection: Perform binary tournament selection with replacement on A_{t+1} in order to fill the mating pool
6. Recombination: Apply recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population
$t = t + 1$, goto Step 2

Additional density information is incorporated to discriminate between individuals having identical raw fitness values. The density estimation technique used in SPEA2 is an adaptation of a kth nearest neighbor method, where the density at any objective vector j is a function $D(j)$ of the distance to the kth nearest objective vector. The final fitness value will be $F(j) = R(j) + D(j)$. In this case, the final fitness values of the non-dominated points are less than one and not zero anymore.

Truncation Method

In this part which is also called environmental selection, the first step is to copy all non-dominated individuals, i.e., those with a fitness lower than one, from archive and population to the archive of the next generation:

$$A_{t+1} = |\{j \mid j \in P_t \cup A_t \wedge F(j) < 1\}| \tag{2.3}$$

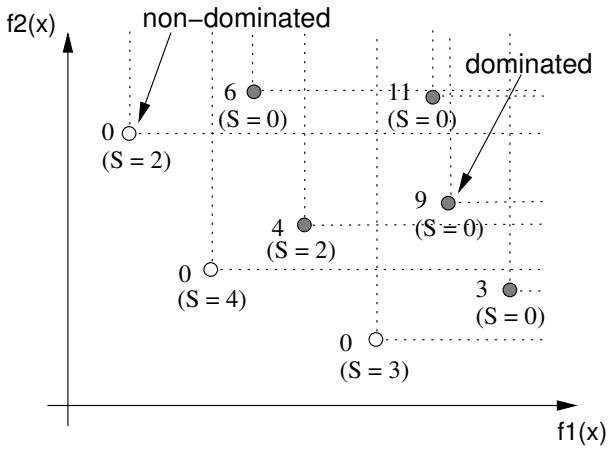


Figure 2.1: Raw fitness values for a minimization problem with two objectives $f_1(x)$ and $f_2(x)$ using SPEA2

where $F(j)$ is the fitness value of j. If the non-dominated front fits exactly into the archive, the environmental selection step is completed. Otherwise, there can be two situations: Either the archive is too small or too large. In the first case, the best dominated individuals in the previous archive and population are copied to the new archive. In the second case, the archive must be truncated and some of the individuals in the archive must be deleted. In SPEA2, the individual with the minimum objective distance to another individual is chosen at each stage; if there are several individuals with minimum distance, the tie is broken by considering the second smallest distance, and so forth.

2.2.2 Summary

By studying previous elitist MOEAs, we can construct a typical simple structure (see Figure 2.2). This typical structure is valid for those elitist MOEAs that store non-dominated solutions in an archive. Indeed, this structure is one possible structure and shows different blocks, which consume most of the computational time. The blocks *Calculating non-dominated solutions*, *Update*, and *Clustering* are related to the elitism. We refer to these blocks as the *Archiving*. Clustering is playing the same role as the truncation method in SPEA2. The blocks *Fitness Evaluation*, *Selection* and *Recombination* are referred to the general steps of a MOEA method.

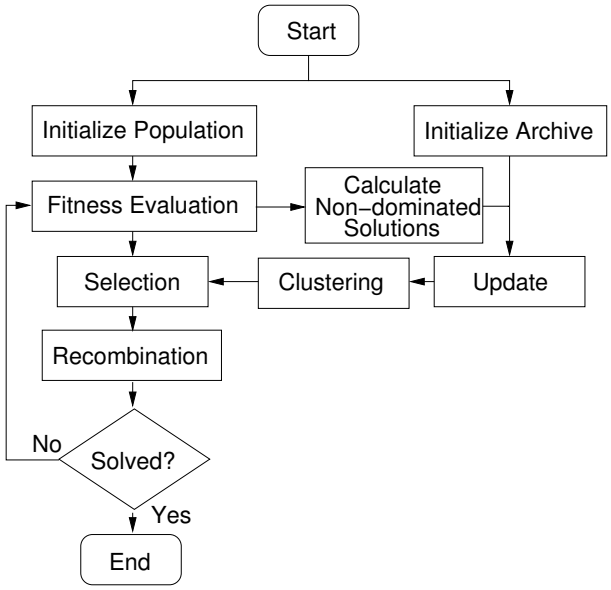


Figure 2.2: A typical structure of an elitist MOEA

2.3 Convergence

Elitism in EAs and MOEAs is aimed at keeping track of the best solutions during the generations. Therefore, one of the reasons for using elitism is to obtain convergence, since the best solutions are not lost during generations. However, it is recorded that under certain conditions the convergence of a single-objective EA does not necessarily depend on elitism [Rud94, Rud98]. It is also shown that the results known from the theory of EAs do not carry over to the MOEA case [Rud98]. Proof of convergence of a MOEA is studied for different structures of MOEAs with Elitism in [RA00, Rud98]. In Appendix A, a basic structure of MOEAs is considered (Figure 2.2) and the convergence proof of such a structure is explained as in [RA00]. Indeed, it is assumed that the Evolutionary Algorithms are *Markov processes* [Häg02], which cope with *partially ordered* fitness values. All the Pareto-optimal solutions will enter the archive in finite time with probability one and the dominated solutions will be discarded. Since Pareto-optimal solutions cannot get lost, one gets convergence with probability one.
Markov chains are also used in the convergence proof of other stochastic iterative search methods such as simulated annealing [Häg02].

2.4 Test Functions

Deb et al. [DTLZ02, Deb01, Zit99] have suggested a series of test functions which are suitable for testing MOEAs. In general form, the test problems must be easy to construct and scalable to have any number of decision variables or number of objectives. Also, the resulting Pareto-optimal front must be easy to comprehend and its exact shape and location must be known. Indeed, the task of the test functions is to test the convergence and diversity abilities of a MOEA method.
Testing the convergence can be achieved by placing some local Pareto-optimal attractors, or biased density of solutions away from the Pareto-optimal front. The diversity ability can be achieved by test functions with non-convex, discrete Pareto-optimal fronts, or having variable density of solutions along the front [DTLZ02].
In this thesis, a good variety of test functions are selected from [DTLZ02, Deb01] to test the ability of the proposed methods. The test functions are listed in Table 2.1. In the following, the properties of the Pareto-optimal fronts and the difficulties integrated in the test functions are summarized.

- ZDT1: (2-objective, 30 parameters)
  Pareto-optimal front: Convex, continuous, uniform distribution of solutions
  Difficulty: Large number of parameters

- ZDT2: (2-objective, 30 parameters)
  Pareto-optimal front: Non-convex, continuous
  Difficulties: Large number of parameters, non-convex front

- ZDT3: (2-objective, 30 parameters)
  Pareto-optimal front: Convex, disconnected-continuous
  Difficulties: Large number of parameters, discontinuous front

- ZDT4: (2-objective, 10 parameters)
  Pareto-optimal front: Convex, continuous
  Difficulty: Large number (21^9) of local Pareto-optimal fronts

- ZDT5: (2-objective, 11 parameters) Boolean function over bitstrings
  Pareto-optimal front: Discontinuous
  Difficulty: Large number (1023) of local Pareto-optimal fronts

- ZDT6 [3]: (2-objective, 10 parameters)
  Pareto-optimal front: Non-convex, non-uniform density of solutions, disconnected-

[3] This function is constructed like ZDT6 in [Deb01]; here the second objective is modified.

continuous
Difficulties: Adverse density of solutions, non-convex front

- TEST1, TEST2: 2- and 3-objective simple examples used in Chapter 4
- CP3: (3-objective, 3 parameters)
  Pareto-optimal front: Comet-like, continuous
  Difficulty: Narrow Pareto-optimal region
- DLZT: (3-objective, 3 parameters)
  Pareto-optimal front: Disconnected-continuous
  Difficulties: Non-linearity (tests the ability to find a good distribution of solutions), non-uniform density in solutions in the search space
- GSPm: (m-objective, m parameters)
  Pareto-optimal front: Continuous, uniform solutions
  Difficulty: Possible high number of objectives

In Table 2.1, GSP and CP are the abbreviations for *Generic Sphere Problem* and *Comet Problem*, respectively.

2.5 Performance Metrics

The result of a MOEA is indeed an approximation of a Pareto-optimal front and this set of solutions should achieve the goals of the MOEA: (i) good convergence of solutions with (ii) high diversity along the obtained approximated front. To measure the quality of the approximated front, we need to examine the convergence and diversity of solutions. This can be achieved by some performance metrics, although in some cases there are visible differences. We also require a metric for comparing the results of different methodologies. Comparing two (approximated) non-dominated sets is possible by comparing the convergence and diversity of solutions, as well as the computational time for obtaining them. Also, there are other metrics, such as the number of obtained solutions. An excellent survey on metrics for comparing non-dominated sets is investigated in [KC02, Zyd03, HJ98, ZTL$^+$02, OJS03]. Here, several comparison metrics are outlined in three categories of cardinality-based, convergence-based, and diversity-based performance metrics.
In the following, R is considered a set of Pareto-optimal solutions or a reference set, and A is dedicated to a non-dominated set of solutions, which is an approximation of a Pareto-optimal front. The goal is to measure the quality of A or compare the qualities of two non-dominated sets A_1 and A_2.

Table 2.1: Test Functions

test	Function	
ZDT1	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g)$	$x_i \in [0,\ 1]$ $n = 30$ $i = 1, 2, \ldots, n$
ZDT2	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ $h(f_1, g) = 1 - (f_1/g)^2$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g)$	$x_i \in [0,\ 1]$ $n = 30$ $i = 1, 2, \ldots, n$
ZDT3	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ $h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g) + 1$	$x_i \in [0,\ 1]$ $n = 30$ $i = 1, 2, \ldots, n$
ZDT4	$g(x_2,\cdots,x_n) = 1 + 10(n-1) + (\sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i))$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g)$	$x_1 \in [0,\ 1]$ $x_i \in [-5,\ 5]$ $n = 10$ $i = 2, \ldots, n$
ZDT5	$g(x_2,\cdots,x_n) = \sum_{i=2}^{n} v(u(x_i))$ $h(f_1, g) = 1/f_1(x)$ $f_1(x_1) = 1 + u(x_1)$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g)$ $v(u(x_i) = \begin{cases} 2 + u(x_i) & if u(x_i) < 5 \\ 1 & if u(x_i) = 5 \end{cases}$	$x_1 = 30$ bits $x_i = 5$ bits $n = 11$ $i = 2, \ldots, n$
ZDT6	$g(x_2,\cdots,x_n) = 1 + 9[(\sum_{i=2}^{n} x_i)/9]^{0.25}$ $h(f_1, g) = 1 - \sqrt{x_1/g}$ $f_1(x_1) = 1 - exp(-4x_1)sin^6(6\pi x_1)$ $f_2(\vec{x}) = g(x_2,\cdots,x_n) \cdot h(f_1, g)$	$x_i \in [0,\ 1]$ $n = 10$ $i = 1, 2, \ldots, n$
TEST1	$f_1(\vec{x}) = (x_1 - 1)^2 + (x_2 - 1)^4$ $f_2(\vec{x}) = (x_1 + 1)^2 + (x_2 + 1)^2$	$x_{1,2} \in [-3,\ 3]$ $n = 2$
TEST2	$f_1(\vec{x}) = (x_1 - 1)^4 + (x_2 - 1)^2 + (x_3 - 1)^2$ $f_2(\vec{x}) = (x_1 + 1)^2 + (x_2 + 1)^4 + (x_3 + 1)^2$ $f_3(\vec{x}) = (x_1 - 1)^2 + (x_2 + 1)^2 + (x_3 - 1)^4$	$x_{1,2,3} \in [-5,\ 5]$ $n = 3$
CP3	$f_1(\vec{x}) = (1 + x_3)(x_1^3 x_2^2 - 10x_1 - 4x_2)$ $f_2(\vec{x}) = (1 + x_3)(x_1^3 x_2^2 - 10x_1 + 4x_2)$ $f_3(\vec{x}) = 3(1 + x_3)x_1^2$	$1 \le x_1 \le 3.5$ $-2 \le x_2 \le 2$ $0 \le x_3 \le 1$
DLZT	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$ $f_3(\vec{x}) = 3.5 - \sum_{i=1}^{n} 2x_i sin(n\pi x_i)$	$x_i \in [0,\ 1]$ $n = 3$ $i = 1, 2, \ldots, n$
GSPm	$f_1(\vec{x}) = (1 + x_m^2)\cos(x_1\pi/2)\cdots\cos(x_{m-1}\pi/2)$ $f_2(\vec{x}) = (1 + x_m^2)\cos(x_1\pi/2)\cdots\sin(x_{m-1}\pi/2)$ $\vdots$ $f_{m-1}(\vec{x}) = (1 + x_m^2)\cos(x_1\pi/2)\sin(x_2\pi/2)$ $f_m(\vec{x}) = (1 + x_m^2)\sin(x_1\pi/2)$	$x_i \in [0,\ 1]$ $n = m$ $i = 1, \ldots, m-1$ $x_m \in [-1,\ 1]$

2.5.1 Cardinality-based Performance Metrics

In the case of Cardinality-based performance metrics, the quality of a non-dominated set is assessed from counting the number of non-dominated solutions. Indeed, the simplest cardinality-based metric introduced in [Vel99] is to count the number of non-dominated solutions, i.e., $|A|$. This can also be achieved by calculating the ratio $|A|/|R|$. Of course, computing this ratio requires prior knowledge of the set R. If we know a Pareto-optimal set R, then we can compute *Error Ratio (ER)* introduced by [Vel99] as follows:

$$ER(A,R) = (\sum_{i=1}^{|A|} e_i)/|A| \tag{2.4}$$

where $e_i = 0$, if the solution $a_i \in A$ is in R, and 1 otherwise. Lower values of ER represent better non-dominated sets.
Another cardinality-based metric is to compute a ratio C_r of the solutions in A, which are not dominated by R [HJ98]:

$$C_r(A,R) = \frac{|\{a \in A | \nexists r \in R : r \prec a\}|}{|A|} \tag{2.5}$$

These performance metrics have several drawbacks. Knowledge of R is required and no information is delivered about the accuracy and distribution of solutions. However, they can be used, for test problems, as a quick and rough means of assessing progress towards the Pareto-optimal set R during generations.

2.5.2 Convergence-based Performance Metrics

Convergence-based performance metrics compute the accuracy of the non-dominated solutions. The accuracy demonstrates how closely the obtained solutions have converged to the Pareto-optimal front. In the following, different methods of this group are briefly reviewed.

Generational Distance (GD)

This method [Vel99, VL00] computes the average distance from the obtained non-dominated set A and a Pareto-optimal set R as follows:

$$GD(A,R) = \left(\sqrt{\sum_{i=1}^{|A|} {d_i}^2} \right) /|A| \tag{2.6}$$

where d_i is the distance in objective space between the solution $a_i \in A$ and the *nearest* member of R. Lower values of GD represent better convergence of solutions. This method requires knowledge of R and cannot be used confidently for non-dominated sets that are changing in cardinality.
Seven Point Average Distance (SPAD) [Sch95] and *Maximum Pareto Front Error (MPFE)* [Vel99] are also other accuracy metrics, which are based on calculating the distance between a Pareto-optimal set and a non-dominated set. However, all of these methods require prior knowledge of a Pareto-optimal set R, which may not be trivial for some real-world applications. These methods can also be influenced by the spread and distribution of R.

Size of dominated space (S metric)

This metric [Zit99] measures the hyper-volume of a region made by the set A and a *reference point* in the objective space. Figure 2.3 shows two sets A_1 and A_2 and a reference point Ref. The set A_1 has a good diversity, therefore the defined region

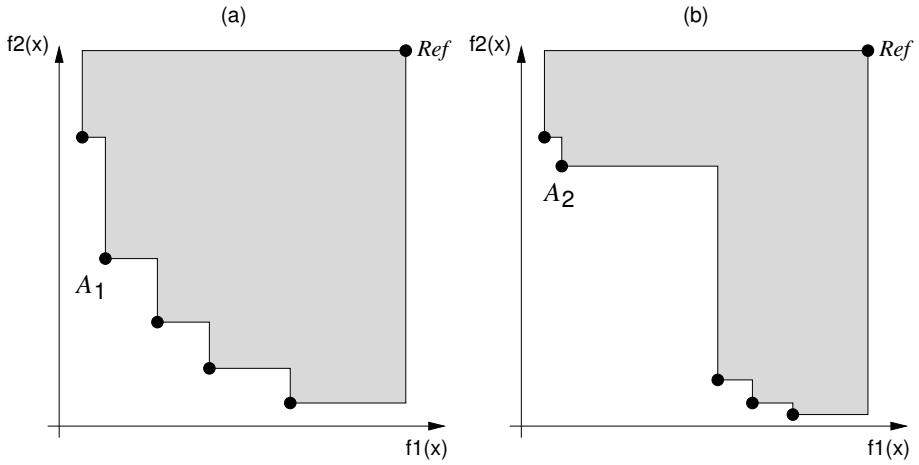


Figure 2.3: S metric (a) Set A_1 (b) Set A_2

has a larger area than the area made by the set A_2.
However, this measurement depends on the position of Ref. This method requires a high computational effort when the number of objectives increases. The result of this measurement is a hyper-volume value which gives us information about the region between the solutions and the reference point. This metric is used when we compare the diversity (and also the convergence) of two non-dominated sets. Therefore, we do not obtain knowledge of the non-dominated front itself.
A simpler way to calculate the hyper-volume of a 2-dimensional non-dominated set is proposed by Dunn and Olague in [DO04]. The computation of the hyper-volume is

simplified by calculating the area of each triangle formed by two adjacent elements of the non-dominated set and the reference point. The function values are normalized in such a way that the maximal total area is 1. Figure 2.4 illustrates this method.

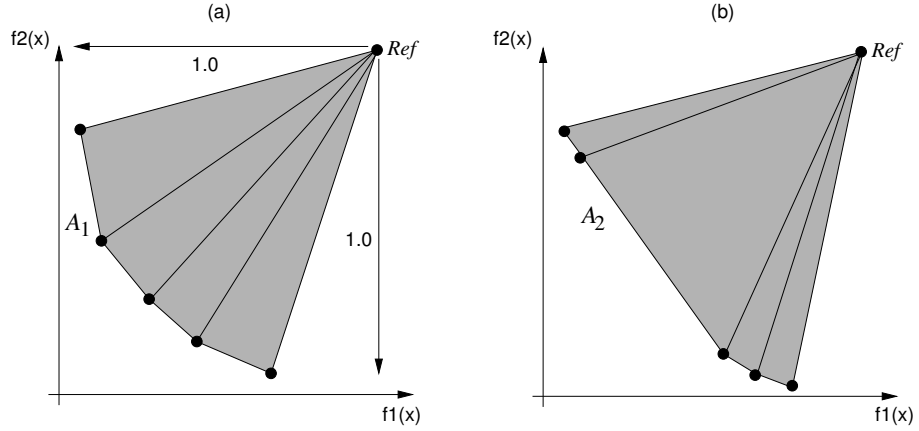


Figure 2.4: Computing the hyper-volume of a two-objective non-dominated set proposed by Dunn and Olague [DO04].(a) Set A_1 (b) Set A_2

This method is efficient when the two non-dominated fronts have similar spread and distribution of solutions. This can also be seen in Figure 2.4.
A modified way of calculating the hyper-volumes is introduced by Fleischer [Fle03]. Fleischer's Algorithm (LebMeasure Algorithm) is a simple and efficient method for computing the hyper-volume of any set of non-dominated solutions. The algorithm is as follows: From a non-dominated set of solutions, we can find a dominated region that does not intersect with other dominated regions, e.g., the region A in Figure 2.5. This region can be easily identified and lopped off. Moreover, the hyper-volume of such a non-dominated region is trivial to compute, as it is a rectangular polytope. In the next step, other such regions are lopped off repeatedly until no region is left. The hyper-volume is the sum of hyper-volumes of the lopped off regions. This method can be easily computed for any number of objectives.
The hyper-volume measurement requires defining an upper boundary of the region within which all feasible solutions lie. This is defined arbitrarily and can cause different ordering of non-dominated sets. It also has a large computational time overhead, $O(|A|^{m+1})$, where m is the number of objectives. The output of this measurement does not have any units and therefore, it can be used in comparing two sets of solutions.

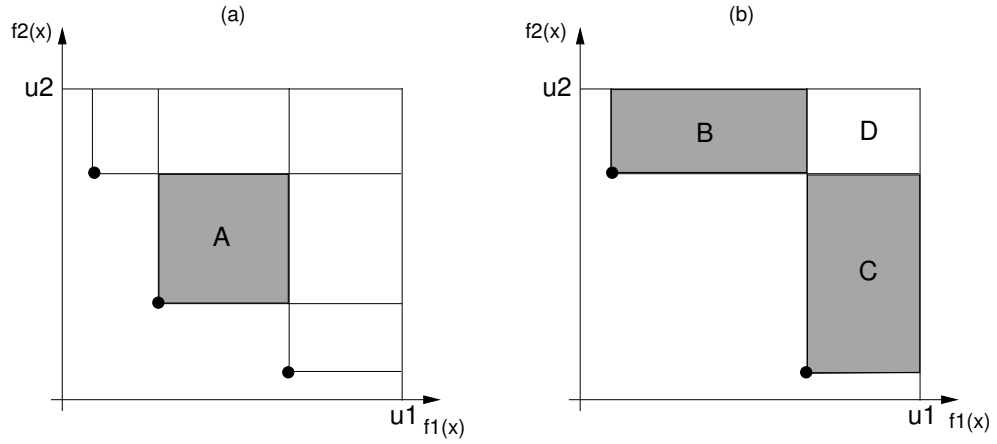


Figure 2.5: Computing the hyper-volume of a two-objective non-dominated set using Fleischer's Algorithm. In the first step, region A is lopped off. The regions B, C, and D are lopped off in the next steps, respectively. u1 and u2 are the upperbounds of each of the objectives.

Convergence of two sets (C metric)

This metric [Zit99] is aimed to compare the convergence of two non-dominated sets A_1 and A_2 as follows:

$$C(A_1, A_2) = \frac{|\{a_2 \in A_2 | \exists a_1 \in A_1 : a_1 \preceq a_2\}|}{|A_2|} \tag{2.7}$$

The value of $C(A_1, A_2) = 1$ means that all the members of A_2 are weakly dominated by the members of A_1. We can also conclude that $C(A_1, A_2) = 0$ means that none of the members of A_2 is weakly dominated by the members of A_1. However, $C(A_1, A_2)$ is not equal to $1 - C(A_2, A_1)$, and we have to consider both $C(A_1, A_2)$ and $C(A_2, A_1)$ for comparison.
This metric has low computational time compared to the S metric. It requires no information about a Pareto-optimal front or a reference point and for two evenly-distributed sets of the same cardinality gives results compatible with the intuitive notation of quality. However, if the sets are of different cardinality or the distribution of the sets are non-uniform, then it gives unreliable results.

2.5.3 Diversity-based Performance Metrics

These methods compute distribution and spread of the solutions along the obtained non-dominated front. Most of these methods consider the objective space.
One possible way to compute the distribution of solutions is to calculate the distances

between the non-dominated solutions. Deb et al. [DAPM00] proposed a method to measure the distribution of non-dominated solutions along the approximated front. In this method, we have to compute the Euclidian distance d_i between the solutions in A and then calculate their average as $\bar{d}$. Then, the distribution metric $\Delta_s^{'}$ is as follows:

$$\Delta_s^{'} = \sum_{i=1}^{|A|-1} \frac{|d_i - \bar{d}|}{|A| - 1} \tag{2.8}$$

This method is able to compute the distribution of solutions in 2-objective spaces. However, it cannot be used for MOPs with more than two objectives, as consecutive sorting is involved. This method is modified by Deb et al. in [Deb01, DAPM00], which also considers the spread of solutions. Also, it works only for 2-objective MOPs.
Another method called *Spacing* is introduced by Schott [Sch95]. This method is based on computing the shortest distances between the non-dominated solutions along each axis. In the cases that solutions are gathered in small groups along the non-dominated front, the distances between the groups are not considered, because only the shortest distances are computed. In this case, this method may be misleading. Also, there are methods for computing the spread and distribution of solutions such as *Maximum Spread* [ZDT00], *Chi-Square-Like Deviation* [Deb01], and *Uniform Distribution* [TLK01]. In the following, two recent diversity metrics, which outperform the previous diversity measurements, are briefly reviewed.

Entropy approach

This method [FMA02] can quantitatively assess the distribution quality of a solution set. Each solution provides some information about its neighborhood in the feasible region. This can be modeled as a function, called the *influence function*. Each individual i will have an influence function Ω_i, which is defined in the objective space, i.e., $\Omega_i : \mathbb{R}^m \rightarrow \mathbb{R}$. This function must be a decreasing function of the distance to the ith individual, e.g., a Gaussian function. Figure 2.6 shows Gaussian functions for several individuals in a one-dimensional domain [FMA02]. In the next step, a density function is defined as an aggregation of the influence functions (Figure 2.6). The sharp peaks of the density function correspond to the high-density areas, and the deep valleys to low-density areas, respectively. Indeed, a flat density function shows well-distributed solutions. An *entropy metric* is used for calculating the flatness of the density function. A non-dominated set with well-distributed solutions has a higher entropy value than other non-well-distributed ones. To apply the entropy

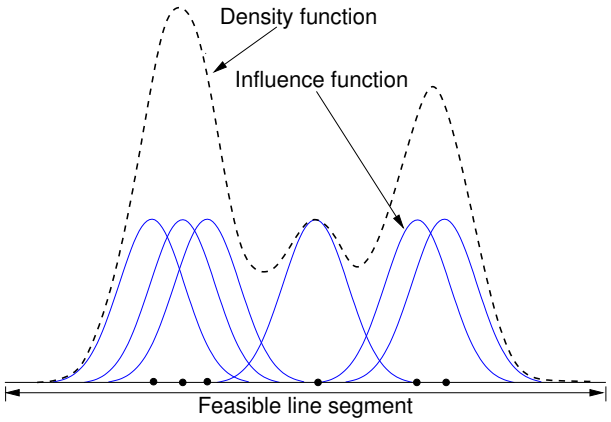


Figure 2.6: Entropy approach [FMA02]

measurement, the solutions of the non-dominated set must be projected on a hyper-plane. The Gram-Schmidt Orthogonalization is applied to find a hyper-plane for mapping the solutions from m-dimensional space to m-1–dimensional space. Then, the entropy measurement is applied to the solutions. The higher the entropy, the better the distribution of the solutions achieved. This diversity measurement is applied on 3-objective test functions and also suggested for m–dimensional objective spaces by using *hypersurface density functions*. The entropy measure has two major applications: The first being a diversity measurement, and the second as a stopping criterion, by applying it at the end of each generation to the non-dominated solutions.
This method does not find the real distribution of solutions, because it projects the solutions from m–dimensional space to m-1–dimensional space. The real distribution of solutions on approximated fronts is distorted by applying the Gram-Schmidt orthogonalization (refer to [FMA02, OJS03] for more details).

Sparsity measure

This method [DMM03a] is similar to the entropy approach. The non-dominated solutions are first projected on a hyper-plane with the unit normal vector η. A hyper-box of a certain size d is centered around each projected solution. Figure 2.7 shows an example of the sparsity measure for a 3-objective space. The total hyper-cube covered by hyper-boxes is a measure of the sparsity of solutions. When the solutions are not well-distributed, the hyper-boxes overlap and therefore the total covered area of the hyper-cube is less than when there are no overlaps. In this measurement the parameter d is playing an important role as well as η. Like the entropy measure, this method cannot find the exact distribution of the solutions

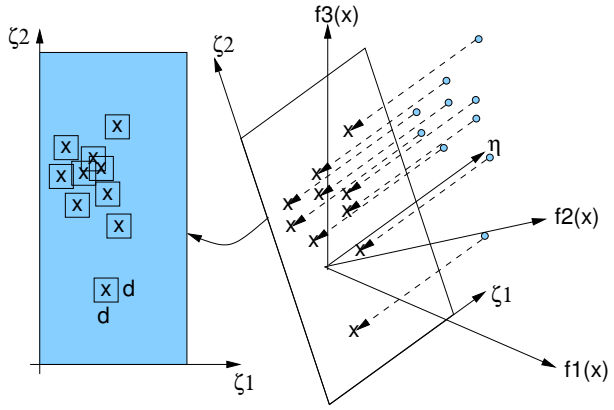


Figure 2.7: Sparsity measure. The solutions are projected on the hyper-plane with coordinate axis of ζ_1 and ζ_2 [DMM03a]

because of the projection of solutions.

Discussion

The above measurements on convergence and diversity of the approximated Pareto-optimal solutions provide very good information, especially when comparing two approximated sets. However, a hyper-volume value or an entropy measure of solutions does not contain any information about the front itself. There are still some open questions, e.g., what percentage of the objective space is occupied by the non-dominated solutions? Where are the solutions concentrated? The answers to these questions are particularly valuable for high dimensional fronts (e.g., for 4- and higher-objective spaces). Also projecting the non-dominated solutions in the entropy approach and Sparsity measure into hyper-planes makes it very difficult to find the exact diversity measure of non-dominated sets.

2.6 Conclusion

In this chapter, the basic concepts of MOEA, test functions and comparison metrics, which are also used in the thesis, are reviewed. It must be emphasized that only those EA and MOEA methods that are referred to in the thesis are explained here. The background of the basic notations in EAs and the MOEA methods, especially the SPEA2 method, are reviewed. For comparison purposes, SPEA2 is used several times in the thesis.
Also, different test functions with various degrees of difficulties are introduced. Indeed, the task of the test functions is to test the abilities of the investigated methods.

Chapter 3

Data Structures for Storing Non-dominated Solutions

The existence of an archive for storing non-dominated solutions is important for the convergence to the Pareto-optimal front (see Section 2.3). The reason is that we must not lose the best solutions obtained so far. On the other hand, having the archive members participate in the selection process increases the probability of achieving better offsprings. Therefore, we never lose the non-dominated solutions until a new solution enters the archive and dominates any of them.
In elitist MOEAs, the data structures for storing and updating archives may have a great impact on the required computational time, especially when optimizing high dimensional problems with large non-dominated sets. In this chapter, different data structures for storing non-dominated solutions, such as Quad-trees and dominated tree data structures, are studied as alternatives to linear lists.
In the following, we assume that at any generation t, the archive A_t must always be kept *domination-free*, i.e., the archive members do not dominate each other.

3.1 Linear Lists

A linear list is the most straightforward way to implement an archive. In order to keep an archive domination-free, the following basic operation must be performed, namely *Insertion with Update* of complexity $O(m \cdot |A| \cdot |P|)$: In the worst case, each candidate $\vec{x}$ of a population P has to be tested against each member of the archive A for inclusion. In case $\vec{x}$ is not dominated by any member of A, we assume it is either inserted (e.g., at the end), or else rejected. On the other hand, if $\vec{x}$ dominates members of A, these members must be deleted from the archive. Hence, the overall run time complexity of maintaining a domination-free linear list is $O(m \cdot |A| \cdot |P|)$.

In a MOEA such as SPEA [ZT99], non-dominated solutions (objective vectors) are stored in an array. In this archive, if a candidate $\vec{x}$ is not dominated with respect to other members of P and also if it does not dominate any vector in the archive A, it is added to the end of the array. On the other hand, if the new vector $\vec{x}$ is dominated by another vector $\vec{y} \in A$, then $\vec{x}$ is rejected. If $\vec{x}$ dominates $\vec{y}$, then $\vec{y}$ is deleted.

3.2 Dominated and Non-dominated Trees

Dominated and non-dominated tree data structures are proposed by Fieldsend et al. [FES02, Fie03] for storing queries and updating the archive through several generations. These data structures are developed to store all the non-dominated solutions in MOEAs, arguing that this improves the convergence of the MOEA. Therefore, these data structures must store a large number of solutions. The dominated and non-dominated trees are constructed in the first generation, before inserting a new solution into the archive:

- The dominated tree is used to check if the new candidate solution for insertion is dominated by the archive members.
- The non-dominated tree is used to check if the archive members are dominated by the new candidate solution for insertion.

In the following, construction of dominated and non-dominated trees are briefly explained. For more details please refer to [FES02, Fie03].

Dominated Tree

The dominated tree consists of a list τ of $L = \lceil |A|/m \rceil$ *composite points* $\vec{c}_1, \ldots, \vec{c}_L$ ordered by a weak dominance relation, where $|A|$ is the cardinality of the archive and m is the number of objectives:

$$\tau = \{\vec{c}_L \preceq \cdots \preceq \vec{c}_2 \preceq \vec{c}_1\} \tag{3.1}$$

In the dominated tree, each composite point is a vector of m elements and each element is constructed from the archive $A = \{\vec{y}_j | 1 \leq j \leq |A|\}$ as follows: The first composite point $\vec{c}_1$ is constructed by finding the solution $\vec{y}_j \in A$ with maximum first coordinates; this value forms the first coordinate of the composite point:

$$c_{1,1} = \max_{\vec{y}_j \in A}(y_{j,1}) \tag{3.2}$$

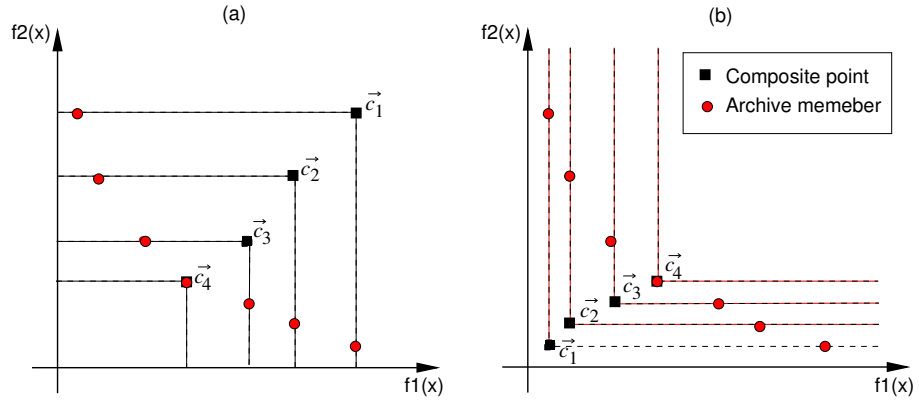


Figure 3.1: Example of (a) dominated and (b) non-dominated trees

The solution $\vec{y}_j$, is now associated with $\vec{c}_1$ and is not considered anymore. Likewise, the second coordinate of $\vec{c}_1$ is given by the maximum second coordinates of the remaining solutions in A:

$$c_{1,2} = \max_{\vec{y}_j \in A \setminus \tau} (y_{j,2}) \tag{3.3}$$

This procedure is repeated to construct subsequent composite points until all elements of A are associated in the tree. In general, the dth coordinate of the ith composite point is given by:

$$c_{i,d} = \max_{\vec{y}_j \in A \setminus \tau} (y_{j,d}) \tag{3.4}$$

Figure 3.1 (a) shows this method for a two-objective example, where $\vec{c}_1, \ldots, \vec{c}_4$ are composite points.

Non-dominated Tree

The non-dominated tree is constructed in the same way as a dominated tree. It consists of a list of $L = \lceil |A|/m \rceil$ composite points ordered by a weak dominance relation:

$$\tau = \{\vec{c}_1 \preceq \vec{c}_2 \preceq \cdots \preceq \vec{c}_L\} \tag{3.5}$$

The construction proceeds by starting with the minimum coordinate of each objective and successively inserting points, with a minimum coordinate remaining in each dimension. An example of a non-dominated tree is shown in Figure 3.1 (b).

Inserting a new solution is achieved in two phases:

1. In the first phase, dominated trees are used to check if a new solution $\vec{q}$ is dominated by an archive member. First, the list τ is searched to find indices h and l. $\vec{c}_h$ dominates $\vec{q}$ and $\vec{c}_l$ is dominated by $\vec{q}$. Then, the least composite point that dominates $\vec{c}_h$ must be found and is named $\vec{c}_H$. Since $\vec{c}_h \prec \vec{q}$, it is clear that all the solutions making the composite point $\vec{c}_i$ ($\vec{c}_i \prec \vec{c}_h$), also dominate $\vec{q}$. Also, since $\vec{q} \prec \vec{c}_l$ and the solutions making the composite point $\vec{c}_l$ have at least one coordinate equal to a coordinate of $\vec{c}_l$, it may be concluded that $\vec{q}$ is not dominated by any solutions making the composite points $\vec{c}_1, \ \ldots, \ \vec{c}_l$. Therefore, $\vec{q}$ must be checked by the solutions making the composite points with indices i, $l < i < H$ to determine whether it is dominated by them or not.

2. In the second phase, the non-dominated tree is used to determine which archive members are dominated by the new solution $\vec{q}$. It is clear that when $\vec{q}$ dominates a composites point $\vec{c}_i$, all the solutions making the composite points $\vec{c}_{i+1}, \ \ldots, \ \vec{c}_L$ are also dominated and candidates for deletion.

If the new solution is dominated by at least one of the archive members, it must be discarded. In the case that it is not dominated, it must be inserted into the archive and also in an appropriate place in the dominated tree. The dominated members of the archive must be deleted from the archive as well. The insertion and deletion processes in dominated trees are as follows.

Insertion

Suppose that the new solution $\vec{y}$ should be inserted into the dominated tree τ:

1. If $\vec{y} \prec \vec{c}_L$, then the corresponding composite point of $\vec{y}$, $\vec{c}\,'$ (weakly) dominates $\vec{c}_L$ and should be appended to τ as $\vec{c}_{L+1}$.

2. If $\vec{y} \not\prec \vec{c}_L$, a bisection search is used to locate a new composite point corresponding to $\vec{y}$. The bisection search is done in the dominated tree such that $\vec{c}_{j+1} \not\prec \vec{y}$ and $\vec{y} \preceq \vec{c}_j$. Then, the new composite point $\vec{c}\,'$ can be constructed as follows.

$$\vec{c_k}\,' = \max_{k=1}^{m}(y_k, c_{j,k}) \tag{3.6}$$

$\vec{c}\,'$ should be inserted in τ between $\vec{c}_{j+1}$ and $\vec{c}_j$.

Deletion

Deletion of an archive member $\vec{y}$ is more complicated than insertion, since the corresponding composite point of it, let's say $\vec{c}_j$, must still remain (weakly) dominated by $\vec{c}_{j+1}$ after deletion. If $\vec{y}$ defines the kth coordinate of $\vec{c}_j$, then upon deletion of $\vec{y}$, the deleted coordinate of $\vec{c}_j$ must be replaced by the kth coordinate of $\vec{c}_{j+1}$. Therefore, $c_{j,k} \leq c_{j+1,k}$ and $\vec{c}_{j+1} \preceq \vec{c}_j$.

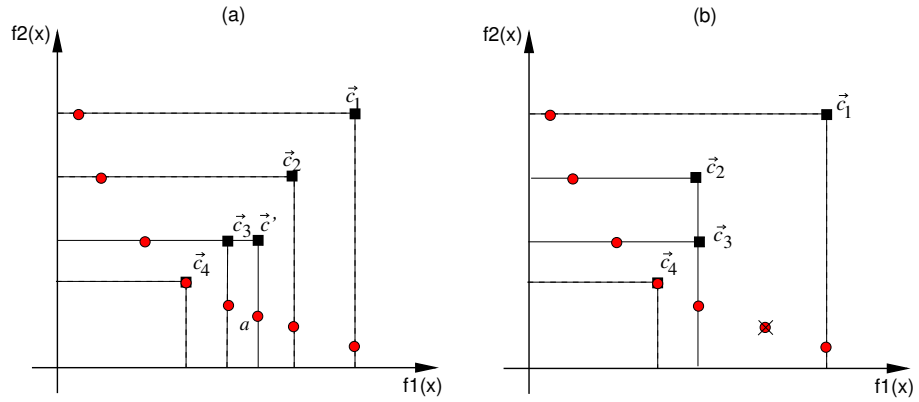


Figure 3.2: Dominated tree (a) inserting a new solution a needs to create the composite point $\vec{c}\,'$, Deletion is illustrated in (b)

Figures 3.2 (a) and (b) show examples of insertion and deletion. It can be observed that there are some archive members contributing to more than one composite points, after insertion and deletion. Indeed, the dominated tree contains more composite points than necessary ($|A|/m$). Therefore, the dominated tree must be cleaned by the *cleaning* process after each insertion and deletion. In practice, it is not efficient to clean the tree after each insertion and deletion, it can be done occasionally, when the number of composite points exceed a certain threshold [FES02].

Discussion

In MOEAs, the archive is represented by one dominated and one non-dominated tree data structure. The computational complexities for constructing the dominated tree, insertion, deletion, and cleaning are estimated as $O(|A|.m)$, $O(log_2(|A|/m))$, $O(km)$, and $O(Lm - |A|)$, respectively (refer to [FES02]), where k is the number of composite points to which the deleted solution has contributed. However, to gain a more realistic idea of the performance of these data structures in MOEA, Fieldsend et al. have carried out different experiments [FES02, Fie03]. The experiments show that these data structures consume particularly low computational time for large number of generations and very low number of individuals in the population. This

has been investigated for different 2-objective test functions. Indeed, the population size has a great impact on the computational time.

3.3 Quad-trees

A Quad-tree [Hab83, SS96b] is a tree-based data structure for storing objective vectors. Each node is a vector with m elements and can have at most 2^m sons (branches), which are defined by a successorship. A Quad-tree is a domination-free data structure. In order to explain this data structure, the following definitions are necessary. Let $\vec{x}$ and $\vec{y}$ denote m-dimensional objective vectors.

Definition 3.1 (k-Successor) *A node $\vec{x}$ is called k-Successor of node $\vec{y}$ where k is a binary string*[1] *of the form $k = (k_1, \ldots, k_m)_2$ and*

$$k_i = \begin{cases} 1 & \text{if } x_i \geq y_i \\ 0 & \text{if } x_i < y_i \end{cases} \tag{3.8}$$

Definition 3.2 (k-Son) *Node $\vec{x}$ is k-Son of node $\vec{y}$, if $\vec{x}$ is a k-Successor of $\vec{y}$ and also the direct son of $\vec{y}$.*

Definition 3.3 (k-Set) *$S_j(k)$ denotes the set of positions k_i, where $k_i = j$*

$$S_0(k) = \{i | k_i = 0, k = (k_1, k_2, \ldots, k_m)_2\} \tag{3.9}$$

$$S_1(k) = \{i | k_i = 1, k = (k_1, k_2, \ldots, k_m)_2\} \tag{3.10}$$

Since a Quad-tree is a domination-free data structure, there are no branches with the successorship $(00\ldots0)$ and $(11\ldots1)$, because nodes with $k = (00\ldots0)$ will by definition dominate the root and the nodes with $k = (11\ldots1)$ will always be dominated by the root.
Nodes with the above properties may be graphically represented by a tree structure.

Example 1:

Figure 3.3 shows an example of a tree for storing non-dominated 3-objective vectors. Each node of this tree has $m = 3$ elements. Therefore, each node can have at most

[1] k can be considered as a scalar value [Hab83, SS96b] as follows:

$$k = \sum_{i=1}^{m} k_i 2^{m-i} \tag{3.7}$$

8 sons. This tree is not domination-free and hence not a Quad-tree, because the root (10 10 10) has a (000)-Son and a (111)-Son. The (000)-Son of the root, (5 5 5), dominates the root and the (111)-Son of the root, (12 15 18), is dominated by the root.

□

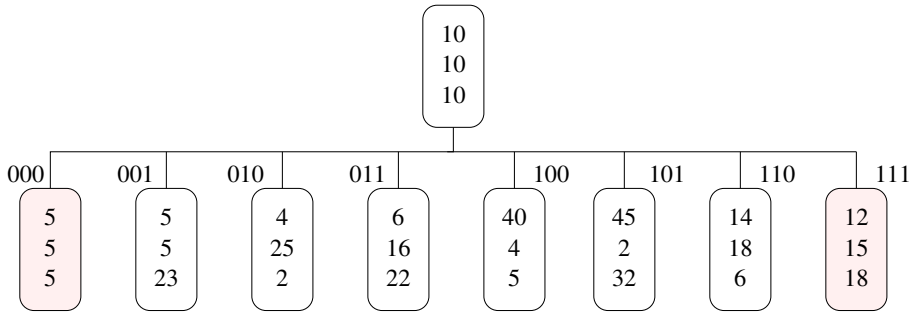


Figure 3.3: Example of a tree-structure for storing non-dominated solutions. Node (5 5 5) dominates the root (10 10 10) and node (12 15 18) is dominated by the root.

Example 2:

Consider the following k-Successor: $k = (111001101)$. The corresponding k-Sets are $S_1(k) = \{1, 3, 4, 7, 8, 9\}$ and $S_0(k) = \{2, 5, 6\}$.

□

When processing a vector for possible inclusion into a domination-free Quad-tree, the vector is either discarded as being dominated by a node already in the Quad-tree, or it is inserted into the Quad-tree. However, when a vector is inserted, it may dominate other nodes in the Quad-tree. Then, these nodes must be deleted as well. By deleting a node, we obviously destroy the structure of the subtree rooted at the deleted node. This means that all the successors of the deleted node must be reconsidered for inclusion in the Quad-tree. The corresponding algorithm is shown in Algorithm 3.
The way in which vectors are processed for possible inclusion in the Quad-tree in order to maintain it domination-free is very important. Suppose that we are processing $\vec{x} \in \mathbb{R}^3$ for possible inclusion into a domination-free Quad-tree rooted at $\vec{y} \in \mathbb{R}^3$, and such that $\vec{x}$ is a k-Successor of $\vec{y}$. With the definitions of $S_0(k)$ and $S_1(k)$, observe that:

- The only places in the Quad-tree where there may be vectors that dominate $\vec{x}$ are in those subtrees whose roots are sons of $\vec{y}$, and have *zeros* in at least

Algorithm 3 : Quad-tree1

Input: $\vec{x}$ to be inserted into a Quad-tree rooted at $\vec{y}$
Output: Updated (domination-free) Quad-tree

1. Start: Let $\vec{y}$ be the root of the tree
2. Successorship: Calculate k such that $\vec{x}$ is the k-Successor of $\vec{y}$
if $k = (11\ldots1)$ or $x_i = y_i, \forall i \in S_0(k)$ **then**
  STOP
else if $k = (00\ldots0)$ **then**
  delete $\vec{y}$ and its subtree {$\vec{y}$ is dominated by $\vec{x}$}
end if
3. Dominance test 1: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $l < k$ and $S_0(k) \subset S_0(l)$, Execute TEST1($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ is dominated by $\vec{z}$ or a son of $\vec{z}$}
4. Dominance test 2: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $k < l$ and $S_1(k) \subset S_1(l)$, Execute TEST2($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
5. Insertion:
if a k-Son of $\vec{y}$ already exists **then**
  replace $\vec{y}$ by the k-Son and goto Step 2
else
  $\vec{x}$ is the k-Son of $\vec{y}$ and STOP
end if

Algorithm 4 : TEST1($\vec{x}$,$\vec{z}$)

TEST1($\vec{x}$,$\vec{z}$): {Check if $\vec{x}$ is dominated by $\vec{z}$ or a son of $\vec{z}$}
Calculate k such that $\vec{x}$ is the k-Successor of $\vec{z}$
if $x_i = z_i, \forall i \in S_0(k)$ **then**
  STOP
end if
for all $\vec{v}$, such that $\vec{v}$ is a l-Son of $\vec{z}$ and $S_0(k) \subset S_0(l)$, Execute TEST1($\vec{x}$,$\vec{v}$)

Algorithm 5 : TEST2($\vec{x}$,$\vec{z}$)

TEST2($\vec{x}$,$\vec{z}$): {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
Calculate k such that $\vec{x}$ is the k-Successor of $\vec{z}$
if $k = 0$ **then**
  delete $\vec{z}$ and reinsert all its subtrees from the global root
end if
for all $\vec{v}$, such that $\vec{v}$ is a l-Son of $\vec{z}$ and $S_1(k) \subset S_1(l)$, Execute TEST2($\vec{x}$,$\vec{v}$)

all locations that k does. That is the subtree rooted at l-Sons of $\vec{y}$ for which $l < k$ and $S_0(k) \subset S_0(l)$. For example, in the case that $k = (110)$, only the $l = (010)$- and (100)-Successors of $\vec{y}$ must be checked whether they dominate $\vec{x}$ or not.

- The only places in the Quad-tree where there may be vectors that are dominated by $\vec{x}$ are in those subtrees whose roots are sons of $\vec{y}$, and have *ones* in at least all locations that k does. That is the subtree rooted at l-Sons of $\vec{y}$ for which $l > k$ and $S_1(k) \subset S_1(l)$. For example, in the case that $k = (100)$, only the $l = (101)$- and (110)-Successors of $\vec{y}$ must be checked whether they are dominated by $\vec{x}$ or not.

An objective vector is admitted to a Quad-tree if and only if it is not dominated by any of the vectors already in the Quad-tree. Moreover, when admitted to the Quad-tree, all objective vectors in the Quad-tree dominated by the new entry must be identified and deleted.

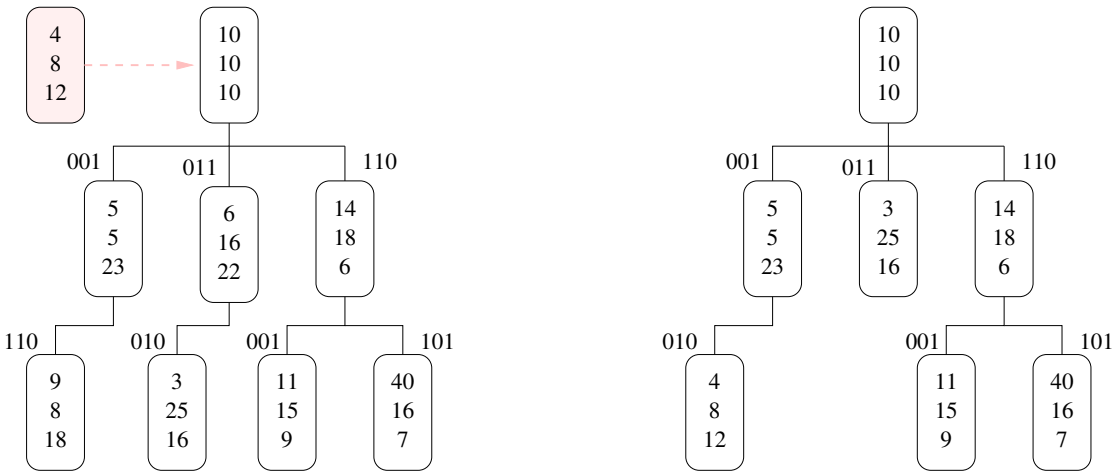
Example 3:

Figure 3.4 (a) shows a Quad-tree in which we want to insert the new vector (4 8 12). According to the Algorithm 3, the k-Successorship of (4 8 12) to (10 10 10) is calculated in Step 2 as $k = (001)$. In the next step (Step 4), we must look in the (011)- and (101)-Successors of (10 10 10) for nodes that are dominated by (4 8 12). The node (6 16 22) is dominated by (4 8 12) and must be deleted. After deleting (6 16 22), we have to reinsert all the nodes in the subtree of (6 16 22), again from the root (10 10 10). Here, (3 25 16) is reinserted as the (011)-Successor of (10 10 10), (see Figure 3.4 (b)). Then, we go to step 5 to insert (4 8 12) as a (001)-Son of (10 10 10). Since (10 10 10) has already a (001)-Son namely (5 5 23), we consider (5 5 23) as the new root in the algorithm (Step 5) and then try to insert (4 8 12) in its subtree. (4 8 12) is the (010)-Successor of (5 5 23), therefore the (011)- and (110)-Successors of (5 5 23) must be checked whether they are dominated by (4 8 12) or not. (9 8 18) is indeed dominated by (4 8 12), so it must be deleted. Since (5 5 23) has no other sons, we insert (4 8 12) as its (010)-Successor. Figure 3.4 (b) shows the Quad-tree after insertion.

□

According to the above algorithm, the data structure Quad-tree1 has two disadvantages:

1. The form and depth of the Quad-tree depends on the order of insertion of a given set of vectors.



(a) Before insertion

(b) After insertion

Figure 3.4: Insertion in a Quad-tree (Example 3)

2. Deleting a node destroys the structure of the Quad-tree and requires the reinsertion of all of the nodes of the subtrees of the deleted node. Therefore, deletions combined with reinsertions will be time-critical and one should carefully avoid having to reinsert a vector multiple times.

Hence, the following variants of the Quad-tree data structure (Quad-tree2, Quad-tree3) are proposed, in which the deletion process is improved [MTT02, MT04b].
In **Quad-tree2**, a discovered dominated node is not deleted immediately, it is marked as deleted by a flag. Then its subtrees are traversed again, setting flags for all encountered dominated nodes instead of immediately reinserting all subtrees. After finishing this recursive descent, only those nodes in the subtrees are reinserted that are not marked deleted. The algorithm called Quad-tree2 is shown in Algorithm 6 and Example 4 illustrates it.

Example 4:

Consider Figure 3.4 (b) and let us tentatively insert the new vector (12 15 5). (12 15 5) is the (110)-Successor of the root (10 10 10). Since (10 10 10) already has a (110)-Son, (14 18 6), we consider (14 18 6) as the new root in the algorithm (Step 5) and try to insert (12 15 5) in its subtree. (12 15 5) is the (000)-Successor of (14 18 6). It means that (14 18 6) must be deleted. Contrary to algorithm Quad-tree1, we do not delete it and reinsert all nodes of corresponding subtrees immediately. Instead,

Algorithm 6 : Quad-tree2

Input: $\vec{x}$ to be inserted into a Quad-tree rooted at $\vec{y}$
Output: Updated (domination-free) Quad-tree

1. Start: Let $\vec{y}$ be the root of the tree
2. Successorship: Calculate k such that $\vec{x}$ is the k-Successor of $\vec{y}$
if $k = (11\ldots1)$ or $x_i = y_i, \forall i \in S_0(k)$ **then**
STOP
end if
if $k = (00\ldots0)$ $\{\vec{y}$ is dominated by $\vec{x}\}$ **then**
Flag = 0
MARK($\vec{x}$, $\vec{y}$)
Replace $\vec{y}$ by $\vec{x}$. Delete all the marked nodes in the subtree of $\vec{x}$
Reinsert the nodes with the reinserted mark in subtrees of $\vec{x}$ at $\vec{x}$
STOP
end if
3. Dominance test 1: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $l < k$ and $S_0(k) \subset S_0(l)$, Execute TEST1($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ is dominated by $\vec{z}$ or a son of $\vec{z}$}
4. Dominance test 2: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $k < l$ and $S_1(k) \subset S_1(l)$, Execute TEST3($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
5. Insertion:
if a k-Son of $\vec{y}$ already exists **then**
replace $\vec{y}$ by the k-Son and goto Step 2
else
Place $\vec{x}$ as the k-Son of $\vec{y}$. Delete all the marked nodes in the Quad-tree. Reinsert the nodes with the reinserted mark from the global root. STOP
end if

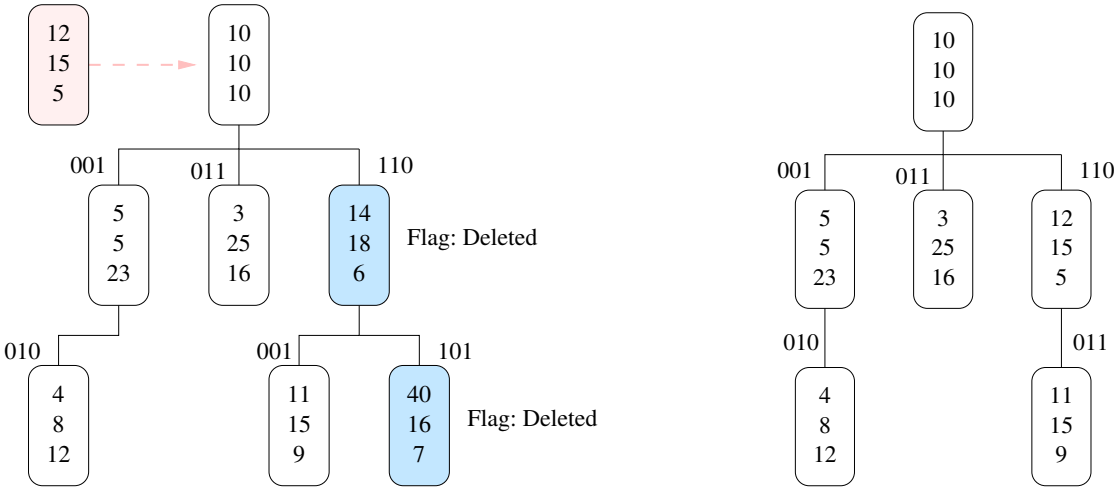
Algorithm 7 : MARK($\vec{x}$,$\vec{z}$)

MARK($\vec{x}$,$\vec{z}$):
for all sons $\vec{z}$ of $\vec{y}$
if $\vec{z}$ is a $(00\ldots0)$-Son of $\vec{x}$ **then**
mark $\vec{z}$ as deleted and Flag = 1
else if Flag = 1 **then**
mark $\vec{z}$ as reinserted
end if
MARK($\vec{x}$, $\vec{z}$)

Algorithm 8 : TEST3($\vec{x}$,$\vec{z}$)

TEST3($\vec{x}$,$\vec{z}$): {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
Calculate k such that $\vec{x}$ is the k-Successor of $\vec{z}$
if $k = 0$ **then**
  mark $\vec{z}$ as deleted and Flag $= 1$
else if Flag $= 1$ **then**
  mark $\vec{z}$ as reinserted
end if
for all $\vec{v}$, such that $\vec{v}$ is a l-Son of $\vec{z}$ and $S_1(k) \subset S_1(l)$, Execute TEST3($\vec{x}$,$\vec{v}$)

we only mark it as deleted. Then we look for other dominated nodes in its subtrees. (40 16 7) is also dominated by (12 15 5) and must be deleted. Therefore, we also mark it as deleted. Then, we replace (14 18 6) with (12 15 5). Before finishing Step 2, we clean the subtrees from the marked nodes and reinsert only those nodes that remain un-flagged, i.e., node (11 15 9) in the example. Figures 3.5 (a) and 3.5 (b) show the corresponding Quad-tree2 before and after inserting the vector (12 15 5). □



(a) Before insertion (b) After insertion

Figure 3.5: Inserting the new solution vector (12 15 5) (Example 4)

In Algorithm **Quad-tree3** [SS96b, MTT02, MT04b] the deletion problem is solved differently. The complete algorithm is shown in Algorithm 9 and the included routines are described as follows.

Algorithm 9 : Quad-tree3

Input: $\vec{x}$ to be inserted into a Quad-tree rooted at $\vec{y}$
Output: Updated (domination-free) Quad-tree

1. Start: Let $\vec{y}$ be the root of the tree
2. Successorship: Calculate k such that $\vec{x}$ is the k-Successor of $\vec{y}$
if $k = (11\ldots1)$ or $x_i = y_i, \forall i \in S_0(k)$ **then**
STOP $\{\vec{x}$ is dominated by $\vec{y}\}$
end if
if $k = (00\ldots0)$ $\{\vec{y}$ is dominated by $\vec{x}\}$ **then**
REPLACE($\vec{y}, \vec{x}$)
STOP
end if
3. Dominance test 1: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $l < k$ and $S_0(k) \subset S_0(l)$, Execute TEST1($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ is dominated by $\vec{z}$ or a son of $\vec{z}$}
4. Dominance test 2: for all $\vec{z}$ such that $\vec{z}$ is a l-Son of $\vec{y}$, $k < l$ and $S_1(k) \subset S_1(l)$, Execute TEST4($\vec{x}$,$\vec{z}$) {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
5. Insertion:
if a k-Son of $\vec{y}$ already exists **then**
replace $\vec{y}$ with the k-Son and goto Step 2
else
$\vec{x}$ is the k-Son of $\vec{y}$ and STOP
end if

Algorithm 10 : TEST4($\vec{x}$,$\vec{z}$)

TEST4($\vec{x}$,$\vec{z}$): {Check if $\vec{x}$ dominates $\vec{z}$ or a son of $\vec{z}$}
Calculate k such that $\vec{x}$ is the k-Successor of $\vec{z}$
if $k = (00\ldots0)$ **then**
DELETE($\vec{z}$)
end if
for all $\vec{v}$, such that $\vec{v}$ is a l-Son of $\vec{z}$ and $S_1(k) \subset S_1(l)$, Execute TEST4($\vec{x}$,$\vec{v}$)

In the following routines, the successorships should be changed from binary strings to scalar values (refer to Equation (3.7)).

DELETE($\vec{z}$): This routine deletes the node $\vec{z}$. If $\vec{z}$ has at least one son, then the lowest numbered son of $\vec{z}$ becomes the new root of the subtree. Therefore, only the nodes in the subtrees of the other sons must be considered for reinsertion:

1. Let $k = 1$. Detach the subtree rooted at $\vec{z}$.
2. If the k-Son of $\vec{z}$ exists, denote it by $\vec{t}$, move $\vec{t}$ to the position of $\vec{z}$ in the Quad-tree and goto 4.
3. Let $k = k + 1$. If $k > 2^m - 2$, RETURN, otherwise goto 2.
4. For each k such that $k + 1 \leq k \leq 2^m - 2$, if the k-Son of $\vec{z}$ exists, denote it by $\vec{s}$ and execute REINSERT($\vec{t}$, $\vec{s}$).

REPLACE($\vec{c}, \vec{s}$): In this routine, $\vec{c}$ is replaced by $\vec{s}$, because $\vec{c}$ is dominated by $\vec{s}$. Therefore, the successorships in the subtrees of $\vec{c}$ are no longer valid and corresponding nodes must be reconsidered:

1. Detach the subtree rooted at $\vec{c}$. Replace $\vec{c}$ by $\vec{s}$.
2. For each k such that $1 \leq k \leq 2^m - 2$, if the k-Son of $\vec{c}$ exists in the detached subtree, denote it by $\vec{t}$ and execute RECONSIDER($\vec{s}$, $\vec{t}$).
3. Discard $\vec{c}$.

REINSERT($\vec{c}, \vec{s}$): This routine finds the right position in the subtree rooted at $\vec{c}$ at which to insert $\vec{s}$ and its successors. This routine is called only when there is no need for domination test. This case happens for example in the routine DELETE, where the lowest successor of the deleted root is selected as the new root. Since all other vectors in the detached subtree are non-dominated with respect to the new root, these vectors need only be reinserted into the subtree, without any dominance tests:

1. For each k such that $1 \leq k \leq 2^m - 2$, if the k-Son of $\vec{s}$ exists, denote it by $\vec{t}$ and execute REINSERT($\vec{c}$, $\vec{t}$);
2. Determine l such that $\vec{s}$ is a l-Successor of $\vec{c}$. If the l-Son of $\vec{c}$ exists, denote it by $\vec{t}$ and execute REINSERT($\vec{t}$, $\vec{s}$). If the l-Son of $\vec{c}$ does not exist, move $\vec{s}$ to the position of the l-Son of $\vec{c}$.

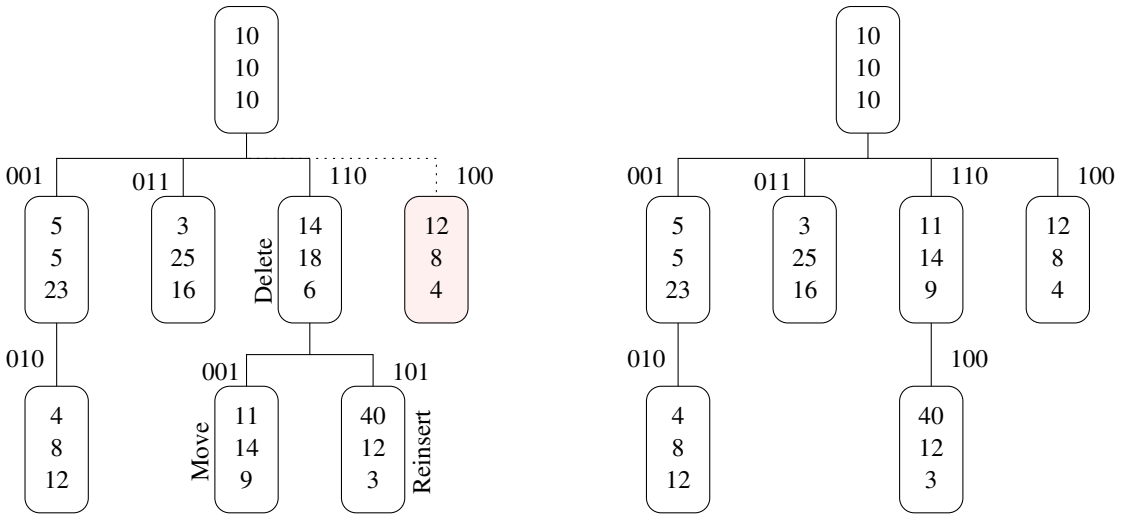
RECONSIDER($\vec{c}, \vec{s}$): This routine is called in the REPLACE routine, when a deleted root is replaced directly by a new vector. In this case, the successors $\vec{s}$ of the deleted root may also be dominated by the new vector $\vec{c}$. This means that each successor $\vec{s}$ of the deleted root must be tested for dominance before being inserted into the new subtree rooted at $\vec{c}$:

1. For each k such that $1 \leq k \leq 2^m - 2$, if the k-Son of $\vec{s}$ exists, denote it by $\vec{t}$ and execute RECONSIDER($\vec{c}$, $\vec{t}$);
2. Determine l such that $\vec{s}$ is a l-Successor of $\vec{c}$
3. If $l = 2^m$, discard $\vec{s}$, RETURN.
4. If the l-Son of $\vec{c}$ exists, denote it by $\vec{t}$ and execute REINSERT($\vec{t}$, $\vec{s}$), else move $\vec{s}$ to the position of l-Son of $\vec{c}$ in the Quad-tree.

Example 5:

This example is dedicated to illustrating the routine DELETE of the Algorithm Quad-tree3. Consider the Quad-tree in Figure 3.6 (a) and let us insert the node (12 8 4). As explained throughout previous examples and in the corresponding algorithms, it is obvious that node (14 18 6) is dominated by (12 8 4) and must be deleted. In the routine DELETE of Quad-tree3, node (14 18 6) is deleted and node (11 14 9), since it is the (001)-Son of it, is moved to its place and the node (40 12 3) is reinserted again from (11 14 9). The result of this insertion is shown in Figure 3.6 (b).

□



(a) Before insertion

(b) After insertion

Figure 3.6: Insertion in a Quad-tree (Example 5)

Discussion

The computational complexities of the Quad-tree approaches depend highly on the shape of the constructed Quad-tree. Habenicht [Hab83] has indicated that the worst case happens when every node has at most two sons, independently of the dimensionality of the vectors under consideration. This case occurs, if all pairs of vectors are different in all components. However, he has carried out different experiments to get a more realistic idea of the computational complexity of the Quad-tree. These experiments are also followed by Sun and Steuer [SS96b]. Here, these data structures are being considered in MOEA and their performances are examined through different experiments in Section 3.5.

3.4 Data Structures in MOEA

The classical data structure for the archive in a group of elitist MOEAs is a linear list [Zit99, Deb01, CVL02]. In the year 2002, Quad-trees and later dominated trees have been developed as alternative data structures [MTT02, MT04b, FES02]. These data structures are used for archiving the non-dominated solutions. As mentioned in Section 2.2.2, archiving refers to a) finding the non-dominated solution, b) updating the archive, c) clustering in MOEAs or bounding the archive size. Here, clustering is not studied, i.e., the size of the archive is not constant. Quad-trees and dominated trees [MTT02, MT04b, FES02] have some common properties as well as some differences when used in MOEA; summarized as follows.

- The domination check using the dominated tree data structure is accomplished by making two kinds of dominated and non-dominated trees. Indeed, two data structures are required to check the domination criterion, whereas in Quad-tress, the successorships are calculated and compared with corresponding branches. Hence, the domination check in Quad-tree is easier than in the dominated tree approach.
- Replacement is a worthwhile step in Quad-trees, whereas it does not exist in dominated trees.
- The dominated tree must be cleaned after each insertion and deletion.
- Deletion is an expensive part for both of the data structures as compared to linear lists.

3.5 Experiments

In this section, the influence of the archive size, population size, and number of objectives on the computational time of a MOEA are studied. In these experiments, the population size varies from 100 to 10000 and the number of objectives from 2 to 10. The archive size is not restricted (intentionally) and the influences of each of these parameters on the computational time is studied. Here, the linear list and the Quad-tree data structures are integrated into the SPEA method [Zit99, ZT99]. Dominated trees are not studied, because they are recorded to be more efficient than linear lists for very low population sizes (e.g., 20) and for a high number of generations [FES02, Fie03], which is not the case here. The data structures are evaluated using different test functions from Section 2.4.
In each generation, each individual of the actual population is tentatively inserted into the archive. It is obvious that only non-dominated solutions do remain in the Quad-tree, because the Quad-tree is domination-free. In SPEA, the archive, as well as the population, undergoes selection, and tournament selection is applied to individuals of the archive and the actual population.

3.5.1 Influence of Population Size $|P|$ on CPU-time for Test Functions with $m = 2$

The first experiments are performed using the test functions ZDT1, ZDT2,... ,ZDT6 with $m = 2$ dimensions (objective space) and unbounded archive size. Therefore, the free parameter is the size $|P|$ of the population. Figure 3.7 and Table 3.1 present and compare the average CPU-times over different runs when running each algorithm for 400 generations and for different population sizes. The recorded archive size for different test functions range between 110 – 127 for ZDT1, 20 – 30 for ZDT2, 190 – 215 for ZDT3, 25 – 33 for ZDT4, 10 – 15 for ZDT5, and 29 – 38 for ZDT6, depending on the initial seed, and independent of the use of linear lists or Quad-trees. Therefore, for all these experiments the archive size $|A|$ is rather small as compared to the population size $|P|$, i.e., $|A| \ll |P|$. Also, the archive sizes did not grow with increasing population size (between 100 and 5000).

Discussion

- Comparison of Quad-tree1, Quad-tree2, and Quad-tree3: In Figure 3.7, it can be seen that for large population sizes, there is almost no difference in CPU-time between the three different Quad-tree variants. However, for small population sizes, Quad-tree3 is always the faster implementation among the

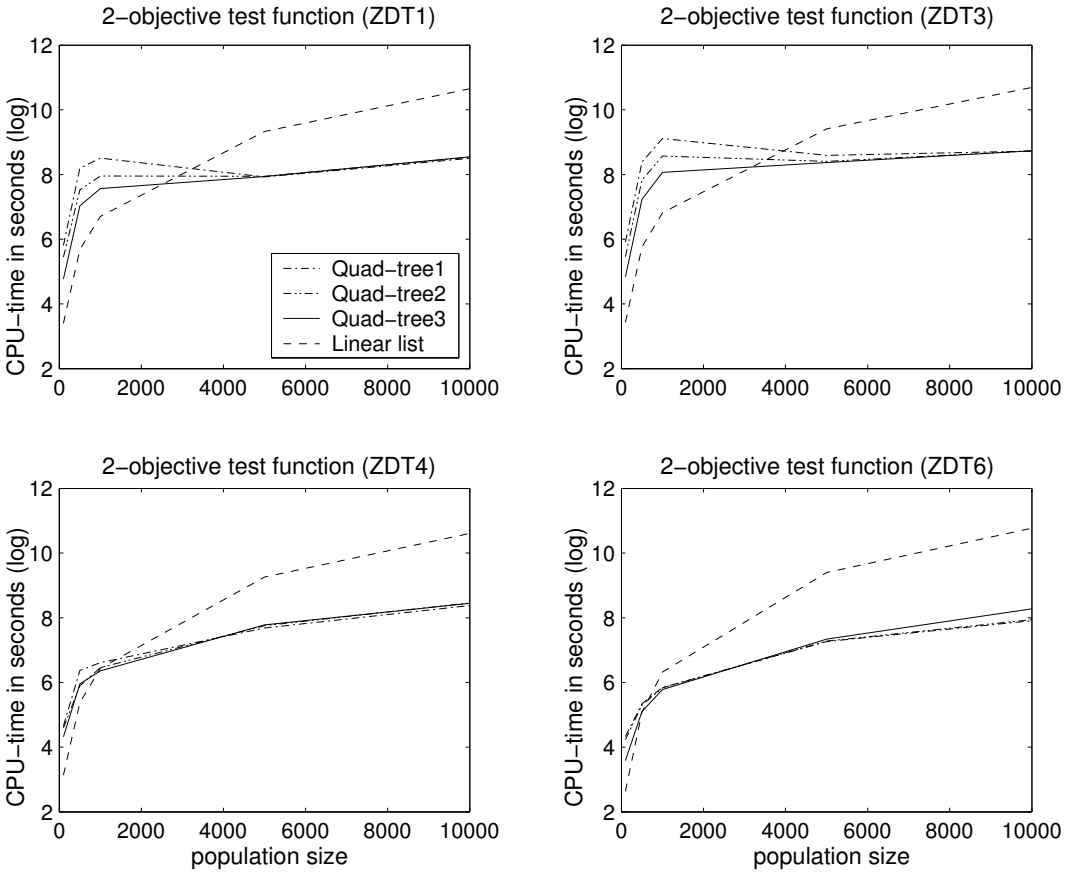


Figure 3.7: Average CPU-time for different 2-objective test functions for different population sizes $|P|$

three.

- Comparison of Quad-tree and linear list archives: In Figure 3.7, one can also see that linear list archives perform better for small population sizes, and depending on the test function, there is a certain population size in the range of 1000 to 5000, which from then on, the Quad-tree implementation becomes faster. For example, in Table 3.1, implementation using the Quad-tree3 is almost 10 times faster than the implementation using the linear list for larger population sizes.

As a summary, it can be concluded that for small dimensions m of the objective space and for problems with $|A| \ll |P|$, Quad-trees are a better data structure for large population sizes.

Table 3.1: Average CPU-times in seconds for different population sizes of the six 2-objective test functions ZDT1 to ZDT6 (from top to bottom) (T_i/L is the ratio of Quad-tree$_i$'s CPU-time to the CPU-time when using linear lists)

Test	$\|P\|$	linear list	Quad-tree1	Quad-tree2	Quad-tree3	T1/L	T2/L	T3/L
ZDT1	100	30.68	332.9	232.83	118.99	10.85	7.59	3.88
	500	294.71	3629.58	1859.02	1133.81	12.32	6.31	3.85
	1000	818.54	4952.93	2840.45	1937.81	6.05	3.47	2.37
	5000	11299.92	2774.41	2826.68	2794.71	0.25	0.25	0.25
	10000	42274.91	4898.97	5100.49	5160.54	0.12	0.12	0.12
ZDT2	100	24.21	120.6	91.05	74.68	4.98	3.76	3.08
	500	186.61	244.68	253.87	234.2	1.31	1.36	1.26
	1000	537.58	418.46	446.45	456.18	0.78	0.83	0.85
	5000	10328.27	2066.73	2218.92	2326.68	0.2	0.21	0.23
	10000	40091.71	4142.21	4385.24	4517.08	0.1	0.11	0.11
ZDT3	100	31.82	370.47	234.36	126.56	11.64	7.37	3.98
	500	318.48	4389.25	2515.66	1387.84	13.78	7.9	4.36
	1000	916.46	9097.29	5283.9	3178.6	9.93	5.77	3.47
	5000	12234.25	5383.5	4484.65	4300	0.44	0.37	0.35
	10000	44055.57	6163.18	6212.26	6195.52	0.14	0.14	0.14
ZDT4	100	23.35	105.75	99.5	75.45	4.53	4.26	3.23
	500	216.72	586.77	363.53	387.83	2.71	1.68	1.79
	1000	626.88	745.08	640.92	580.44	1.19	1.02	0.93
	5000	10524.19	2176.9	2336.12	2393.23	0.21	0.22	0.23
	10000	40317.49	4355.43	4707.76	4655.86	0.11	0.12	0.12
ZDT5	100	20.2	41.66	44.25	45.8	2.06	2.19	2.27
	500	176.19	207.11	216.82	228.72	1.18	1.23	1.3
	1000	546.19	417	437.26	459.84	0.76	0.8	0.84
	5000	10415.29	2104.2	2223.27	2375.99	0.2	0.21	0.23
	10000	40410.43	4244.57	4454.04	4632.82	0.11	0.11	0.11
ZDT6	100	14.5	76.92	69.04	36.82	5.3	4.76	2.54
	500	161.5	209.84	201.86	165.53	1.3	1.25	1.02
	1000	555.29	345.7	342.95	324.13	0.62	0.62	0.58
	5000	12090.32	1430.13	1451.23	1543.43	0.12	0.12	0.13
	10000	47489.3	2728.1	2819.97	3038.54	0.06	0.06	0.06

3.5.2 Influence of Number of Objectives m and Population Size $|P|$ on CPU-time

The second set of experiments involves the test of the influence of the dimensionality m of the objective space and archive size $|A|$ on the CPU-time, again compared to the linear list and the three Quad-tree implementations.
Figure 3.8 shows the measured CPU-times of different test functions for different number of objectives, and again for different population sizes. Since the measured results of Quad-tree3 were also better than the two other Quad-tree implementations (this will be discussed later), we only compare Quad-tree3 with the linear list archive.

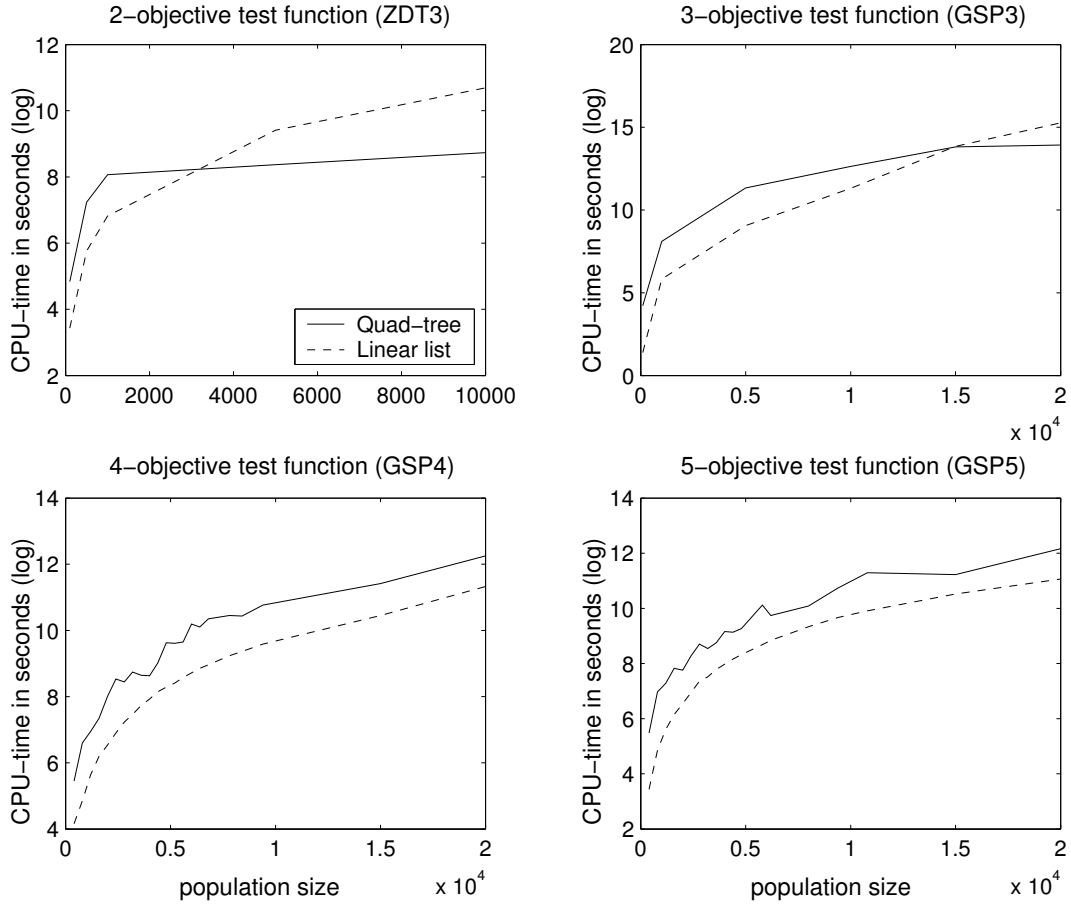


Figure 3.8: Average CPU-time of m-objective test functions for different population sizes

Discussion

- Influence of m on CPU-time: In Figure 3.8, we can see that for the 3-objective test function, the implementation using the linear lists can be up to 10 times faster than the implementation using the Quad-tree archive with the break-even point of equal CPU-time shifted here up to a population size of $|P| = 15000$. The implementation using the Quad-tree is faster here, only for larger population sizes. In the case of the even higher-dimensional test functions (4 and 5), the linear list implementation is faster – for population sizes up to 20000 individuals – than the Quad-tree.

- Influence of archive size $|A|$ on CPU-time: Contrary to the test functions with only 2 objectives, we found out that in all of these higher-dimensional test functions, the archive sizes increased with increasing population size. Figure 3.9 shows the recorded archive sizes for different population sizes for the 4-objective test function GSP4. The archive size of this test function contains more than 60000 vectors, when the population size has been chosen to be 20000. As it seems that it is not the number m of objectives alone but the dimensionality of the approximated Pareto-front, and as a consequence, the archive size $|A|$ that must be taken into account. In order to compare linear lists with Quad-trees, we need to take a deeper look at the number of reinsertions that must be performed as that might become the bottleneck in performance for problems with large archive sizes.

3.5.3 Influence of Archive Size $|A|$ and m on CPU-time for Large Constant Population Size $|P|$

Figure 3.10 shows the CPU-times of the Quad-tree and the linear list implementations in relation to different archive size bounds and a fixed population size of $|P| = 20000$ for the four test functions GSPm with m = 3, 4, 5, 6. It can be seen that the Quad-tree algorithm is faster when the archive size is less than about 2000 for the 3-objective test function, 3000 for the 4-objective test, 6000 for the 5-objective test, and 7000 for the 6-objective test function.
Discussion:

- The above observation confirms that for increasing archive sizes, the Quad-tree implementation becomes less competitive than the linear list implementation. This may be related to the fact that for larger archive sizes, the number of costly reinsertions of already inserted points can destroy the advantage gained

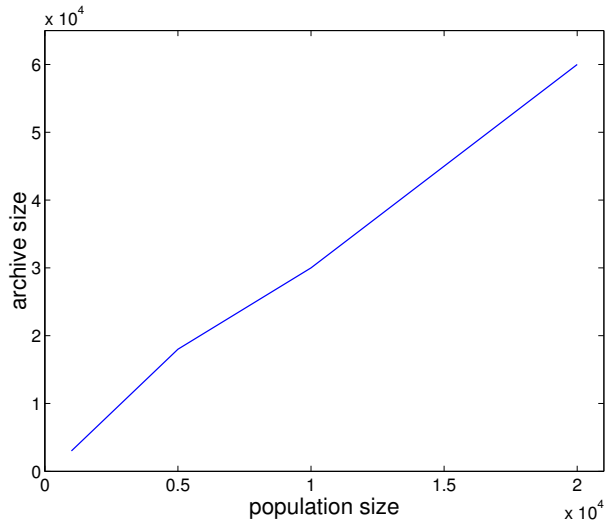


Figure 3.9: Archive sizes of the 4-objective test function according to different population sizes

when not having to traverse the whole list of non-dominated solutions. Reinsertion takes place for all nodes of the subtrees when their root node is dominated and must be deleted. Therefore, we also record the number of deletions and reinsertions relative to the archive size.

Table 3.2 lists the number of recorded deletions and reinsertions in a Quad-tree for different population and archive sizes and test functions with different numbers of objectives.

Discussion:

- We can see that the above test functions and SPEA behave such that when the population size increases, the archive size increases as well.
- The archive size can be much larger than the population size ($|P| \ll |A|$).
- Consider now all rows of equal population size 1000 in Table 3.2. We observe that for a certain archive size, higher values of m reduce the number of reinsertions and deletions. This is because each node in the tree can have at most $2^m - 2$ sons, and for higher values of m, the width of the tree is considerably higher, whereas for smaller values of m, the depth is considerably higher. For example, for $m = 10$, each node can have up to 1022 sons, whereas for $m = 4$, it can have only up to 14 sons. Hence, for a given archive size and large m, the tree will not have a large depth but rather a large width.

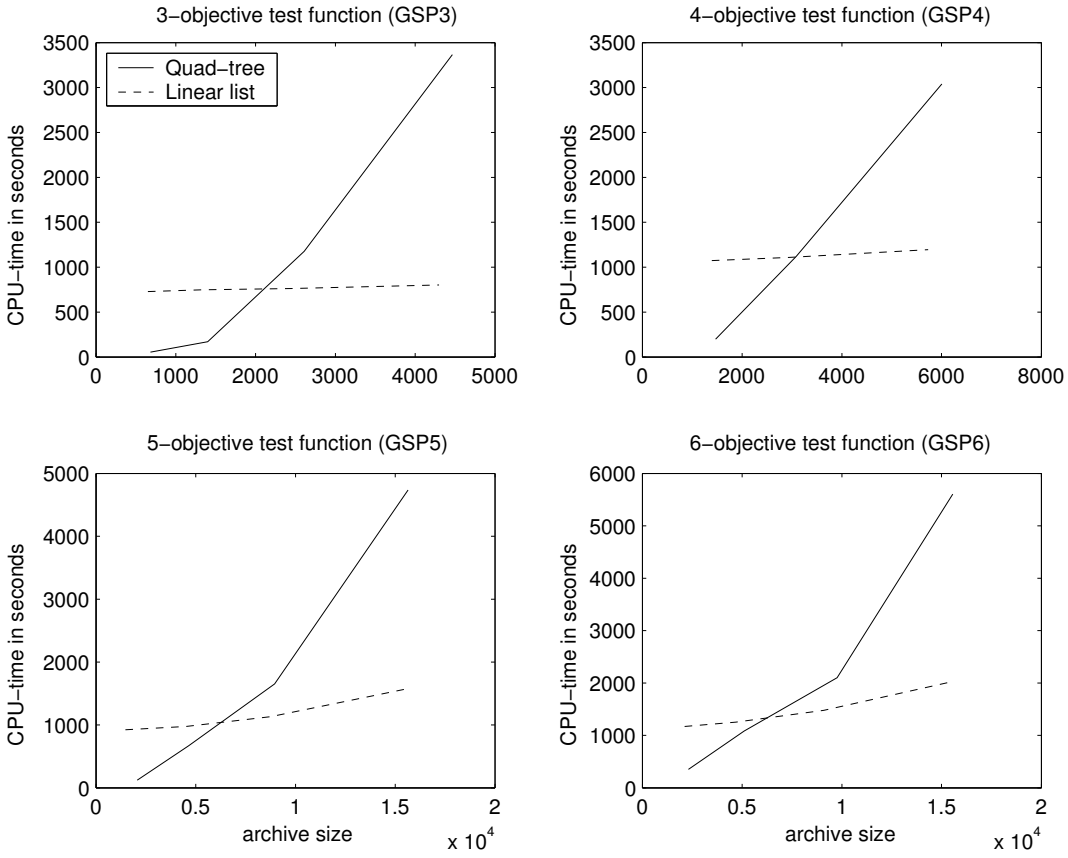


Figure 3.10: Average CPU-time of m-objective test functions GSPm, m = 3, 4, 5, 6 according to different archive size bounds $|A|$ and fixed population size $|P| = 20000$

- For lower values of m and given archive size, the deletion of a dominated node therefore requires a large number, in average, of reinsertions because the subtrees will be longer in average than for trees with higher values of m.
- Finally, the CPU-time increases with increasing m because the number of required comparisons increases. For each insertion, at most $2^{m-1} - 1$ nodes must be checked if they are dominated by the new vector.

All the above experiments present average values obtained from several runs of the MOEA with different seeds and equal number of generations. The values of crossover and mutation probabilities have also been constant.

Table 3.2: Number of deletions and reinsertions in the Quad-tree archive for the m-objective test functions GSPm (T is the CPU-time in seconds).

m	$\|P\|$	$\|A\|$	Deletion	Reinsertion	T
4	100	290	1111	2600	22
	500	1180	6080	15600	432
	1000	**3300**	**12200**	**36150**	**1280**
	2000	7150	22050	49500	3235
	5000	16350	68650	149770	17200
5	**1000**	**4200**	**12500**	**30400**	**1830**
	5000	19950	50071	105600	17850
6	**1000**	**3800**	**12710**	**29000**	**2280**
	5000	18750	57488	127750	30500
7	**1000**	**5200**	**10000**	**23000**	**4240**
	5000	23400	56400	117650	56700
10	**1000**	**5150**	**3170**	**11245**	**17400**
	5000	23542	33216	88074	314131

3.5.4 Comparison

From the results and discussions in the previous section, it can be concluded that the number of comparisons and the computational time depend on three factors:

- number of objectives (m),
- archive size[2]($|A|$),
- population size ($|P|$).

It must be emphasized that the number of generations is also playing an important role in the computational time. Here, it has been kept constant during the comparison of the data structures in MOEA. Also, the convergence and diversity of solutions are not studied here.
It is recorded by Sun and Steuer [SS96b] that Quad-trees always have better computational times than linear lists for storing non-dominated solutions, especially when there is a large number of solutions to insert. They have studied 2- to 8-objective tests with 100 – 10000 randomly generated vectors to be considered for inclusion into an initially empty tree. Hence, they never obtained a tree of a size larger than

[2]Here, only continuous problems have been considered. In reality, there are many discrete problems with small cardinality of the Pareto-optimal solutions.

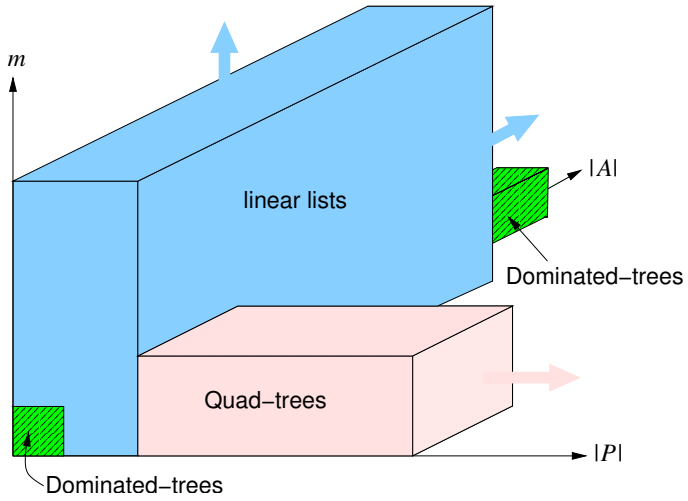


Figure 3.11: Regions of applicability of linear list, Quad-tree, and dominated tree as archive data structures within MOEA relative to archive size $|A|$, population size $|P|$, and number of objectives m

this number of solutions, i.e., 10000. Indeed, with randomly generated sequences, the archive size in their experiments was always much smaller. These conclusions are valid in our context for optimization problems with only one generation.
In MOEA, the archive size can grow much larger than the population size, as the results of implementing the archive data structure into a MOEA, and dynamically updating the archive with elements of the actual population from generation to generation, have been shown. In that case, the Quad-tree archives can become slower than linear list implementations. For example, in experiments done by Sun and Steuer [SS96b], the archive size for an example with 8 objectives and 10000 vectors for inclusion has been about 2400 (when the search space is [0,1]). However, here in the GSPm test functions, the archive size can even become as big as 60000. This is the reason why the Quad-trees can be worse than linear list implementations. Unfortunately, little is known about the average height of a tree e.g., binary trees, when both insertion and deletion are used [CLR90]. Therefore, as a rule of thumb, Quad-trees are more efficient than linear list archives when used in MOEAs when a) the archive sizes are small and b) the population sizes are large. When dealing with smaller population sizes and larger archive sizes, the experiments indicate that linear lists take less computational time than Quad-trees. Figure 3.11 tries to identify regions of better usefulness of both kinds of data structures.
Therefore, Quad-trees must be used if the size of the archive is rather small. Keeping the size of the archive as a fixed value is also a desired property of the MOEA methods, like SPEA, PAES, and others.

Under this restriction, we are able to conclude that Quad-trees take less computational time than linear lists when used inside MOEAs with restricted archive size. Also, Figure 3.11 identifies the region, where dominated trees are more efficient than linear lists in terms of the computational time. Fieldsend et al. [FES02] computed the computational time of an implementation using the dominated trees for several continuous test functions and compared them with linear lists. Their results show that for small population sizes, e.g., $|P| = 20$, and for a large number of generations, e.g., 5000, the implementation using the dominated trees require less computational time than the implementation using linear lists. This approach is applied to large unrestricted archive sizes, e.g., $|A| = 8000$.

Discussion

As mentioned several times in the thesis, archiving is an important block in MOEAs, which helps to attain better convergence of solutions. However, as studied in this chapter, the size of the archive has a great impact on the computational time. It is one of the reasons to use bounding methods for fixing the cardinality of the archive members to a certain (relatively small) amount. By bounding the archive size, it is possible to use Quad-trees to reduce the computational time. However, there are questions such as: Is there a possibility to deal with large archive sizes? How can we obtain the solutions of a problem with many Pareto-optimal solutions? Indeed, there are different possibilities to deal with such problems, e.g., reducing the computational time by approximating the Pareto-optimal front or by using an approximation model of the problem [ND03, KTW02, ETP03].
Nain and Deb [ND03] propose a concept of combining an Evolutionary Algorithm with an approximate evaluation technique in order to achieve a computationally effective search and optimization procedure by using an approximation model of the problem. The major idea is to apply EAs on computationally expensive problems. This method is tested in conjunction with Artificial Neural Networks (ANNs) as an approximation technique. There, EA is run for a relative small number of generations, then the output is used as the training data of ANN. In the next steps, the ANN is utilized to determine the function values. Indeed, this method is efficient when we solve problems with very expensive function evaluations.
Klamroth et al. [KTW02] propose another method to approximate the Pareto-optimal front linearly. In their approach, the approximation comes in the form of a polyhedral distance measure that is being successively constructed during the execution of the algorithm. The measure is being utilized both to evaluate the quality of the approximation and to generate additional non-dominated solutions. However, the algorithm requires an initial approximation of a (or a set of) non-dominated

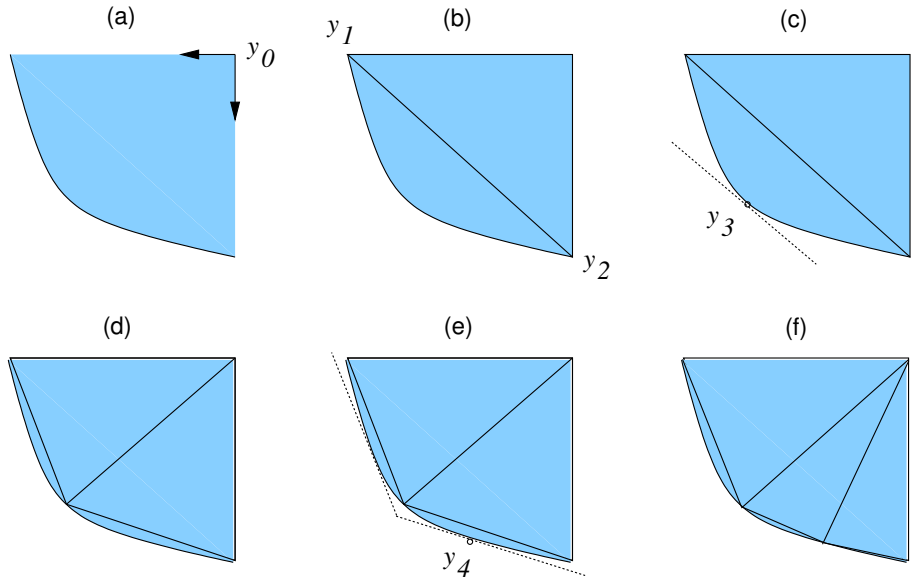


Figure 3.12: An example of inner approximation technique in the objective space [KTW02]. The linear approximation is achieved through several iterations.

solution(s), e.g., Nadir point[3]. In order to deal with convex and non-convex Pareto-optimal fronts, they propose inner and outer approximation techniques. Figure 3.12 shows an example of inner approximation for a 2-objective space. In this figure, starting from (a), the initial approximation is achieved according to (b). This initial approximation is done according to the knowledge of having the Nadir point $\vec{y}_0$ and therefore $\vec{y}_1$ and $\vec{y}_2$. In each iteration a polyhedral distance measure is executed. It is therefore resulted in other non-dominated solutions $\vec{y}_3$ in (c) and $\vec{y}_4$ in (e). The stopping criteria for this algorithm can be a maximum number of iterations (for more details refer to [KTW02, SKW01]). A similar approach is also studied by Ehrgott and Tenfelde-Podehl [ETP03], to find Nadir points and solve discrete problems.

3.6 Conclusion

This chapter is dedicated to archiving and the required computational time for keeping the elite solutions in the archive. The following data structures have been studied:

[3]The Nadir point $\vec{y}^N$ is characterized by the component-wise supremum of all the solutions in the Pareto front P (in the objective space): $y_j^N = \sup_{x \in P} f_j(x); \qquad j = 1, \cdots, m.$

1. Linear list: It is the conventional data structure to store non-dominated solutions of a MOEA in the archive. It is efficient for a low number of objectives, small populations, and large archive sizes.

2. Dominated and non-dominated trees: These data structures can be used to decrease the computational time, when dealing with small population sizes and a high number of generations.

3. Quad-trees: These data structures decrease the computational time, particularly for MOEAs with high population sizes that have a high number of objectives but low archive sizes. This is usually the case in MOEAs when optimizing discrete test functions or continuous test functions with bounded archive sizes.

Here, Quad-trees and linear lists have been implemented as data structures for the archive of an elitist MOEA. It has been shown through different experiments that the Quad-tree data structures take less computational time than linear lists when used as the archive for the MOEAs with large population sizes and small archive sizes. Hence, as a rule of thumb, it is recommended to use the Quad-trees for problems with large populations but small archive sizes. This is the case for problems with large search spaces but where the Pareto-fronts are non-continuous curves or just point sets of small cardinality.

Chapter 4

Hybrid Multi-Objective Evolutionary Algorithm (HMOEA)

In MOEA, the search space is explored by applying recombination operators. However, after terminating the MOEA, we are not sure if we could obtained the Pareto-optimal solutions. One reason is that we have no control on the search process. The search process explores the space by local and global searching (cross-over and mutation operators), but there may be some parts of the space that are not explored, and some parts which are explored several times.

One possible solution for having a controllable exploration is to use *subdivision* techniques [DJ98, DH97]. Indeed, subdivision techniques are based on the division of the search space (parameter space) into subspaces (boxes). Then, the boxes which contain *good* solutions are divided again into boxes. This procedure is repeated several times until an acceptable granularity for the approximated Pareto-optimal front is reached. Subdivision techniques have been used by interpreting the iteration scheme as a dynamical system in [Sch04, SSW02, DSH03].

The goal of this chapter is to introduce a combined global/local search technique that combines subdivision exploration techniques with local search, using MOEA in each box during subdivisions. This hybrid technique is called Hybrid MOEA (HMOEA).

The combination of MOEA methods with other methods is not a new topic. Many researchers have tried to combine MOEAs - because of their properties - with other optimization techniques, e.g., Memetic algorithms [CDG99]. Also, dividing the space into subspaces is not a new idea. In 2003, E. Hughes has used binary space subdivision to approximate the unexplored regions in [Hug03]. There is another algorithm,

which uses interval constraints [BH03] to obtain a controllable search. However, interval constraints are used in the case of uncertainties of objective functions [Tei01]. The main interest here, in using the combination of MOEAs with subdivision technique, is to obtain a) a high convergence and b) high diversity of solutions for problems of a high number of parameters, where obtaining a desirable solution is not possible by using MOEAs or subdivision methods alone.
In the following, subdivision techniques and the HMOEA are explained. Then, the HMOEA is tested on a real world example and used to cover the approximated Pareto-optimal fronts.

4.1 Subdivision Technique

Figure 4.1 shows an example of the subdivision technique [DH97, DJ98, DSH03, SSW02]. This figure shows four steps of the subdivision technique in the parameter space. In the first step, the parameter space is divided into two parts (boxes). Then, a selection process decides to keep either both of the boxes, or one of them. In this figure, both of the boxes are being kept, because both contain at least one *good* solution (this will be explained later). In the next step, each of the boxes is divided into two boxes (2 divisions) and the selection process keeps just three of these boxes for further division. This process of subdivision and selection is iteratively repeated until a desired granularity of solutions is obtained. Algorithm 11 explains the subdivision technique.

Algorithm 11 : Sampling Algorithm

1. Subdivision
Construct from $\mathcal{B}_{k-1}$ a new system $\hat{\mathcal{B}}_k$ of subsets such that
$\bigcup_{B\in\hat{\mathcal{B}}_k} B = \bigcup_{B\in\mathcal{B}_{k-1}} B$
and $\operatorname{diam}(\hat{\mathcal{B}}_k) = \theta_k \operatorname{diam}(\mathcal{B}_{k-1})$, where $0 < \theta_{min} \leq \theta_k \leq \theta_{max} < 1$.
2. Selection
$\forall B \in \hat{\mathcal{B}}_k$: Choose a set of test points P_B
$N :=$ non-dominated points of $\bigcup_{B\in\hat{\mathcal{B}}_k} P_B$
$\mathcal{B}_k := \left\{ B \in \hat{\mathcal{B}}_k : \exists p \in P_B \cap N \right\}$

The algorithm[1] starts with a (large) compact set (box[2]) in parameter space. By a

[1] In this algorithm $\operatorname{diam}(B_{c,r}) = 2\|r\|_2$ and $\operatorname{diam}(\mathcal{B}) = \max_{B\in\mathcal{B}} \operatorname{diam}(B)$.

[2] An n-dimensional box can be represented by a center $c \in \mathbb{R}^n$ and a radius $r \in \mathbb{R}^n$. Thus $B = B_{c,r} = \{x \in \mathbb{R}^n : c_i - r_i \leq x_i \leq c_i + r_i \; \forall i = 1, .., n\}$.

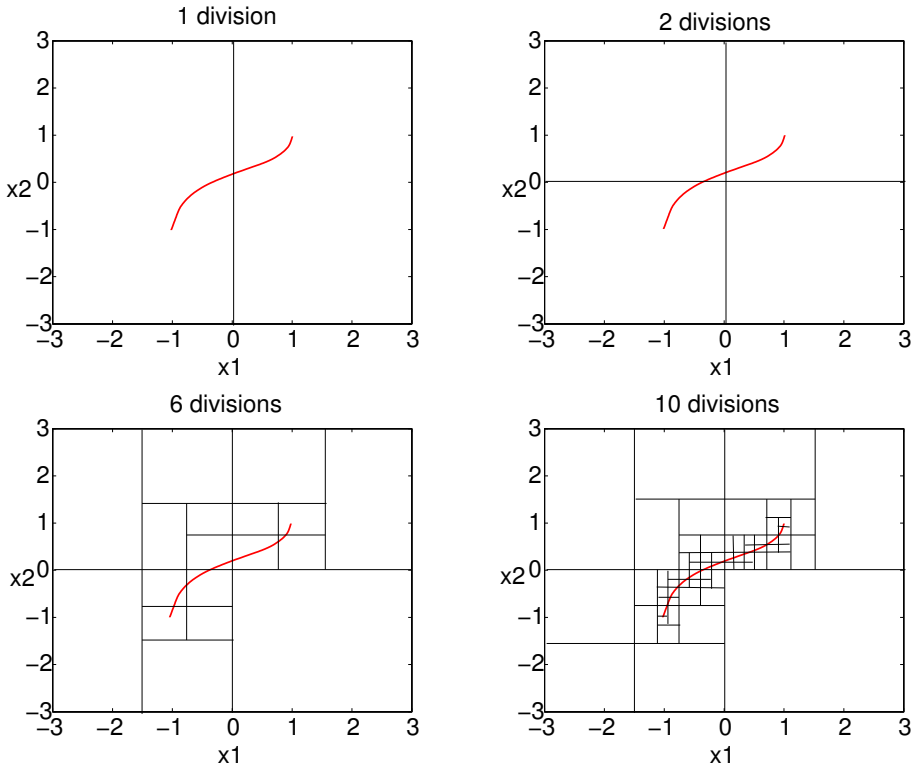


Figure 4.1: An example of the subdivision technique (2-dimensional parameter space)

repeated bisection and selection of boxes, the box coverings $\mathcal{B}_k$ get tighter until the desired granularity of this outer approximation is reached. The selection process in Step 2 of Algorithm 11 is based on a finite set of *test points* within each box. Indeed, in this step several test points are defined in each box. Then, the non-dominated solutions among all the test points in all the boxes are stored in the set N. The boxes which contain at least one of the non-dominated solutions in the set N can survive for further subdivision.

Discussion

If the number of test points in each box is chosen too high, the computational time will be too long. On the other hand, too few test points causes the algorithm to not find a good approximation of the Pareto-optimal front. By using this algorithm it is possible to detect a good approximation of the Pareto-optimal front. This subdivision technique possess a "smoothing" property: A box is kept if it contains *at least* one "good" point.
Subdivision techniques work particularly well for low number of objective functions

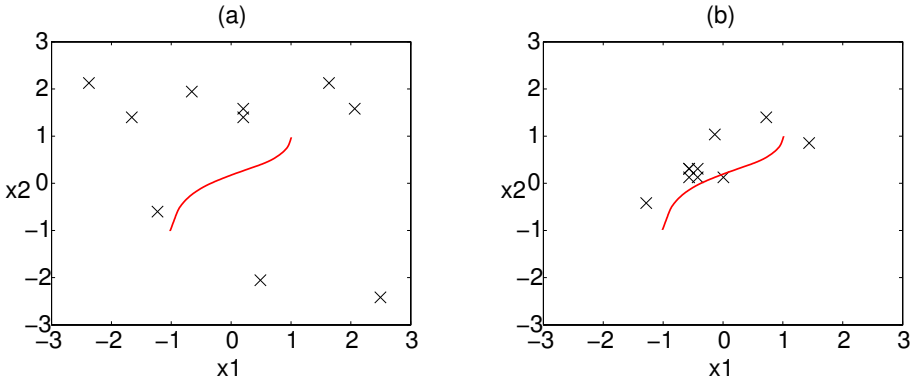


Figure 4.2: Parameter space (a) 10 initial individuals (b) 10 individuals after 10 generations. The Pareto-optimal set is illustrated with a solid line.

and parameters. Otherwise, the number of boxes created in the subdivision procedure is getting too large and indeed, the computational time increases.

4.2 HMOEA

Due to selection, recombination, and elitism, MOEAs can typically generate some solutions very close to the true Pareto-optimal front in low number of generations, in other words very quickly. This is illustrated by the following minimization example. Consider the test function TEST1 in Table 2.1. Figure 4.2 shows the parameter space and the Pareto-optimal front of the test function.
In Figure 4.2 (a), a starting population consisting of 10 randomly chosen individuals in the domain $Q = [-3, 3] \times [-3, 3]$ is shown. Figure 4.2 (b) shows the resulting population after 10 generations using the SPEA2 method (see Section 2.2.1). Here, we observe that after just 10 generations, there are some individuals close to the Pareto-optimal front. This property makes it possible to improve the sampling algorithm described above.
Instead of using many test points to evaluate a (high-dimensional) box, our idea is to take just a few test points as the initial population of a "short" MOEA [3]. The MOEA must be run for a short time in a box B. The box B is kept if it contains at least one solution in N, namely the set of non-dominated solutions of the total set of test points (Step 2, Algorithm 11).
Therefore, the HMOEA is a combination of MOEA and the subdivision technique: The exploration of each box created by subdivision is achieved by running a short MOEA.

[3] A short MOEA is characterized by a short running time; that means small initial population and few generations.

This is shown in Algorithm 12, where $MOEA(B)$ means to run a MOEA in the box B. The output of the MOEA (as it is shown in Algorithm 2) is the set of solutions in the last archive.

Algorithm 12 : Hybrid MOEA Algorithm

1. Subdivision
The same as in Algorithm 11
2. Pre-optimization
$\forall B \in \hat{\mathcal{B}}_k$, $P_B = MOEA(B)$
3. Selection
$N :=$ non-dominated points of $\bigcup_{B \in \hat{\mathcal{B}}_k} P_B$
$\mathcal{B}_k := \left\{ B \in \hat{\mathcal{B}}_k : \exists p \in P_B \cap N \right\}$

Example 1

The HMOEA is tested on a 3-objective test function TEST2 from Table 2.1. This test function has 3 parameters and 3 objectives. The Pareto-optimal front is continuous in both the parameter and objective space. Figure 4.3 shows the result of the HMOEA in the parameter space.

□

As it is shown in Figure 4.3, there are some empty regions in the approximated Pareto-optimal set. Indeed, it may be the case that in the course of the subdivision procedure, boxes get lost although they are part of the Pareto-optimal set.

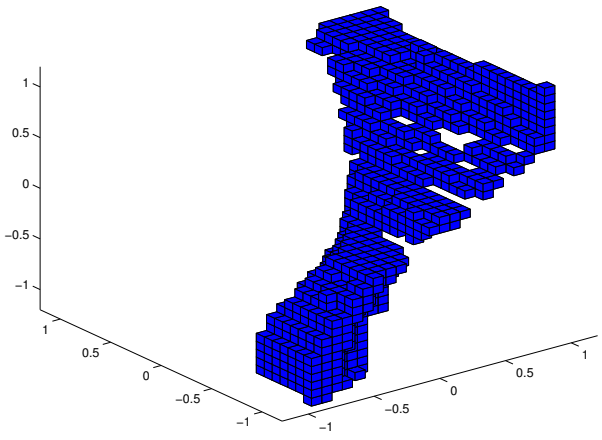
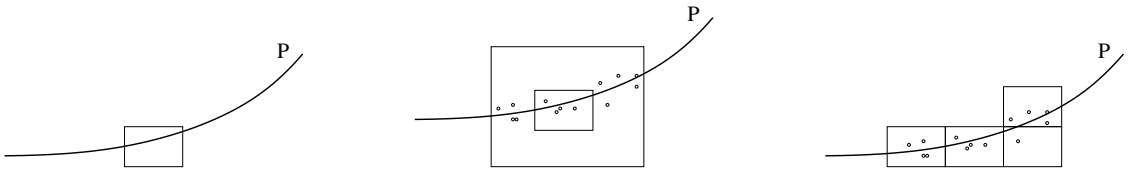


Figure 4.3: Result of applying HMOEA to test function TEST2 (parameter space)



(a) covering (b) local search (c) recovering

Figure 4.4: Working principle of Static Recovering [SMDT03]. The solid line illustrates the Pareto-optimal set P in the Parameter space. (a) Just one box is found. (b) The size of the box is extended to a larger box. In the extended box, HMOEA is run. (c) The result of the HMOEA inside the extended box is a set of four boxes.

To avoid gaps in the Pareto-optimal front, a strategy called *Static Recovering* is proposed [SMDT03]. The idea is to run the HMOEA and obtain possible optimal solutions. Then, the Static Recovering is applied on the solutions to fill the gaps.

4.2.1 Static Recovering

Static Recovering is proposed by Schütze et al. [SMDT03] to fill the gaps made in the Pareto-optimal set in HMOEA (for the case that the covering of the Pareto-optimal set is not complete).
The aim of the algorithm is to extend the given box collection step by step along the covered parts of the Pareto-optimal set until no more boxes are added. In order to find the corresponding neighboring boxes of a given box B with center c and radius r, a MOEA is run in the extended box $\hat{B}$ given by center c and radius $\lambda \cdot r$ with $\lambda > 1$, say $\lambda = 3$. Afterwards, the box collection is extended by the boxes, which contain points from the resulting population. In the beginning of the Static Recovering, this process is completed for all boxes from the box collection. Otherwise, this local search has to be done only in the neighborhood of the boxes that were added in the preceding step.

Example 2

The Static Recovering algorithm is applied to the result obtained from the Example 1. Figure 4.5 shows the obtained result. Here, we observe that there are no gaps in the approximated Pareto-optimal set.

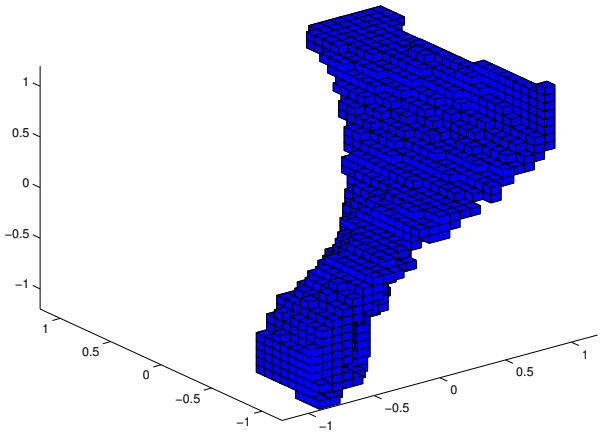


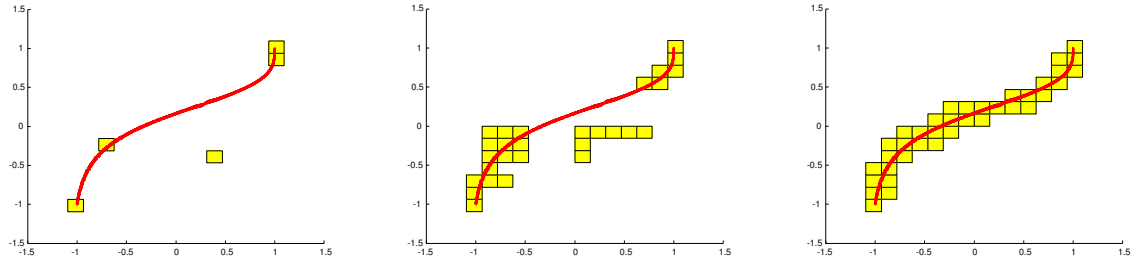
Figure 4.5: Result of applying Static Recovering on the result shown in Figure 4.3

Discussion

Static Recovering allows the addition of boxes into the given collection. The desired coverage of the set of Pareto-optimal solutions cannot get worse, but will improve if the parameters of the algorithm are adjusted properly. On the other hand, Static Recovering does not work adequately in the case where a box does not contain some parts of the Pareto-set but is possibly far away. In this case, the algorithm would extend the box covering by many undesired regions on their way towards the Pareto-optimal set (in particular in higher dimensions). Thus, when there are some boxes far from the Pareto-optimal set, a dynamic recovering strategy has been proposed [SMDT03].

4.2.2 Dynamic Recovering

The principle of the Dynamic Recovering algorithm is the same as Static Recovering. The sizes of existing boxes are extended by a factor λ. Then, a MOEA is run in each extended box. In Dynamic Recovering, only those boxes are kept which contain at least one non-dominated solution (Static Recovering keeps boxes containing the final population).
This recovering algorithm has the disadvantage that some boxes close to the Pareto-optimal set can be deleted while they have been computed once. The speed of the algorithm depends – besides the MOEA – on the choice of the extension factor λ. A larger value of λ yields faster convergence. In general, the number of generations and the size of the population should increase with λ. Dynamic Recovering is indeed a local recovering technique and can be used to improve the result of a MOEA. In



(a) (b) (c)

Figure 4.6: Application of Dynamic Recovering in a simple example. The solid line illustrates the Pareto-optimal set in parameter space. (a) initial box collection; there is one box far from the Pareto-optimal set, (b) one step after Dynamic Recovering, (c) last step in recovering. The box collection covers the Pareto-optimal set

this case, MOEA should deliver solutions with a good diversity and convergence to the Pareto-optimal set. The convergence of the MOEA should be good enough in order not to insert too many superfluous boxes.

Example 3

An example of Dynamic Recovering technique is shown in Figure 4.6. In this figure, Dynamic Recovering is applied to a chosen initial box collection. The algorithm stops after 2 iterations with a total covering of the Pareto-optimal set.

□

4.3 Experiments

The HMOEA together with the Static Recovering technique are tested on a real-world MOP in antenna design ([JJK97]). The MOP is a two-objective problem with 12 parameters. The objective functions are as follows:

$$f_1(x_\nu, y_\nu) = -4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(\ell)(x_\nu + iy_\nu) \right|^2 ,$$

$$f_2(x_\nu, y_\nu) = \max_{\eta=0,..,5} \left(4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(\ell)(x_\nu + iy_\nu) e^{i\nu s_\eta} \right|^2 \right)$$

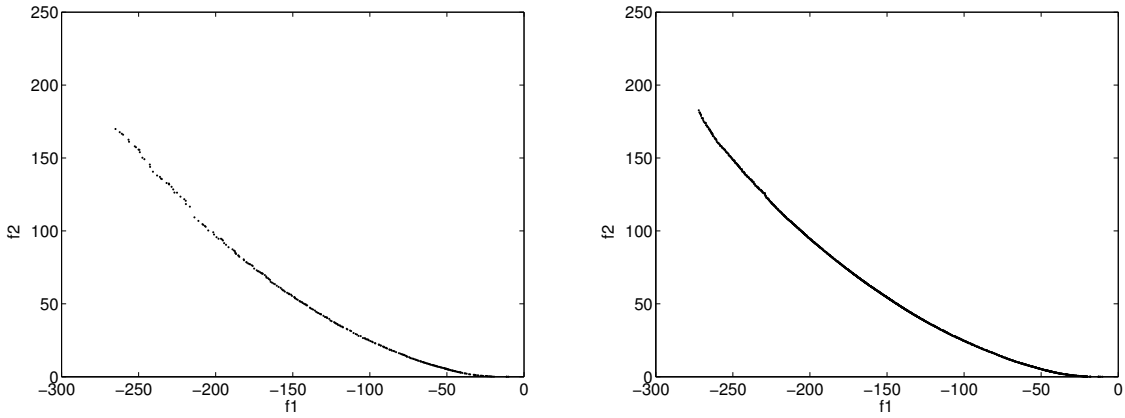


Figure 4.7: Results of (left) MOEA method and (right) HMOEA using Static Recovering

subject to the constraints

$$x_\nu, y_\nu \in \mathbb{R} \quad (\nu \in \mathbb{Z},\ |\nu| \leq n),$$
$$2\pi \sum_{\nu=-n}^{n} (x_\nu^2 + y_\nu^2) \leq 1$$

with the specific discretization points $s_\eta = \frac{3}{4}\pi + \eta\frac{\pi}{10}$. Here, $\mathcal{J}_\nu$ denotes the Bessel function of ν-th order. The algorithms are tested for $n = 5$ and $\ell = 10$. Since $\mathcal{J}_\nu(x) = (-1)^\nu \mathcal{J}_{-\nu}(x)$ and $\mathbb{C} \cong \mathbb{R}^2$, this leads to a model with 12 free parameters. Figure 4.7 shows the results of running a MOEA and the HMOEA using the Static Recovering technique. The MOEA method tested here is the SPEA [Zit99] algorithm with the following parameter settings:

- population size: 300,
- length of individual bitstrings: 7,
- number of generations: 300,
- archive size : 300.

The total running time is 20 minutes for the MOEA and 15 minutes for the HMOEA. Indeed, the Static Recovering technique is used to speed up the running time while the total performance of HMOEA is maintained.
Furthermore, another experiment using the MOEA with the following parameters has been performed:

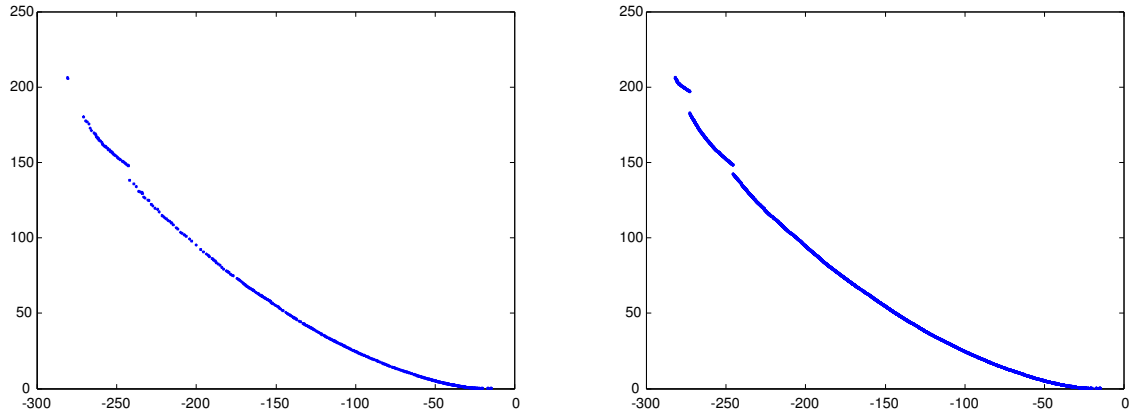


Figure 4.8: Results of (left) MOEA method and (right) HMOEA using Static Recovering technique

- population size: 1000,
- length of individual bitstrings: 13,
- number of generations: 500,
- archive size: 1000.

Figure 4.8 shows the results of the MOEA and the HMOEA using the Static Recovering technique. Here, the selected population size is very high (1000), since a high convergence and diversity of solutions is needed. Also, the archive size is selected very high (1000). The existence of the restricted archive is necessary to obtain good diversity of solutions. On the other hand, the high value of the archive size helps the recovering technique to operate fast enough to cover the front. As observed in Figure 4.8, there are local improvements. The total running time is 7.5 hours for the MOEA method, and 20 minutes for the application of the HMOEA using Static Recovering on it. Figure 4.9 shows the comparisons of selected areas from Figures 4.7 and 4.8. Indeed, the HMOEA together with the recovering technique provides a better convergence within comparable computational time.

4.3.1 Discussion

The above experiments illustrate a possible covering of the approximated Pareto-optimal front by the HMOEA. Here, covering means to find a finite number of solutions, which are very close to each other. Indeed, it is impossible to find all the Pareto-optimal solutions of a continuous front. Also, the Static Recovering

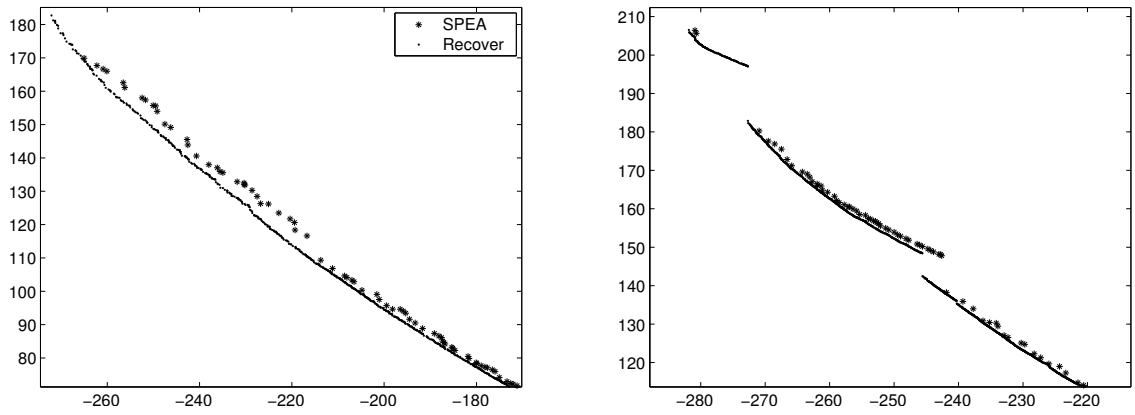


Figure 4.9: Comparison of selected areas of Pareto-fronts from (left) Figures 4.7 and (right) 4.8

method improves the results of the MOEA. The question is, can the MOEA cover the approximated Pareto-optimal front itself? Indeed, it should be possible by running the MOEA for a large number of generations. It must be emphasized that the class of MOEA methods, which use the domination criteria are able to approximate some solutions, but fail to find the (exact) Pareto-optimal front. Figure 4.10 shows the above argument for a very simple example. In this figure, some solutions are not on the Pareto-optimal front, although all the solutions are indifferent to each other. This is because of the domination criteria, and it occurs when a finite number of solutions are found by the MO method. This is also argued by D. Büche et al. [BMK03]. In this case, the recovering process may facilitate a way to overcome this problem in some cases. This can be observed in Figure 4.9.
The recovering techniques reduce the computational time for obtaining a good approximation of a Pareto-optimal front. This has also been studied in the experiments. The MOEA requires a high computational time to find a large number of solutions. This is due to the size of the archive. As discussed in the last chapter, large archive sizes increase the computational time of the MOEA. Therefore, the idea of running a MOEA with a small fixed size archive, and then applying a recovering technique, may decrease the computational time. However, the recovering techniques improve the solutions of the MOEA locally.

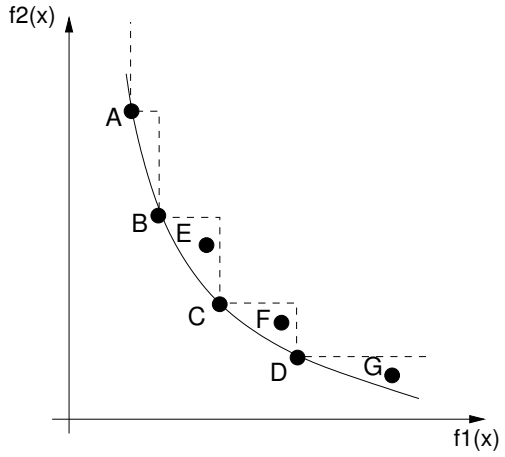


Figure 4.10: Each of the solutions E, F, and G are indifferent to each of the Pareto-optimal solutions (A, B, C and D), but do not lie on the Pareto-optimal front.

4.4 Conclusion

This chapter is dedicated to studying the controllable exploration of the search space by MOEAs. MOEAs find good convergence and diversity of solutions, but covering the Pareto-optimal front in low computational time is not often possible with MOEAs. Therefore, a combination of MOEA and subdivision techniques is investigated here to obtain high convergence of solutions and to cover the Pareto-optimal front. The subdivision methods cannot solve discrete problems and need high computational time when optimizing problems with a high number of parameters. Therefore, the Hybrid MOEA (HMOEA) is studied to omit the problems of both of the methods. It is also tested on a real-world example. However, it is also shown in an example that the subdivision method may in some cases fail to find an exact approximation of the Pareto-optimal front. This is then solved by proposing a Static Recovering technique to cover the Pareto-optimal front. Static Recovering is tested on an antenna design problem. It must be mentioned that the HMOEA depends on the result obtained by the MOEA, indeed the recovering process is just a local improvement of the obtained Pareto-optimal front. The results of this chapter are further analyzed in Chapter 7.

Chapter 5

Multi-Objective Particle Swarm Optimization (MOPSO)

Different from evolutionary computation techniques, Particle Swarm Optimization (PSO) [KE95, SE98] is motivated from the simulation of the social behavior of bird flocking and fish schooling. PSO was originally designed and developed by Eberhart and Kennedy [KE01]. However, it shares many similarities with evolutionary computation techniques. The system is initialized with a population of random solutions and searches for optima by updating generations. Unlike EA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions fly through the problem space by following the current optimum particles.

PSO simulates the behaviors of bird flocking. Suppose the following scenario: A group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what is the best strategy to find the food? The effective one is to follow the bird nearest to the food. PSO learned from this scenario and uses it to solve optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of the particles have fitness values, which are evaluated by a fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particle called guide.

Recently, researchers are paying more and more attention to PSO, for solving multi-objective problems [CL02, FS02, HE02, PV02a, Fie03, MT03b]. Changing a PSO to optimize a multi-objective problem requires a redefinition of what a guide is in order to obtain a front of optimal solutions. In Multi-Objective Particle Swarm Optimization (MOPSO), non-dominated solutions must be used to determine the guide for each particle. Selecting the guide (the best local guide) from the set of

non-dominated solutions for each particle of the population [FS02] is very difficult, yet an important problem for attaining convergence and diversity of solutions.
In this chapter, the basic concepts of PSO and MOPSO are briefly reviewed. Then, a new method called *Sigma method* is introduced. Sigma method is actually a technique for selecting the best local guides for particles in the population. This method is tested on different test functions and compared with other methods. The implementation results show that by using the Sigma method in a MOPSO, we can achieve a very good convergence and diversity of solutions. Following the Sigma method, a new metric for measuring the diversity of a non-dominated set is also introduced in this chapter.
Covering the Pareto-optimal front is another task that is easily solvable by MOPSO. This is also investigated at the end of this chapter.

5.1 Particle Swarm Optimization (PSO)

PSO consists of a population of particles, which contrary to EA, survive up to the last generation. The particles search the variable space by moving with a special speed towards the best global particle (guide) by using their experience from the past generations. A PSO method can be formulated as follows. A set of N particles are considered as a population P_t in the generation t. Each particle i has a position defined by $\vec{x}^i = (x_1^i, x_2^i, \cdots, x_n^i)$ and a velocity defined by $\vec{v}^i = (v_1^i, v_2^i, \cdots, v_n^i)$ in the variable space S. In generation $t+1$, the velocity and position of each particle i is updated as below:

$$\begin{aligned} v_{j,t+1}^i &= wv_{j,t}^i + c_1 R_1 (p_{j,t}^i - x_{j,t}^i) + c_2 R_2 (p_{j,t}^{i,g} - x_{j,t}^i) \\ x_{j,t+1}^i &= x_{j,t}^i + v_{j,t+1}^i \end{aligned} \tag{5.1}$$

where $j = 1, \ldots, n$, $i = 1, \ldots, N$, and

- w is the so called *inertia weight* of the particle,
- c_1 and c_2 are two positive constants,
- R_1 and R_2 are random values in the range $[0, 1]$,
- $\vec{p}_t^{\,i,g}$ is the position of the global best particle in the population, which guides the particles to move towards the optimum,
- $\vec{p}_t^{\,i}$ is the best position that particle i could find so far. Indeed, it is like a memory for the particle i and is updated in each generation.

In PSO, the performance of each particle is measured according to a pre-defined fitness function, which is related to the problem to be solved.

Inertia Weight

The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thus to influence the trade-off between global and local exploration abilities of the particles [SE98, KE01]. A larger inertia weight w facilitates global exploration while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight w can provide a balance between global and local exploration abilities and thus requires less iterations on average to find the optimum [SE98, KE01]. A nonzero inertia weight introduces the preference for the particle to continue moving in the same direction it was going on the previous iteration.

Control Parameter

In Equation (5.1), c_1R_1 and c_2R_2 are called control parameters or *acceleration constants* [KE01]. These two control parameters determine the type of trajectory the particle travels. If R_1 and R_2 are 0.0, it is obvious that $v = v + 0$ and $x = x + v$ (for $w = 1$). It means the particles move linearly. If they are set to very small values, the trajectory of x rises and falls slowly over time.

Particle Interaction

The effectiveness of the particle swarm algorithm comes from interactions of particles with their neighbors. As one particle discovers an optimum, it becomes the best in the neighborhood and attracts (guides) the other particles to itself. Indeed, there is no selection process in a PSO method, unlike EAs.

5.2 MOPSO

The important part in multi-objective particle swarm optimization (MOPSO) is to determine the best global particle $\vec{p}_t^{\,i,g}$ for each particle i of the population. In single-objective PSO, the global best particle is determined easily by selecting the particle that has the best position. Since the goal in multi-objective optimization is to cover the Pareto-optimal front, each particle of the population must select one of the Pareto-optimals as its global best particle, which we call the *best local guide*. Algorithm 13 shows a possible structure of a MOPSO with elitism, where t denotes

the generation index, P_t the population, and A_t the archive at generation t. The algorithm begins with N particles in the population. Here, elitism is considered by storing the non-dominated solutions in the archive A_t. In Algorithm 13, the function $Evaluate$ evaluates the particles in the population P_t, the function $Update(P_t, A_t)$ compares whether members of the current population P_t are non-dominated with respect to the members of the actual archive A_t, how and which of such candidates should be considered for insertion into the archive, and which should be removed. Thereby, the archive is kept domination-free. Obviously, during execution of the function $Update$, dominated solutions must be removed.

Algorithm 13 : MOPSO Algorithm

Input: N
Output: A
1. Initialization: Initialize population P_t, $t = 0$:
for $i = 1$ to N **do**
Initialize $\vec{x}_t{}^i$, $\vec{v}_t{}^i = \vec{0}$ and $\vec{p}_t{}^i = \vec{x}_t{}^i$
end for
Initialize the archive $A_t := \{\}$
2. Evaluate: $Evaluate(P_t)$
3. Update: $A_{t+1} := Update(P_t, A_t)$
4. Move: $P_{t+1} := Move(P_t, A_t)$
for $i = 1$ to N **do**
$\vec{p}_t^{\,i,g} := FindBestLocal(A_{t+1}, \vec{x}_t{}^i)$
for $j = 1$ to n **do**
$v^i_{j,t+1} = wv^i_{j,t} + R_1(p^i_{j,t} - x^i_{j,t}) + R_2(p^{i,g}_{j,t} - x^i_{j,t})$
$x^i_{j,t+1} = x^i_{j,t} + v^i_{j,t+1}$
end for
if $\vec{x}^{\,i}_{t+1} \prec \vec{p}_t{}^i$ **then**
$\vec{p}^{\,i}_{t+1} = \vec{x}^{\,i}_{t+1}$
else
$\vec{p}^{\,i}_{t+1} = \vec{p}_t{}^i$
end if
end for
5. Termination: Unless a *termination criterion* is met $t = t + 1$ and *goto* Step 2

Selecting the best local guide is achieved in the function $FindBestLocal(A_{t+1}, \vec{x}_t{}^i)$ for each particle i. In this function, each particle has to change its position $\vec{x}_t{}^i$ towards the position of a local guide, which must be selected from the updated set of non-dominated solutions stored in the archive A_{t+1}. This function will be discussed in the next section. In Step 4, the position $\vec{p}^{\,i}$ of the particle i is updated.

$\vec{p}^{\,i}$ is like a memory for the particle i and keeps the non-dominated (best) position of the particle by comparing the new position $\vec{x}_{t+1}^{\,i}$ in the objective space with $\vec{p}_t^{\,i}$ ($\vec{p}_t^{\,i}$ is the last non-dominated (best) position of the particle i).
The steps of an elitist MOPSO are iteratively repeated until a termination criterion is met, such as a maximum number of generations (T), or when there has been no change in the set of non-dominated solutions found for a given number of generations. The output of an elitist MOPSO method is the set of non-dominated solutions stored in the final archive.

5.3 Finding Best Local Guides

As mentioned before, several MOPSO methods [HE02, HES03, CL02, FS02, PV02a, Fie03, MT03b, Li03] are already available. In each of these methods (apart from the differences in the main algorithm), there is also a suggestion for finding the best local guides. In most of these methods, there is an inspiration from Multi-Objective Evolutionary Algorithms. In this section, some of the MOPSO methods are briefly reviewed and then a new method for finding best local guides is introduced.

Hu and Eberhart's MOPSO

Hu and Eberhart [HE02] present a MOPSO that uses a dynamic neighborhood strategy. In their method explained for two-objective optimization, the best local guide $\vec{p}^{\,i,g}$ for the particle i is found in the objective space as follows.

- The distance of the particle i to other particles is calculated in terms of the first objective value, which is called *fixed objective*.
- k local neighbors based on the calculated distances are found.
- The local optimum among the neighbors in terms of the second objective value is selected as the best local guide $\vec{p}^{\,i,g}$ for the particle i.

In this method, the selection of the *fixed objective* requires a priori knowledge of the objective functions and one-dimensional optimization is used to deal with multiple objectives. Therefore, selecting the best local guides depends on just one of the objectives. It also depends on the value of k.

Coello Coello and Lechuga's MOPSO

Coello Coello and Lechuga [CL02] propose a MOPSO which has the same structure as in Algorithm 13. In this method, the following steps are carried out, before

selecting the best local guide $\vec{p}^{\,i,g}$ for each particle i:

- The objective space is divided into hypercubes, as if putting a grid on the objective space.
- A fitness value is assigned to each hypercube depending on the number of elite particles that lie in it. The more elite particles in a hypercube, the less its fitness value.
- Then, roulette-wheel selection is applied on the hypercubes and one of them is selected.
- $\vec{p}^{\,i,g}$ is a random particle selected from the selected hypercube.

Therefore, the best local guide is selected by using the roulette-wheel selection method, which is a random selection. Indeed, it is possible that a particle does not select a suitable guide as its local guide.

Fieldsend and Singh's MOPSO

Fieldsend and Singh [FS02] present a MOPSO which uses an unconstrained archive. In their method, a dominated tree data structure (see also Section 3.2) is used for storing the elite particles, which facilitates the choice of a best local guide for each particle of the population. By using this special archive, they address the issue of finding the best local guide and thereby consider all the objectives. In their method, they store the non-dominated solutions in the archive called dominated tree. The dominated tree is explained in Section 3.2.
Figure 5.1 shows this method for a two-objective example, where $\vec{c}_1, \ldots, \vec{c}_4$ are composite points. The selection of the best local guide for a particle in the population is based on its closeness (in objective space) to a particle in the archive. The best local guide for a particle i is that archive member of the composite point $\vec{c}_j$ contributing to the vertex, which is less than or equal to the corresponding objective in i. This is also shown in Figure 5.1. In the case that a composite point has more than one vertex less or equal to particle i, one of the vertices that meets the condition is selected at random (for more details see [FS02]).

Discussion

The way in which the local guides are selected in Fieldsend and Singh's MOPSO is considered better than the method proposed by Coello Coello and Lechuga [FS02]. However, they have both tested their algorithms for two-objective test functions.

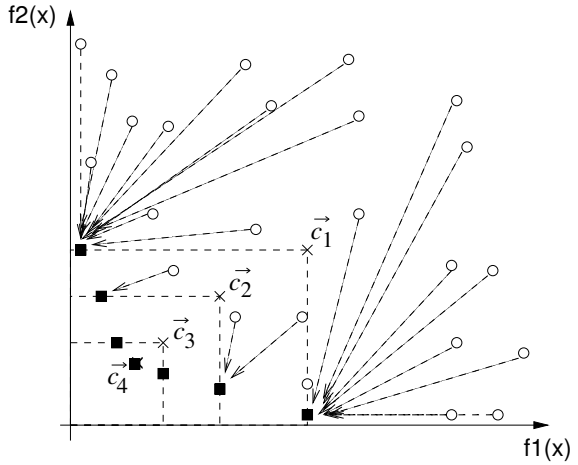


Figure 5.1: Choosing the best local guide among the archive members for each particle in the population by Fieldsend and Singh's method [FS02]. (■: archive member, ○: particle of the current population and ×: composite points)

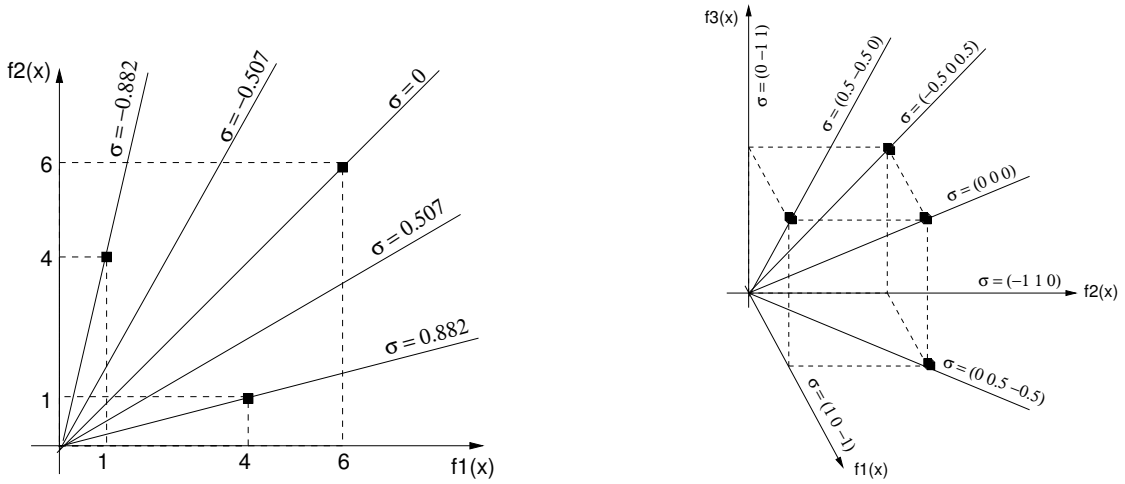
Considering higher dimensional objective spaces, Fieldsend and Singh's MOPSO just tries to guide the particles by particular members of the archive. This can also be observed in Figure 5.1 for a two objective example. The particles that have both of their objective values greater than $\vec{c}_1$ must select one of the archive members making the composite point $\vec{c}_1$, whereas for most of them, other guides coming from composite points $\vec{c}_2$ or $\vec{c}_3$ are more suitable. However, the aim of MOPSO is to let the particles move towards the best solutions, and in this method the particles are blocked by the composite points. This fact creates difficulties in attaining diversity of solutions, especially for high dimensional objective spaces. In the next section, another variant of MOPSO methods is studied, which avoids blocking particles when moving towards the Pareto-optimal front.

5.3.1 Sigma Method

In this section, a new method for finding the best local guide for each particle - called *Sigma method* - is proposed. Before explaining the method, the basic idea of the Sigma method is discussed in its general form. Later, finding the best local guide for each particle of the population in the objective space is explained.
In the Sigma method, a value σ_i is assigned to each point with coordinates $(f_{1,i}, f_{2,i})$ so that all the points which are on the line $f_2 = af_1$ have the same value of σ. Therefore, we can define σ as follows:

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{5.2}$$

According to Equation (5.2)[1], all the points on the line $f_2 = af_1$ have the same σ values: $\sigma_i = (1 - a^2)/(1 + a^2)$. Figure 5.2 (a) shows the values of σ for different lines. For the point with the coordinates $(f_{1,i}, f_{2,i})$, if $f_{1,i} = f_{2,i}$, $\sigma = 0$. In the case that $f_{2,i} = 0$, $\sigma = 1$ and in the case $f_{1,i} = 0$, $\sigma = -1$. Therefore, when $a > 1$, σ is negative and when $a < 1$, σ has a positive value. Indeed, σ states the angle between the lines $f_2 = \frac{f_{2,i}}{f_{1,i}} f_1$ and $f_1 = f_2$.



(a) 2-objective

(b) 3-objective

Figure 5.2: Examples of the Sigma method

In the general case, let $\vec{\sigma}$ be a vector of $\binom{m}{2}$ elements, where m is the dimension of the objective space. In this case, each element of $\vec{\sigma}$ is the combination of two coordinates in terms of the Equation (5.2). For example, for three coordinates f_1, f_2, and f_3, it is defined as follows:

$$\vec{\sigma} = \begin{pmatrix} f_1^2 - f_2^2 \\ f_2^2 - f_3^2 \\ f_3^2 - f_1^2 \end{pmatrix} / (f_1^2 + f_2^2 + f_3^2) \tag{5.3}$$

Different values of $\vec{\sigma}$ for different values of f_1, f_2, and f_3 are shown in Figure 5.2 (b). In the general case, when a point has the same position in each dimension (e.g., $f_1 = f_2 = f_3$ in 3 dimensional space), $\vec{\sigma} = \vec{0}$.

[1]It must be emphasized that Equation (5.2) can have other definitions, e.g., $\sigma = \frac{f_1 - f_2}{f_1 + f_2}$. The important issue is that all the points lying on the line $f_2 = af_1$ must have the same σ value. The first idea of Equation (5.2) comes from the value of $\cos(\alpha) \cdot \sin(\alpha)$, where α is the angle between the lines $f_2 = f_1$ and $f_2 = af_1$

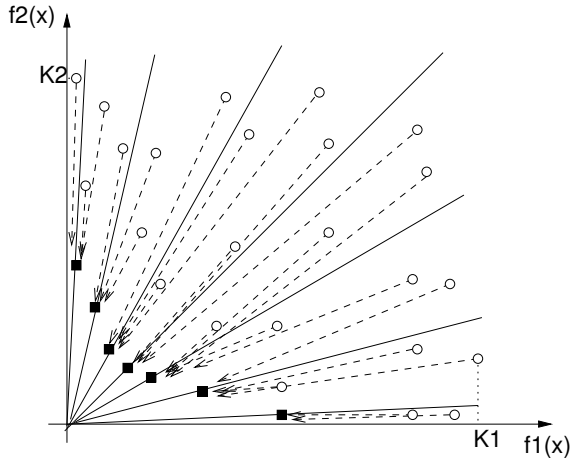


Figure 5.3: Finding the best local guide for each particle of the population using the Sigma method. (■: archive member, ○: particle of the current population)

Finding the Best Local Guides

By using the basic idea of the Sigma method and by considering the objective space, the best local guide ($\vec{p}_t^{\,i,g}$) among the archive members for particle i of the population, can be found as follows:

- For each particle j in the archive and particle i in the population, the values of σ_j and σ_i are calculated, respectively.
- Then, the distance between σ_i and σ_j, $\forall j = 1, \ldots, |A|$ must be computed.
- The particle k in the archive, where σ_k has the minimum distance to σ_i is selected as the best local guide for particle i.

Therefore, the particle $\vec{p}_t^{\,i,g} = \vec{x}^k$ is the best local guide for particle i. In other words, each particle that has a *closer*[2] Sigma value to the Sigma value of the archive member, must select that archive member as the best local guide.
Figure 5.3 shows how we can find the best local guide among the archive members for each particle of the population for a 2-objective example.
The reason for selecting particle k from the archive members as the best local guide is that σ_k has the closest distance to σ_i among the archive members. When two σ values are close to each other, it means that the two particles are on two lines (e.g., $f_2 = a_1 f_1$ and $f_2 = a_2 f_1$in two-dimensional space) that are close to each other

[2]In the case of two dimensional objective space, *closer* means the difference between the Sigma values and in the case of m-dimensional objective space, it means the m-Euclidian distance between the Sigma values.

(there is just a small angle between them). When comparing the Sigma method to the method proposed by Fieldsend and Singh [FS02] (see Figure 5.1), we observe that the Sigma method lets the particles fly directly towards the Pareto-optimal front, whereas in [FS02] the particles are blocked by the composite points.
The algorithm of the Sigma method is shown in Algorithm 14. There, the function *Sigma* calculates the value of $\vec{\sigma}$ and *calcdist* computes the Euclidian distance between its inputs. In this algorithm, $\vec{y}_j$ denotes the objective vector of the jth element of the archive A and the set σ_a contains the Sigma vectors of the archive members.

Algorithm 14 : FindBestLocal Algorithm

Input: A, $\vec{x}^i$, $\sigma_a = \{\vec{\sigma}_{a,1}, \vec{\sigma}_{a,2}, \cdots, \vec{\sigma}_{a,|A|}\}$
Output: $\vec{p}^{\,i,g}$

Calculate $\vec{\sigma}_i$ for the particle i:
$\sigma_i = Sigma(\vec{f}(\vec{x}^i))$
$dist = calcdist(\sigma_{a,1}, \sigma_i)$
for $j = 2$ to $|A|$ **do**
  $tempdist = calcdist(\sigma_{a,j}, \sigma_i)$
  if $tempdist \leq dist$ **then**
    $dist = tempdist$
    $g = j$
  end if
end for

Discussion

The Sigma method can be applied to problems with an arbitrary number of objectives. For each particle in the space, a σ value must be calculated. However, it must be noticed that:

- By definition of the Sigma value in Equation (5.2), the Sigma method works for those MOPs which all of their objectives are either in the positive or negative part of the coordinate axis. For example, for a 2-objective MOP with $f_1 \in [0, 10]$ and $f_2 \in [-5, 2]$, we have to transform either f_1 to $[-10, 0]$ and f_2 to $[-7, 0]$ or f_2 to $[0, 7]$.

- In the Sigma method, it is desired to have the line with $\sigma = 0$, in the middle of the space (e.g., in 2-objective space, $f_1 = f_2$ line has an angle of $\pi/2$ to

f_1). This is possible when two objective functions are in the same ranges. In the case that the objectives are not in the same range, σ can be scaled. For example, for a two-objective space it can be changed to:

$$\sigma = \frac{(K_2 f_1)^2 - (K_1 f_2)^2}{(K_1 f_1)^2 + (K_2 f_2)^2} \tag{5.4}$$

where K_1 and K_2 are the maximum values of the first and second objective values of the particles in the population, respectively (Figure 5.3). The values of K_1 and K_2 can also be calculated from the maximum corresponding objective values of the particles in the current generation.

5.4 Turbulence Factor

To avoid local optima, a new factor called *turbulence* is added to MOPSO. This is known as *craziness* in PSO and is introduced by Kennedy and Eberhart [KE95]. Later, its name is changed to *turbulence factor* [FS02]. This parameter is similar to the mutation operator in EAs and is applied by adding a random value to the current position of each particle. The turbulence factor can be defined as follows:

$$x^i_{j,t} = x^i_{j,t} + R_T \; x^i_{j,t} \tag{5.5}$$

where R_T is a random value in $[-1, 1]$, which is added to the updated position of each particle with a probability. The probability, like the probability of mutation in EAs, has a very low value e.g., 0.01. However, it depends on the problem to be solved.

5.5 MOPSO vs. MOEA

Figure 5.4 shows a possible structure of a MOPSO method. This structure is valid for those elitist MOPSOs that store non-dominated solutions in an archive. Indeed, it shows different blocks of the algorithm, which consume most of the computational time. Compared to Figure 2.2 in Section 2.2.2, MOPSO and MOEA methods have several blocks in common, as well as some differences. The blocks *Calculating non-dominated solutions*, *Update*, and *Clustering* are related to the elitism, which are the same in both methods. The blocks *Evaluate* and *Move* refer to the main parts of the Algorithm 13. Comparing the simple structures of a MOEA with MOPSO in Sections 2.2.2 and 5.5, some major differences of the methods can be observed. In the following, the differences as well as the common parts of these methods are outlined.

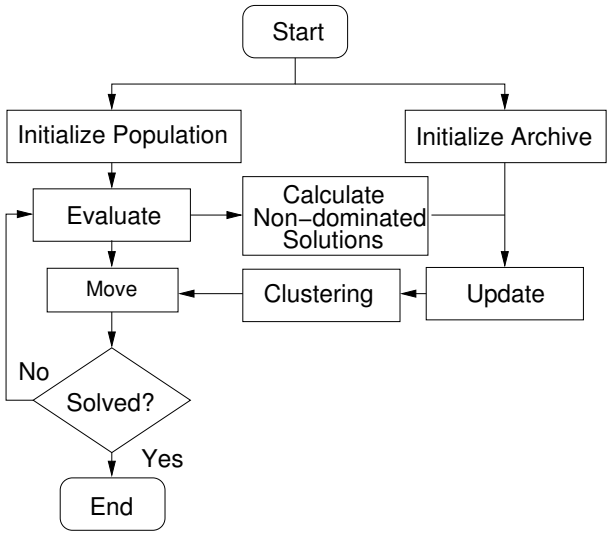


Figure 5.4: A possible structure of a MOPSO

Evaluation

In MOPSO there is no fitness evaluation like in MOEAs, e.g., in the SPEA2 method, strength and raw fitness values are calculated. Only the positions of the particles are evaluated in each generation.

Recombination

MOPSO does not have a cross-over operator. Acceleration towards personal and local best solutions is a similar concept. A mutation operator is necessary in MOEAs, as well as the turbulence factor in MOPSO. However, in MOPSO without a turbulence factor, any particle can eventually go anywhere in the space.

Selection

There is no selection in MOPSO; all particles survive until the last generation.

Interaction

In MOEA, there is interaction between randomly selected population members, whereas in MOPSO the topology is constant; a neighbor is a neighbor.

Archiving

Archiving is used in both of the methods. It helps to attain better convergence of solutions. The size of the archive is an important issue in MOPSO.

Parameters

Inertia weight is the only key parameter in MOPSO which must be selected carefully. In MOEAs, cross-over and mutation probabilities, the selection operator (the size of the mating pool) have a great impact on the solutions. However, in both of the methods, population size, archive size, and number of generations must be defined by the optimizer in the beginning.
The influences of the parameters on the MOPSO method are studied later in this chapter.

5.5.1 Archiving in MOPSO

The archive is used as an external population to store the non-dominated solutions. However, in MOPSO methods it is not only used as an archive to store the solutions, but also its members guide the population members towards the Pareto-optimal front. The same holds for MOEAs. The archive is also used in the selection process and it increases the probability of making better offsprings. However, in MOPSO, each archive member takes the responsibility of guiding the particles in its neighborhood. Therefore, the archive in MOPSO has more impact on the diversity and convergence of solutions than in MOEA. In other words, the good diversity of archive members leads the particles to a good diversity of solutions.
In the MOPSO methods, the initial archive is typically *empty*. In the first generation, the non-dominated solutions of the initial population are stored in the archive. Therefore, the particles of the population should select their best local guides among these archive members. Selecting the first local guides from the archive has a great impact on the diversity of solutions in the next generations, especially when using the Sigma method or the method proposed by Fieldsend and Singh [FS02, MT03b]. But if the initial archive is not empty and contains some well-distributed non-dominated solutions, the solutions converge faster than before, while keeping a good diversity. Figure 5.5 (a) shows an example of an initial population and the non-dominated particles among them, which are stored in the previously empty archive. In this figure, particles select one of these archive members as the local guide by using the Sigma method, and one can imagine that after one generation, particles will move towards the left part of the space. In Figure 5.5 (b), the initial archive was

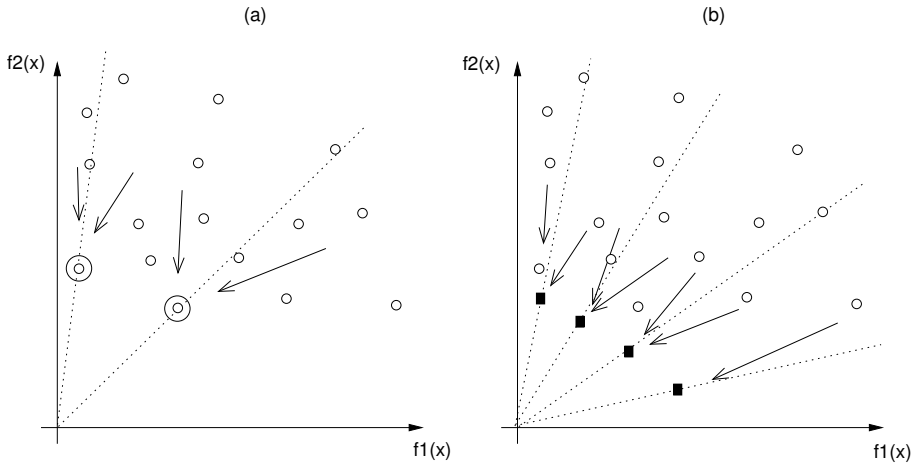


Figure 5.5: The influence of the initial archive (a) There is no initial archive. The particles must select one of the non-dominated solutions as the local best guide (b) The particles select one of the members of the initial archive (○: particle of the current population, ◎: non-dominated solution of the current population, ■: member of the initial archive)

chosen not to be empty, but it has some members which dominate all the particles in the population. This time, the particles will obtain a better diversity in the next generation than in Figure 5.5 (a).
Now, the question is how to find a good initial archive. One possibility is to run the MOPSO with an empty archive for a large population and few generations. The large population gives a good diversity and few generations (e.g., 5 generations) develop the population to just a little convergence. Another possibility is to use the results of a short MOEA (see Section 4.2). Here, short means a MOEA with few individuals and a small number of generations (e.g., 10 individuals and 10 generations).

5.6 Sigma Diversity Metric

In Section 2.5, different existing diversity metrics, such as the S metric, the Entropy approach, and the Sparsity measure are briefly studied. These methods compare the diversity of two non-dominated sets by calculating hyper-volume, or an entropy measure. However, as mentioned before, these measurements are suitable when comparing two sets and do not obtain information about a non-dominated set, itself. This is more important when the number of objectives is higher than 3, because the diversity of the solutions cannot be illustrated graphically. Indeed, for decision makers it is important to know where the solutions are located and how many percent

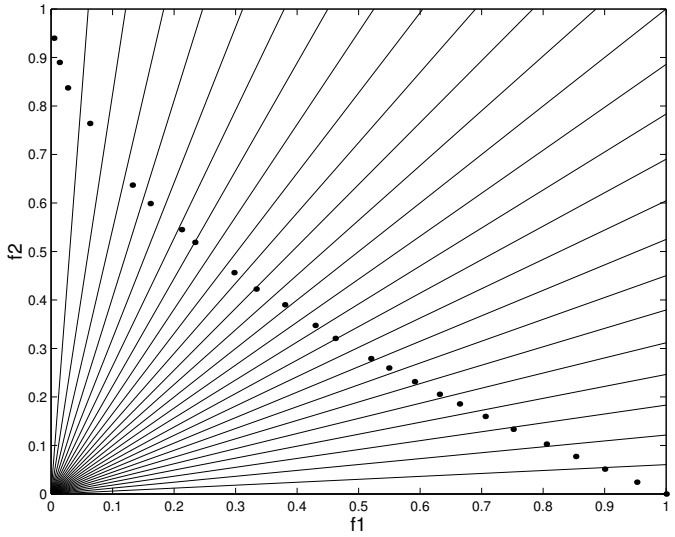


Figure 5.6: 2-objective Sigma Diversity Metric. Black points are the solutions of a 2-objective test function and the lines are reference lines

of the space is occupied by them. The disadvantages of the existing methods can thus be outlined as follows:

- High computational time, especially for high number of objectives (S metric).
- The methods can be used for comparing only two non-dominated sets. The hyper-volume does not give us information about the non-dominated set.
- Mapping of solutions does not reflect the exact diversity of convex non-dominated fronts (in Entropy and Sparsity measures).

Therefore, another diversity metric is studied here, which is inspired from the Sigma method. Indeed, the position of each solution in 2- and 3-objective spaces may be considered by a different coordinate axis, as well as by polar coordinates (r and θ) and spherical (r, θ and ϕ) coordinates, respectively. Inspired from these coordinates, we can formulate the diversity of solutions by a well distribution in terms of their angles θ for 2-objective spaces and θ and ϕ for 3-objective spaces. However, for higher dimensional objective spaces, we cannot define a coordinate axis which gives us a simple distribution, like in polar or spherical coordinates. Therefore, a new method is suggested to calculate the position of the solutions in the objective space, which uses the concept of the Sigma method. Actually, the Sigma method is easy to implement for any desirable number of objectives, and therefore it is easy to use it as a diversity metric. Figure 5.6 shows the idea of using the Sigma method

as a diversity metric for 2-objective spaces. In this figure, $|A|$ solution points are illustrated. Also, $|A|$ lines are drawn from the origin. A possible good diversity of solution is to have one solution on each line, or enclosed between two lines. But how can we find these lines for any number of objectives?

Reference Lines

As is shown in Figure 5.6, $k+1$ lines with different Sigma values are drawn from the origin. These lines are called *reference lines* and have the angle $\frac{i}{k}(\frac{\pi}{2})$ to the f_1-axis, where $i = 0, 1, \ldots, k$. We consider $k+1$ reference lines for computing the diversity of an archive with the size of $|A|$ ($|A| = k+1$). For higher dimensional spaces, the reference lines are also defined by lines passing through the origin. In order to find the angles between the lines and each coordinate axis, it is easier to find *reference points* located on the reference lines. Hence in an m-dimensional space, the coordinate of each reference point is a vector of m elements. Algorithm 15 calculates the coordinates of the reference points. For example, we obtain $(1, 0)$, $(1, tan(\frac{\pi}{6}))$, $(1, tan(\frac{\pi}{3}))$, for $m = 2$ and $k = 3$. Indeed in this algorithm, the first coordinate of point i is kept constant ($f_{1,i} = 1$) and the other coordinates are changing. However, to obtain the entire set of reference points, the algorithm must be repeated m times, each time one of the coordinates must be kept constant. In our example, we keep the second coordinate constant in the second run and obtain $(0, 1)$, $(tan(\frac{\pi}{6}), 1)$, $(tan(\frac{\pi}{3}), 1)$.

Algorithm 15 : Calculate coordinates of reference points

Input: $m,\ k$
Output: $(f_{1,i}, f_{2,i}, \ldots, f_{m,i}),\ \forall i = 1, 2, \ldots, I$

```
i = 1
for j_1 = 0 to k − 1 do
  for j_2 = 0 to k − 1 do
    ...
    for j_{m−1} = 0 to k − 1 do
      f_{1,i} = 1
      f_{2,i} = f_{1,i} tan(π j_1/2k)
      f_{3,i} = f_{1,i} tan(π j_2/2k)
      ...
      f_{m,i} = f_{1,i} tan(π j_{m−1}/2k)
      i = i + 1
    end for
  end for
end for
I = i
```

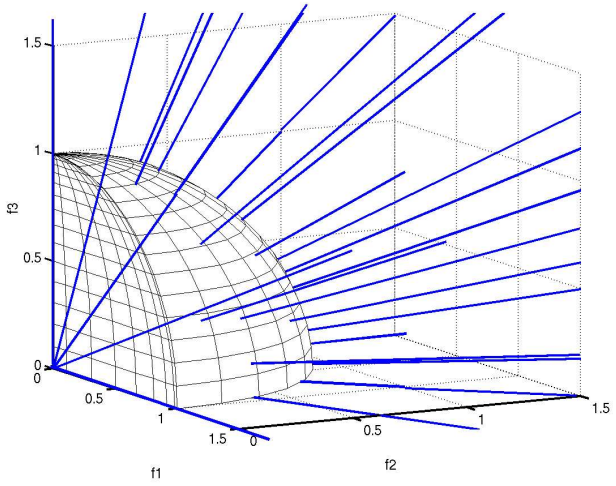


Figure 5.7: Reference lines in 3-objective space ($k = 4$)

The number of reference points produced by Algorithm 15 depends on k and m. In 2-objective spaces, the number of reference lines is equal to $k + 1$. In higher dimensional spaces k is the number of regions, which are separated by reference lines on the plane generated by only two of the coordinate axes. For example, in Figure 5.7, there are four regions separated by reference lines on the plane generated by the f_1- and f_3-axes. In higher dimensional spaces, the number of reference points made by Algorithm 15 is more than the required number of reference lines, because by repeating the algorithm, some points lie on the same reference line. In the previous example, the points $(1, tan(\frac{\pi}{6}))$ and $(tan(\frac{\pi}{3}), 1)$, and $(1, tan(\frac{\pi}{3}))$ and $(tan(\frac{\pi}{6}), 1)$ are on the same line. Therefore, the number of reference lines can be calculated after finding the Sigma value (vector) of each reference point (this will be explained later). Table 5.1 shows the number of reference lines for different values of k in 3-objective spaces.

Diversity Metric

The Sigma diversity metric can easily be computed as follows:

- Find reference lines according to Algorithm 15.
- Compute the Sigma value of each reference line (*reference Sigma value*) according to Section 5.3.1. The points located on a reference line will have the same Sigma vectors and the number of reference lines is the number of non-repeated reference Sigma vectors.

Table 5.1: k is the number of regions separated by reference lines on the plane generated by only two of the coordinate axes, number of ref. is the number of reference lines, and d is the radius of the neighborhood defined around each reference line.

k	number of ref.	d
4	25	0.15
6	67	0.1
8	133	0.1
10	223	0.1
12	337	0.1
14	475	0.05
16	637	0.05
18	823	0.05
20	1033	0.05

- Keep a binary *flag* beside each reference Sigma vector. The flag is 0 in the beginning. The flag of each reference Sigma vector can only turn to 1, when at least one solution has an equal Sigma vector or an Euclidian distance less than d, to it. The value of d depends on the test function, however it should decrease by increasing the number of reference lines. Table 5.1 shows an example of choosing d for a continuous 3-objective test function.

- A counter C counts the reference lines with flags equal to 1 and the diversity metric D becomes:

$$D = \frac{C}{\text{number of reference lines}} \tag{5.6}$$

The Sigma diversity measurement expresses the percentage of the space that is occupied by the found non-dominated solutions. This metric is easy to implement and it is easy to compute the diversity of solutions in high dimensional spaces. The 2-objective Sigma diversity metric seems to have some similarities to the Entropy approach [FMA02] and the Sparsity measurement [DMM03a, KYD03] (see also Section 2.5), especially when measuring the diversity of convex objective fronts. But in comparison to them, it is easy to calculate the diversity of solutions in high dimensional spaces. The Sigma diversity metric, like the Sigma method, can also be scaled for different ranges of the objective values. However, the objective values must only contain the positive values, and the negative values must be transferred

to the positive part (i.e., upper right quadrant of a circle in two dimensions). This is possible without any loss of generality.

5.6.1 Median Sigma Value ($\tilde{\sigma}$)

Another valuable information that can be obtained from the Sigma method is to find the position of the solutions by using a simple measurement. In other words, we know from D in Equation (5.6), that the solutions are distributed along the non-dominated front with D percent. If the value of D is high, it means that the solutions are well distributed. But when D is small, it means that the solutions are either concentrated in one part of the space, or distributed in small groups along the front. Indeed, there is a difference between these two kinds of solutions. Figure 5.8 shows the difference between two sets of solutions with the same D values. But their diversities (spreads) are different, because the solutions have different positions. Using the concept of the Sigma method again, we can measure the positions of the solutions. Let's consider 2-objective spaces, first. As shown in Figure 5.2 (a), the Sigma value is changing from 1 to 0 and to -1, by changing the angle between the line $f_2 = a \cdot f_1$ and the coordinate axis f_1. We can use this property and find the *median* of the Sigma values of the solutions. If our solutions have good diversity and spread, then the median of their Sigma values is zero. Negative and positive values of the median mean that the solutions are concentrated in the left and right hand side of the front, respectively. However, the zero value of the median, is also valid when the solutions are all concentrated in the middle of the non-dominated front. The same idea applies for higher number of objectives as well. The line in the middle has the Sigma vector equal to $\vec{0}$ (see Figure 5.2 (b)). This time, the median vector must be considered.

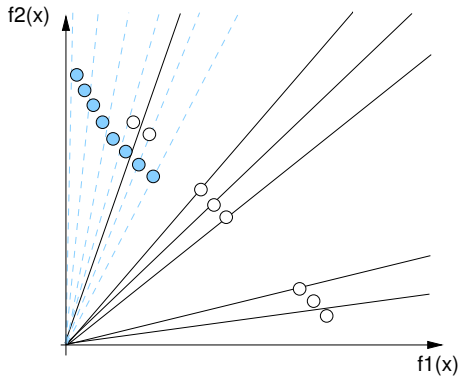


Figure 5.8: Different diversities (spreads) of solutions, but with the same D values

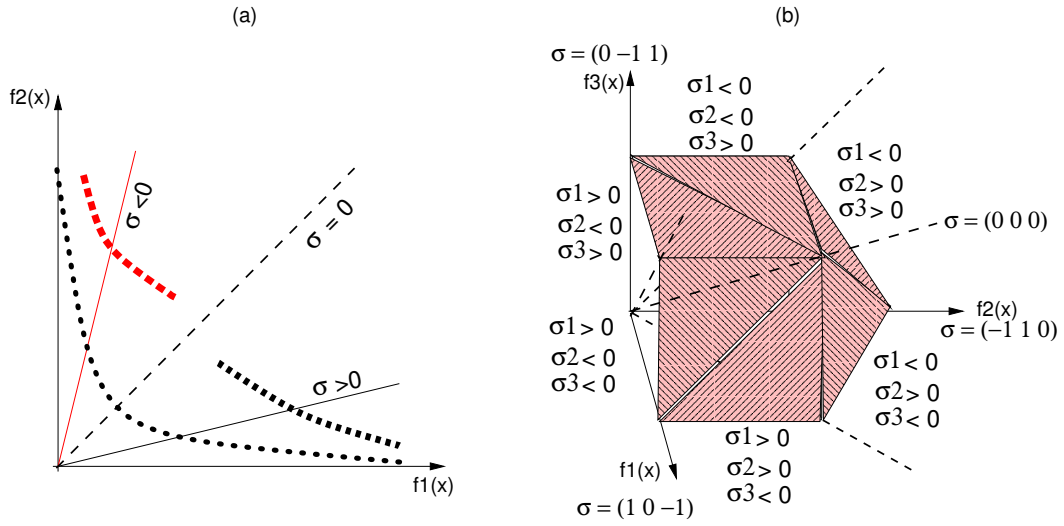


Figure 5.9: Properties of the median of the Sigma values in (a) 2- and (b) 3-objective spaces

Consider $\sigma = \{\vec{\sigma}_1, \cdots, \vec{\sigma}_{|A|}\}$ as the set of Sigma vectors of the solutions in the archive A. Then, the jth element of the median vector $\tilde{\vec{\sigma}}$ is defined as follows:

$$\tilde{\sigma}_j = \begin{cases} \sigma_{l,j}, & \text{if } |A| = 2l+1 \\ \frac{1}{2}(\sigma_{(l+1),j} + \sigma_{l,j}) & \text{if } |A| = 2l \end{cases} \tag{5.7}$$

Figure 5.9 shows different parts of the space for different median values for 2- and 3-objective spaces. In this figure, σ_i is the ith element of the Sigma vector. Calculating the median value is useful when the diversity measure D is small.

Discussion

The Sigma diversity metric and the median Sigma value deliver information about the diversity and spread of obtained solutions along the approximated Pareto-optimal front. The advantages of these measurements in comparison to other diversity and spread metrics such as the S metric, the Entropy, and Sparsity measurements, are as follows:

- Computing the metric is very easy for any desired number of objectives.
- They can deliver information about *one* approximated front, where the other methods compare two approximated fronts. Indeed, the output of the Sigma diversity metric is a percentage of the objective space; the other methods deliver a scalar value representing the hyper volume or Entropy measure, which are used to compare two sets.

- The median Sigma value computes the positions of the solutions in the objective space. This is a good metric for the objective spaces with high dimensionality, i.e., $m > 3$, where the graphical illustration of solutions is not possible.

Figure 5.10 illustrates an example of three different non-dominated sets with different spreads and diversities of solutions. In Figure 5.10 (a), the solutions are well-distributed. Therefore, both of the Sigma diversity measure and the median Sigma value are satisfactory, i.e., $D = 100\%$ and $\tilde{\sigma} = 0$. In Figure 5.10 (b), there is a large gap between the solutions and therefore, the Sigma diversity measure is less than the desired value. The positive median value indicates that most of the solutions are concentrated on the right hand side of the non-dominated front. In fact, the median value can be applied as a measurement of the spread of the solutions. In Figure 5.10 (c), solutions are not well-distributed. Therefore, the Sigma diversity measure is less than 100%. These solutions have better spread than those in Figure 5.10 (b). This is also indicated by the median Sigma value.
The Sigma diversity metric requires information about d, the neighborhood defined around each reference line. The value of d depends highly on the shape of the Pareto-optimal front. The reference lines must be computed once for each number of objectives and the corresponding values can be stored in a table.

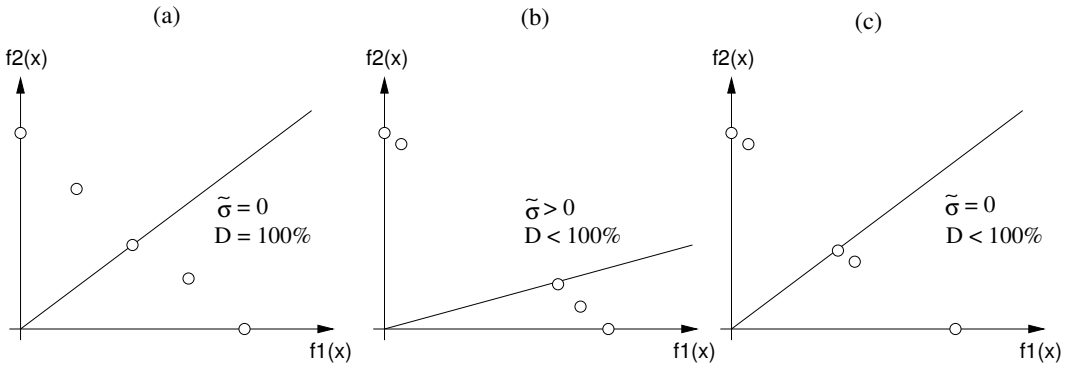


Figure 5.10: An example of different non-dominated sets with different diversity of solutions. (a) Solutions are well-distributed: $D = 100\%$ and $\tilde{\sigma} = 0$. (b),(c) Solutions are not well-distributed: $D \neq 100\%$. Median Sigma value, $\tilde{\sigma}$, indicates the spread of solutions.

5.7 Experiments

The experiments are applied on the 2-, 3- and 4-objective test functions from Table 2.1. Here, three kinds of MOPSO methods and one MOEA method are implemented for comparison. The MOPSO methods are Sigma, Dtree, and Random. Sigma and Dtree are implementations of the Sigma method and Fieldsend and Singh's method [FS02] of the MOPSO (outlined in Algorithm 13). The Random method is a simple implementation of a MOPSO. In this method, the local guides are chosen from the archive randomly. SPEA2 method is selected as the MOEA method (see Section 2.2.1) for more details.
Each experiment consists of 5 runs with different initial seeds and run on 500 MHZ SUN UltraSPARC IIe workstation. The mean values are recorded.
The comparison of different methods is based on the Sigma diversity metric and the C metric (for convergence). In the case that the archive size exceeds the maximum defined size, clustering is applied on the elite particles in the archive [Zit99]. The initial population consists of uniformly distributed particles in the variable space.

5.7.1 2-objective MOPs

2-objective test functions are selected from Table 2.1, i.e., ZDT1—ZDT6. The selected parameters for these test functions are:

- Inertia weight: 0.4,
- Turbulence factor: 0.07,
- Population size: 100 for ZDT6, 200 for ZDT1 and ZDT3, 300 for ZDT4,
- Number of generations: 200 for ZDT1 and ZDT3, 2000 for ZDT4 and ZDT6,
- Archive size: 50,
- Recombination: (for SPEA2 method) cross-over probability of 0.8 and mutation rate of 0.01.

Here, the termination criterion is defined as a maximum number of generations. For the test functions ZDT4 and ZDT6 the number of generations is increased to 2000. This is because the test function ZDT4 has 21^9 local minima and converging requires a high number of generations. Also, the test function ZDT6 has adverse density of solutions across the Pareto-optimal front, coupled with the non-convex nature of the front which make it difficult to converge.
Figures 5.11–5.14 show the results of the MOPSO techniques and SPEA2 on the test functions ZDT1-ZDT6, respectively.

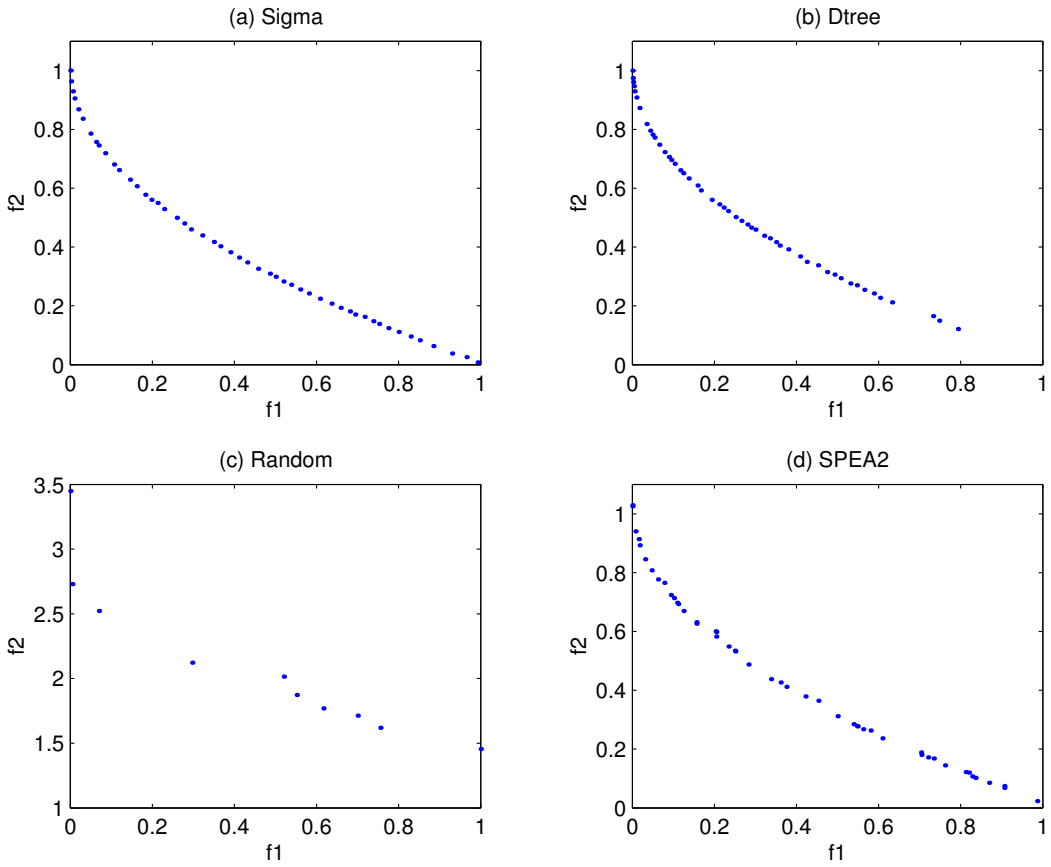


Figure 5.11: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function ZDT1 in Table 2.1

Comparison

Tables 5.2 and 5.3 summarize a quantitative comparison of the methods. Table 5.2 shows the comparisons of the diversity of solutions using the Sigma diversity metric D and the median Sigma value $\tilde{\sigma}$. Large values of D indicate a better diversity, and low absolute values of $\tilde{\sigma}$ show that the solutions are symmetrically distributed along the front. Table 5.3 shows the convergence comparison of different methods with the Sigma method using the C metric. Large values of $C(A, B)$ and low values of $C(B, A)$ mean that A has better solutions than B. However, when both of them have low values, it means that they have many indifferent solutions, which do not dominate each other.
In the following, each method is compared with other methods in terms of convergence and diversity of solutions:

- Random: As it is shown in Figures 5.11–5.14, Random cannot find solutions

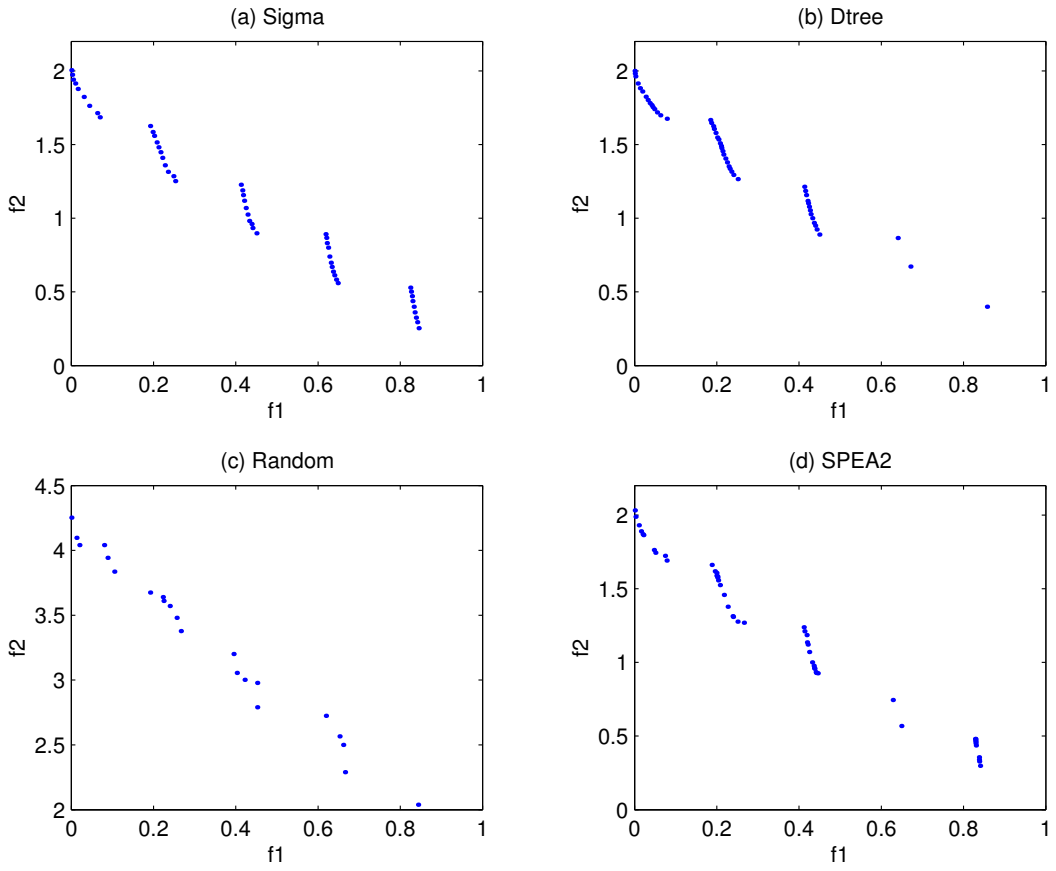


Figure 5.12: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function ZDT3 in Table 2.1

with good diversity and convergence like the other methods. The results of this method are clear enough in the figures, therefore no quantitative comparison is applied. This method cannot find solutions with acceptable convergence in a low number of generations.

- Sigma: This method finds solutions with very good diversities for all of the test functions. The high values of the Sigma diversity measure D, together with very low values of the median Sigma values show that the solutions are well-distributed and symmetric along the approximated Pareto-optimal front. The values of the C metric obtained for the test function ZDT4 expresses that the Sigma method outperforms the SPEA2 and Dtree method. Also it outperforms the SPEA2 method for the test functions ZDT1, ZDT3, and ZDT6.

- Dtree: High values of the median Sigma value shows that the solutions are

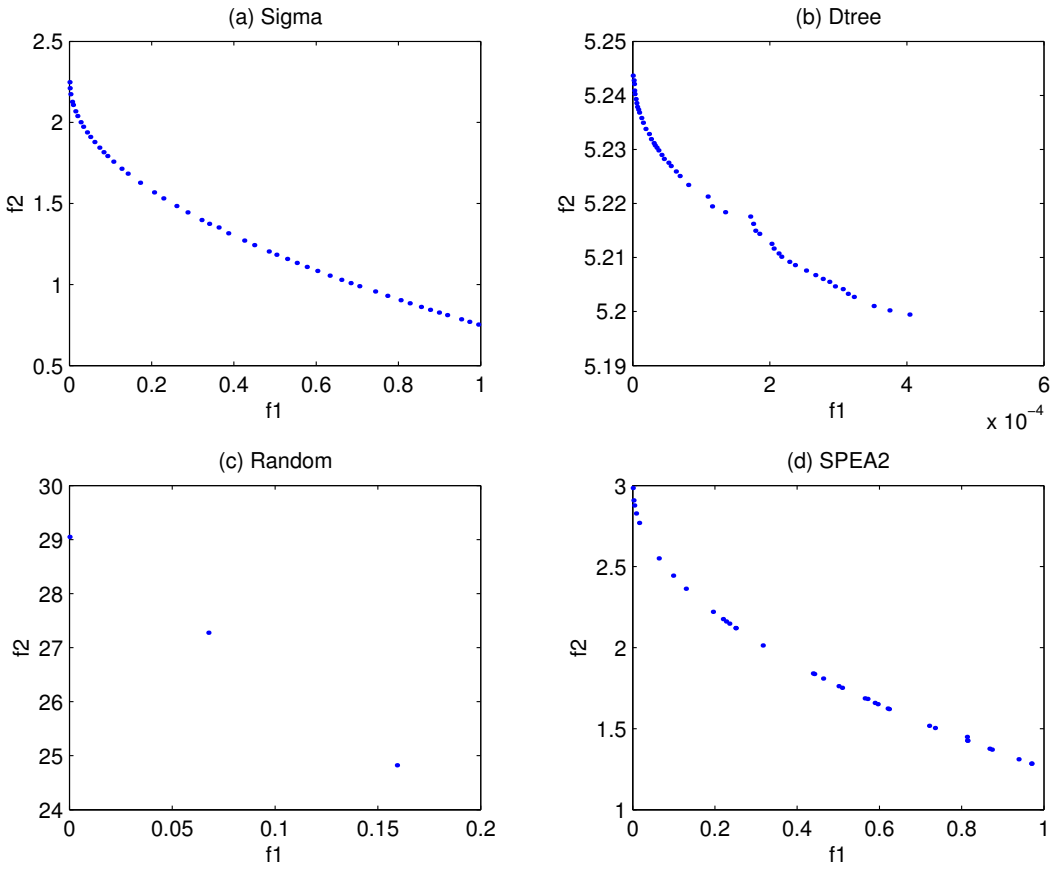


Figure 5.13: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function ZDT4 in Table 2.1

concentrated in one part of the space. This can also be observed in Figure 5.11–5.13. This method cannot obtain a good diversity of solutions. However, these solutions have better convergence than the solutions of the Sigma method for the ZDT1 and ZDT3 test functions. This can be observed in the convergence comparisons. Dtree method concentrates only on one part of the space, so it cannot find a good diversity of solutions. On the other hand, the Sigma method works on a wide area in the search space and finds non-dominated solutions in most parts of the search space.

- SPEA2: This method finds a good diversity of solutions for the ZDT1, ZDT3, and ZDT4 test functions. The results, however, are not as good as for the Sigma method. Also, all of these solutions have less convergence than the solutions of the Sigma method. The results for the ZDT6 test function have a worse diversity than the Sigma and the Dtree methods.

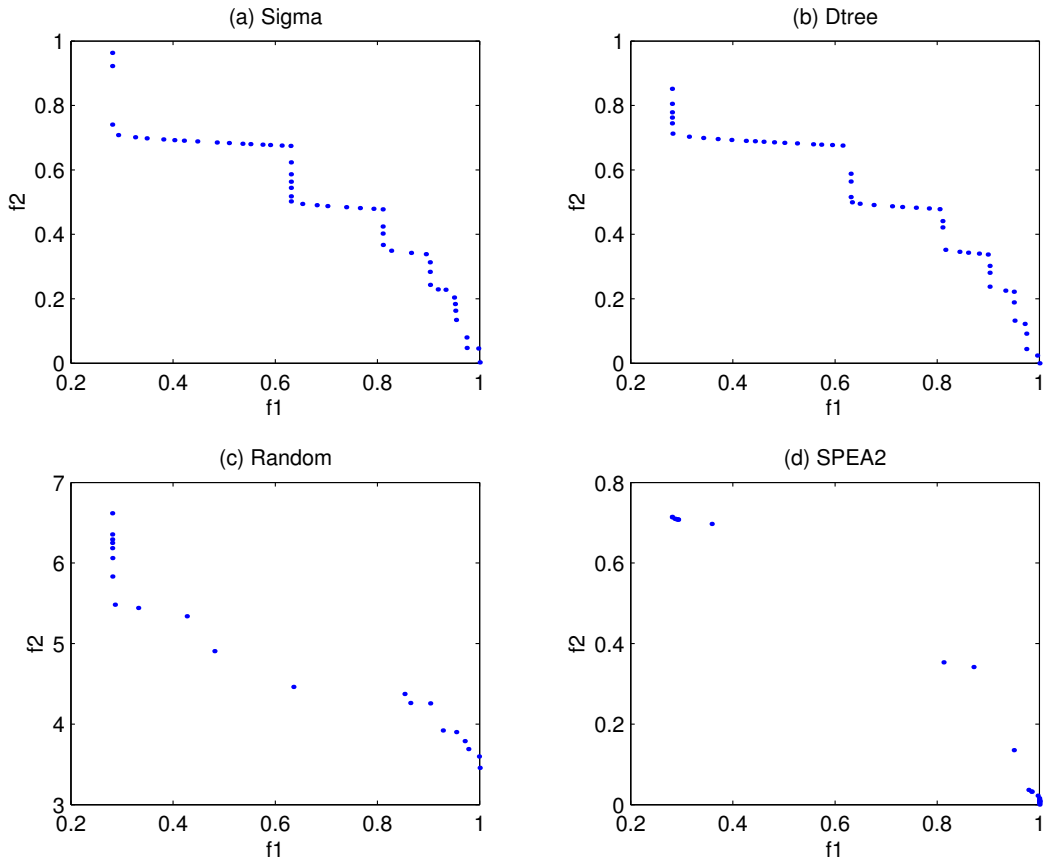


Figure 5.14: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function ZDT6 in Table 2.1

It must be emphasized that all of these methods are able to find solutions with high diversity and convergence when we increase the number of generations.

5.7.2 3-objective MOPs

The MOPSO methods are studied on the 3-objective GSP3, CP3, and DLZT test functions from the Table 2.1. Figures 5.15–5.17 show the results of the Sigma, Dtree, Random, and SPEA2 methods with the following parameters:

- Inertia weight: 0.4,
- Turbulence factor: 0.07,
- Population size: 200,
- Number of generations: 300,
- Archive size: 100 for GSP3 and CP3, 200 for DLZT,

Table 5.2: Diversity measures of the 2-objective test functions, S: Sigma, Dt: Dtree, R: Random, and SP: SPEA2 method (D: Diversity metric in percent, $\tilde{\sigma}$: median Sigma value)

Test	D_S	$\tilde{\sigma}_S$	D_{Dt}	$\tilde{\sigma}_{Dtree}$	D_{SP}	$\tilde{\sigma}_{SP}$
ZDT1	84%	+0.07	78%	-0.63	68%	-0.13
ZDT3	66%	-0.20	32%	-0.84	50%	-0.37
ZDT4	74%	-0.58	6.0%	-0.99	44%	-0.54
ZDT6	80%	+0.29	74%	+0.28	20%	+0.99

Table 5.3: Convergence comparison of the 2-objective test functions, S: Sigma, Dt: Dtree, R: Random, and SP: SPEA2 method (C: C metric)

Test	$C(S, Dt)$	$C(Dt, S)$	$C(S, SP)$	$C(SP, S)$
ZDT1	0.04	0.14	0.76	0.0
ZDT3	0.1	0.2	0.32	0.02
ZDT4	1.0	0.0	1.0	0.0
ZDT6	0.1	0.2	0.0	0.1

- Recombination: (for SPEA2 method) cross-over probability of 0.8 and mutation rate of 0.1.

The selected test functions have continuous or disconnected-continuous Pareto-optimal fronts.

Comparison

As is shown in Figures 5.15–5.17, Tables 5.4, and 5.5, almost all three MOPSO methods find solutions with good diversity. The Sigma method finds solutions with better diversity than the other methods for the test function GSP3. The very low values of the median Sigma vectors express a symmetric spread of solutions on the approximated Pareto-optimal front. Here, the Random method also finds some good solutions. This is due to the low number of parameters. It means that the search space in these test functions is not so large (in comparison with the 2-objective test functions with 30 parameters). The SPEA2 method finds solutions with comparable convergence to the Sigma method, but with lower diversity. This can be observed for the test function GSP3 (Table 5.4). The convergence values of the solutions are equal for all of the MO methods.

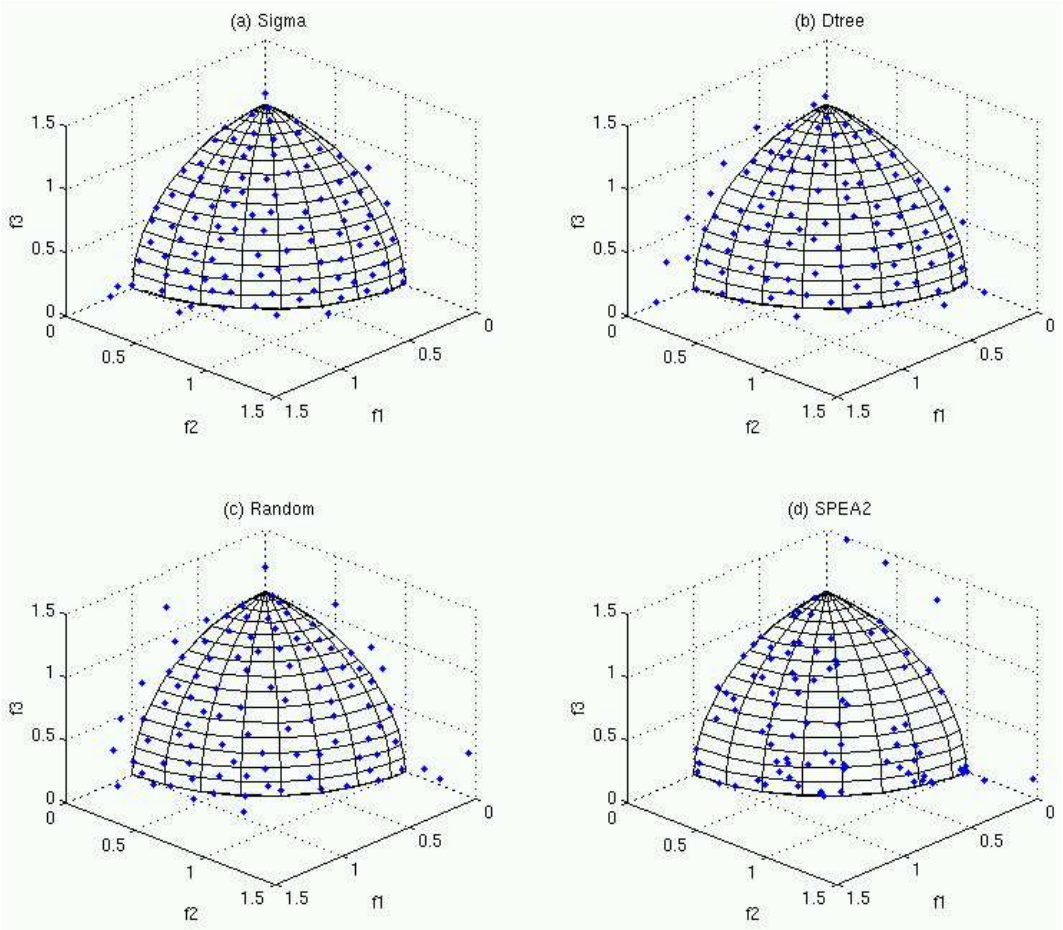


Figure 5.15: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function GSP3 in Table 2.1

Altogether, the Sigma method can find solutions with good diversity. The median Sigma values in Table 5.4 indicate that the solutions lie in the same parts of the space (see Figure 5.9) for all the MO methods. The $\tilde{\sigma}$ has all of its elements near zero for the test function GSP3. It means that all the solutions are symmetrically well-distributed on the front. It can be observed that for the other test functions the median Sigma value is not zero, because the solutions are just concentrated in one part of the space. The low values of C metric in Table 5.5 is due to the differences in the diversities of the solutions, and therefore, the solutions are incomparable.

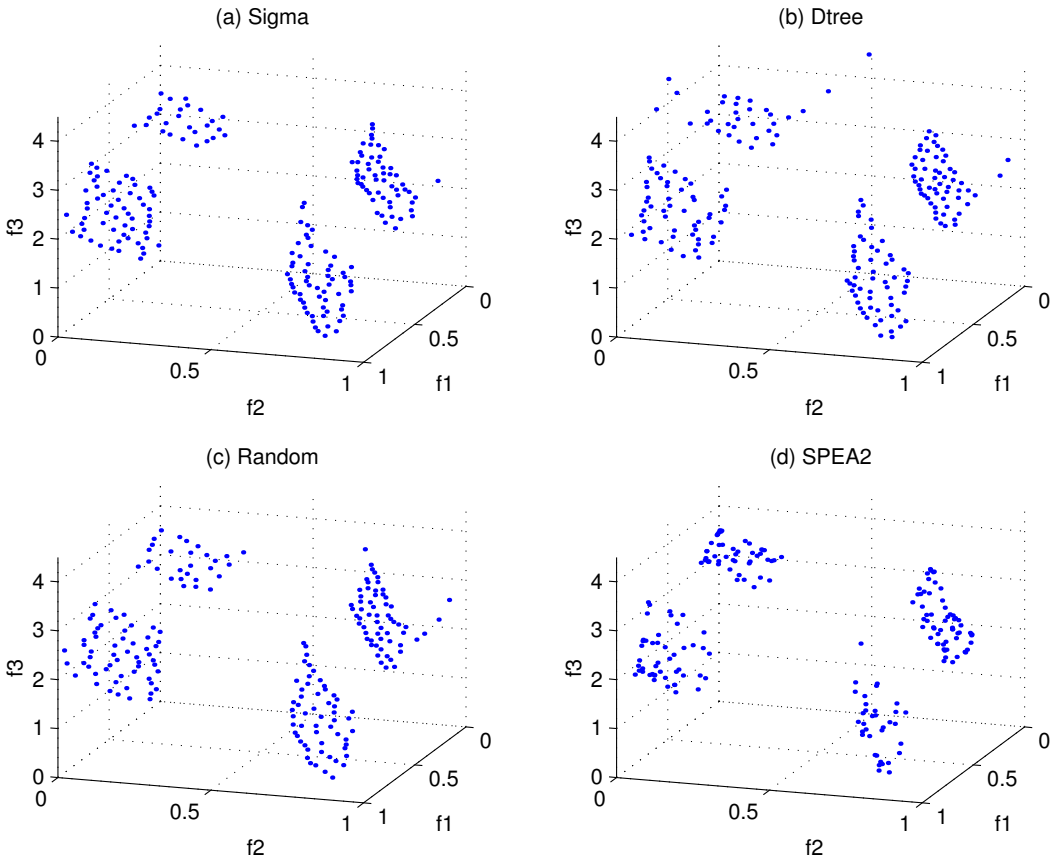


Figure 5.16: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function DLZT in Table 2.1

5.7.3 4-objective MOPs

The Sigma and the SPEA2 methods are studied on the 4-objective test function GSP4 from Table 2.1. The following parameters are selected:

- Inertia weight: 0.4,
- Turbulence factor: 0.07,
- Population size: 200,
- Number of generations: 300,
- Archive size: 100,
- Recombination: (for SPEA2 method) cross-over probability of 0.8 and mutation rate of 0.1.

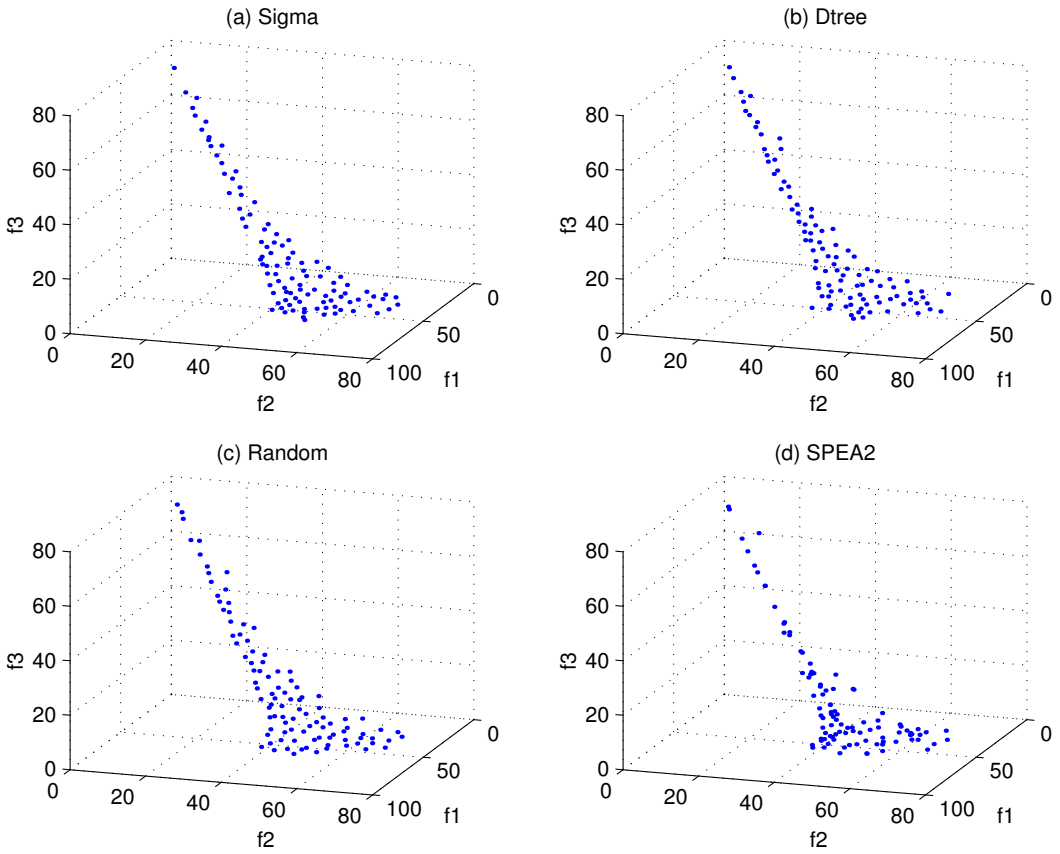


Figure 5.17: Results of the (a) Sigma, (b) Dtree, (c) Random, and (d) SPEA2 methods applied on the test function CP3 in Table 2.1

Table 5.6 shows the diversity values. The convergence comparison between these two methods shows that they both obtain indifferent solutions ($C(Sigma, SPEA2) = 0.02$ and $C(SPEA2, Sigma) = 0.0$).

Comparison

From Table 5.6, it can be concluded that the Sigma method finds solutions with higher diversity than the SPEA2 method. The $\tilde{\sigma}$ indicates that the solutions of the Sigma method are symmetrically well-distributed on the non-dominated front in the objective space. Because all the elements of the $\tilde{\sigma}$ are close to zero. Also, the SPEA2 method cannot find solutions with high diversity. Also, the solutions are not symmetrically distributed on the non-dominated front. The convergence comparison indicates that both of the methods find solutions, which do not dominate each other. This arises from the differences in the diversity of solutions.

Table 5.4: Diversity measures of the 3-objective test functions, S: Sigma, Dt: Dtree, R: Random, and SP: SPEA2 method (D: Diversity metric in percent, $\tilde{\sigma}$: median Sigma value)

Test	D_S	$\tilde{\sigma}_S$	D_{Dt}	$\tilde{\sigma}_{Dtree}$	D_{SP}	$\tilde{\sigma}_{SP}$
GP3	88%	$\begin{pmatrix} +0.01 \\ -0.00 \\ -0.01 \end{pmatrix}$	77%	$\begin{pmatrix} +0.02 \\ -0.01 \\ +0.00 \end{pmatrix}$	71%	$\begin{pmatrix} +0.00 \\ +0.02 \\ -0.00 \end{pmatrix}$
DLZT	19%	$\begin{pmatrix} -0.00 \\ -0.81 \\ +0.80 \end{pmatrix}$	22%	$\begin{pmatrix} -0.00 \\ -0.82 \\ +0.82 \end{pmatrix}$	16%	$\begin{pmatrix} +0.00 \\ -0.83 \\ +0.84 \end{pmatrix}$
CP3	22%	$\begin{pmatrix} -0.00 \\ +0.38 \\ -0.37 \end{pmatrix}$	29%	$\begin{pmatrix} +0.00 \\ +0.35 \\ -0.36 \end{pmatrix}$	28%	$\begin{pmatrix} +0.03 \\ +0.38 \\ -0.42 \end{pmatrix}$

Table 5.5: Convergence comparison of the 3-objective test functions, S: Sigma, Dt: Dtree, R: Random, and SP: SPEA2 method (C: C metric)

Test	$C(S, Dt)$	$C(Dt, S)$	$C(S, SP)$	$C(SP, S)$
GP3	0.06	0.04	0.0	0.04
DLZT	0.02	0.02	0.0	0.0
CP3	0.08	0.02	0.02	0.1

5.7.4 Parameter Setting of a MOPSO

Selecting parameters such as population size, archive size, number of generations, turbulence factor, and inertia weight is a confusing part of the optimization method. However, MOPSO methods have less number of parameters compared to MOEAs. For example, there is no mating pool, cross-over, and selection in MOPSO techniques. All the parameters have a great impact on convergence and diversity of solutions and like the MOEAs, it is difficult for the user to fix the parameters. In this section, the influence of each of the parameters on the convergence and diversity of solutions is studied. The tests are performed on the two test functions ZDT1 and ZDT3 from Table 2.1. These functions have 30 parameters. The influence of the parameters are tested as follows:

The set A of solutions is found by selecting parameters as before (population size = 200, number of generations = 200, archive size = 50, turbulence factor = 0.07,

Table 5.6: Diversity measures on the 4-objective test function GSP4 (D: Diversity metric in percent, $\tilde{\sigma}$: median Sigma value)

method	D	$\tilde{\sigma}^T$
Sigma	79%	$(+0.00, -0.00, +0.01, -0.02, -0.00, -0.00)$
SPEA2	62%	$(-0.00, -0.14, +0.12, +0.06, +0.03, +0.18)$

and inertia weight = 0.4). Then the tests are run with different parameters, and the results are stored in the set B.

Influence of Population Size

By increasing the population size, the diversity of solutions increases. This is valid for all the MOPSO methods. However, the computational time increases as well. The convergence of the solutions depends on the population size and at the same time on the turbulence factor. For higher values of turbulence factor and low number of particles in the population, it is not possible to obtain good convergence of solutions in a limited number of generations. This is clear, because a high value of turbulence factor makes many particles change their positions to find the global optimum. Therefore, it takes time for them to converge. The population size depends also on the MOP, number of parameters, and the number of objective functions.

Table 5.7: Different population sizes

test	size	$C(A,B)$	$C(B,A)$	$D(B)$	$\tilde{\sigma}_B$
ZDT1	50	0.5	0.02	66%	-0.80
	100	0.02	0.2	78%	-0.41
	150	0.0	0.26	82%	+0.04
	200(A)	–	–	84%	+0.07
ZDT3	50	0.56	0.06	44%	-0.85
	100	0.2	0.1	42%	-0.86
	150	0.18	0.14	42%	+0.03
	200(A)	–	–	66%	-0.20

Table 5.7 shows the diversity and convergence comparisons for the test functions ZDT1 and ZDT3. In this table, A means the results of the Sigma method with population size 200 and B is the method with various population sizes. Also, in this table the diversity of solutions increases by increasing the population size. The convergence comparisons indicate that with low values of C(Sigma, Dtree) and C(Dtree,

Sigma), the solutions are indifferent to each other, i.e., both of the Sigma and Dtree methods find solutions with the same convergence.
The size of the population also has an upper limit, which depends on the test function. The reason for having an upper bound is as follows. In the first generation, we have uniformly distributed solutions in the search space. If we increase the population size, there will be some solutions on the Pareto-optimal front which dominate the other solutions. It means that the dominated solutions must leave their positions and move towards the non-dominated solutions. Therefore we loose diversity.

Influence of Archive Size

The existence of the archive increases the convergence. In the Sigma and Dtree methods, the archive must have a fixed size: If the archive is not restricted, the selection process (selecting the best local guide) of each particle of the population will take a lot of computational time. On the other hand, the restricted archive size helps us to obtain a good diversity of solutions. If we just keep a fixed size of well-distributed solutions in the archive, then the particles will also obtain a better diversity in the next generations. The archive size depends on the population size and turbulence factor. The size of the archive also depends on the decision maker, since the archive is the output of the optimization method. For 2-objective MOPs, low values of archive size (e.g., 30 to 50) lead the population to a good diversity and also convergence (Table 5.8). However, for obtaining good convergence, the number of generations must be increased. For higher number of objectives, the archive size must be increased, e.g., to 100 for 3-objective MOPs.

Table 5.8: Different archive sizes

test	size	$C(A,B)$	$C(B,A)$	$D(B)$	$\tilde{\sigma}_B$
ZDT1	10	1.0	0.0	80%	+0.01
	30	0.23	0.04	76%	-0.32
	50(A)	–	–	84%	+0.07
	70	0.16	0.0	80%	+0.07
	90	0.05	0.34	64%	-0.89
ZDT3	10	1.0	0.0	60%	-0.60
	30	0.0	0.14	68%	-0.35
	50(A)	–	–	66%	-0.20
	70	0.11	0.2	41%	-0.83
	90	0.16	0.34	40%	-0.84

Influence of Turbulence Factor

The turbulence factor aims to avoid local optima. In MOPSO methods, a high value of turbulence decreases the convergence of solutions in a limited number of generations. On the other hand, very low values of the turbulence factor decrease the diversity as well as the convergence. Also, it highly depends on the MOP, the number of parameters, and objectives. Proposed values are between 0.01 and 0.07. Table 5.9 shows the diversity and convergence comparisons when varying the turbulence factor. This table indicates that the diversity and convergence of

Table 5.9: Different turbulence factors; tf denotes the turbulence factor

test	tf	$C(A,B)$	$C(B,A)$	$D(B)$	$\tilde{\sigma}_B$
ZDT1	0.07(A)	–	–	84%	+0.07
	0.025	0.0	0.14	64%	-0.82
	0.015	0.28	0.02	74%	-0.61
	0.01	0.3	0.0	82%	+0.15
ZDT3	0.07(A)	–	–	66%	-0.20
	0.025	0.1	0.08	68%	-0.29
	0.015	0.0	0.18	66%	-0.25
	0.01	0.1	0.1	30%	-0.89

solutions decrease when the turbulence factor decreases. However, high values of the turbulence factor have the same effect. It means that the turbulence factor is necessary to find a global optimum. But its value must not be more or less than a certain limit.

Influence of Inertia Weight

Inertia weight is a parameter, which controls the exploration of the search space. Indeed, it considers the effects of the previous velocities of the particles on the new velocities. Very low values of inertia weight makes the methods very slow and the particles search only their neighborhoods. In other words, it forces the particles to follow their previous paths. The proposed value is a real value between 0 and 1. Here, 0.4 is considered. Table 5.10 shows the results of various inertia weights. Very low values of the w (e.g., 0.0 and 0.2) lead to high diversity of solutions with very low convergence, and very high values (e.g., 0.8 and 1.0) lead to high convergence and low diversity of solutions. High absolute values of the $\tilde{\sigma}$ for $w = 1.0$ in Table 5.10 indicate that the solutions are all concentrated on the upper left part of the objective space.

Table 5.10: Different inertia weights; w denotes the inertia weight

test	w	$C(A,B)$	$C(B,A)$	$D(B)$	$\tilde{\sigma}_B$
ZDT1	0.0	0.96	0.0	84%	-0.02
	0.2	0.64	0.0	84%	-0.14
	0.4(A)	–	–	84%	+0.07
	0.6	0.02	0.12	50%	-0.93
	0.8	0.02	0.3	64%	-0.89
	1.0	0.0	0.3	80%	-0.16
ZDT3	0.0	0.84	0.0	62%	-0.57
	0.2	0.28	0.0	68%	-0.32
	0.4(A)	–	–	66%	-0.20
	0.6	0.1	0.22	48%	-0.84
	0.8	0.12	0.14	48%	-0.81
	1.0	0.04	0.38	66%	-0.77

However, these solutions have better convergence than other solutions obtained from lower w values.

Influence of Number of Generations

The number of generations is the simplest parameter in a MOPSO. The higher the number of generations, the more convergence of solutions we obtain. However, the desired convergence and diversity of solutions can be obtained after some generations when we have a restricted archive size.

5.8 Covering Pareto-fronts by MOPSO

A multi-objective optimization problem is mathematically solved, when all its Pareto-optimal solutions are found. Indeed, the goal of MO methods is to find the set of optimal solutions in one simulation run, in contrast to classical optimization methods e.g., weighting sum methods. However, it is impossible to find the entire set of Pareto-optimal solutions of a continuous front. Indeed, the size of the output, which in our case is the archive, is always kept constant. Restricting the size of the archive also has influences on the diversity of solutions and the computational time, as studied in the previous section. Therefore, the results obtained by most MOPSO methods have restricted amount of solutions in the output, by keeping a good diversity along the Pareto-optimal front. Diversity of solutions is studied by applying methods

like niching, clustering, or truncation, by several researchers [Deb01, Zit99, ZLT02]. These techniques often require a high computational time and at last we have a restricted number of solutions in the output [Zit99, ZLT02].
In most of the MOPSO and MOEA methods, the archive is restricted to a certain size. This is done because of the following reasons:

- Most of MO methods require a high computational time when the size of the archive increases (see Chapter 3).
- Diversity of solutions increases when the archive size is fixed, particularly in MOPSO.
- The computational time for finding the best local guides increases, if we store a high number of solutions in the archive.

Therefore, the solutions stored in the archive are just a selection of the approximated set of Pareto-optimal solutions. The MO methods try to keep a good diversity of solutions, so the decision maker has the possibility to access the entire approximated Pareto-optimal front. If we can find a good cover of the entire approximated Pareto-optimal front, in some cases, the task of the decision makers would be easier in selecting the desired solutions.
A new method for covering the approximated Pareto-optimal front is proposed here to perform the covering in low computational time. The MOPSO methods are able to find solutions with high convergence and diversity (see Section 5.2). One can use this knowledge and cover the approximated front as follows:

- Initial run: The MOPSO is run with a restricted archive size. We expect relatively well-distributed solutions very close to the Pareto-optimal front. The restriction on the archive can be achieved using the ϵ-dominance strategy presented in [MT03b] (see also Chapter 6).
- Covering: The non-dominated solutions obtained from the initial run are considered as the input archive of a new MOPSO called *covering MOPSO*. The covering MOPSO must have more particles in the population and no restriction on the archive size. The particles in the population are divided into subswarms around each non-dominated solution after the first generation. The task of the subswarms is to cover the gaps between the non-dominated solutions obtained from the initial run. Not having a restriction on the archive makes this method faster, because no clustering or truncation method is needed.

The initial run searches the entire space for obtaining good diversity of solutions. The covering is aimed to send subswarms of particles around the non-dominated

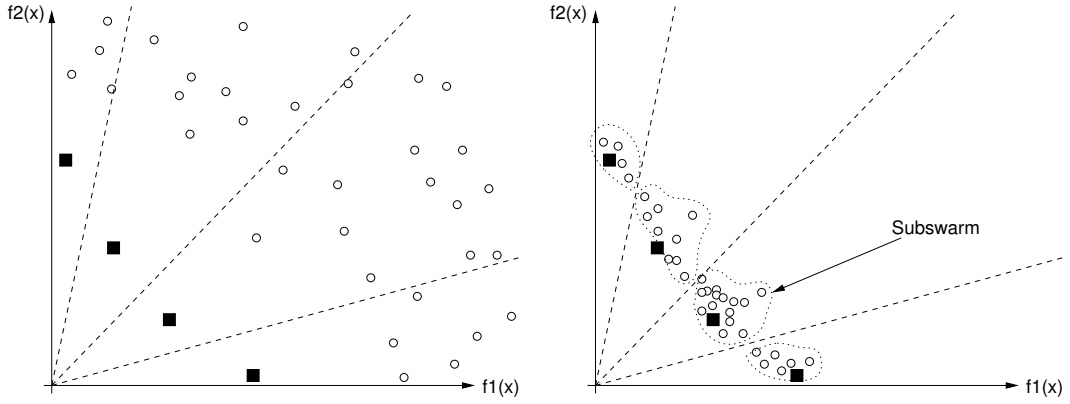


Figure 5.18: Covering the Pareto-front using initial archive members. The initial archive members guide subswarms of solutions around themselves to cover the Pareto-front. (○: Particles of the initial population of the covering MOPSO, ■: Result of the initial MOPSO)

solutions on the non-dominated front. Covering the front is then completed by these subswarms, which search only the neighborhood around each non-dominated solution. Figure 5.18 shows this scenario for covering the approximated Pareto-optimal front by MOPSO. The subswarms are generated by the Sigma method in finding the local best guides. All the particles in a subswarm have one common local best guide, which is in the non-dominated set made by the initial run. The size of each subswarm varies through generations and depends on the size of the initial population of the covering MOPSO.
It must be considered that any knowledge of the solutions helps the methods to find better solutions than before. In the covering process, the size of the population has a great impact in covering the front faster. The larger the population size, the more particles gather in a subswarm, and the front will be searched faster.

5.8.1 Experiments

Figure 5.19–5.22 show the results of covering the ZDT1, ZDT3, and ZDT6 test functions. The parameters are selected the same as in the last section for the initial run. The covering MOPSO has the following parameters:

- Inertia weight: 0.4,
- Turbulence factor: 0.07,
- Population size: 200,
- Number of generations: 500 for ZDT1 and ZDT3, 2000 for ZDT4 and ZDT6,

- Archive size: not restricted.

Covering the test function ZDT4 is not as simple as the other test functions. This test function has a lot of local Pareto-fronts. By running the covering MOPSO to cover the front, it is observed that the solutions are not already converged to the true Pareto-optimal front. In this case, the covering MOPSO improves the convergence of solutions. The solutions shown in Figure 5.22 (1st Run) are not actually on the true Pareto-optimal front, although the MOPSO method has obviously obtained higher converged solutions than the other methods, e.g., [Zit99]. One reason is the restricted number of generations. We have to notice that the obtained solutions from the first run have a good diversity. Therefore, these solutions are considered as the initial archive of another run and the output of this second run is given as the initial archive of the next run. This procedure is repeated several times, until the approximated Pareto-optimal front is found. Now the covering process can be started in order to cover the front.
Figures 5.22 (a) and (b) show different steps in optimizing the test function ZDT4. The reason for repeating the MOPSO instead of running it for a large number of generations is that when there are many local optima, the optimization method requires a lot of computational time to find solutions with high convergence and diversity of solutions, simultaneously. The initial archive helps the initial particles to be divided in small groups, and to search the space faster. The covering result in Figure 5.22 (b) shows the good potential of MOPSO (Sigma method) in solving very difficult MOPs, where the recorded results of the MOEA method [Zit99] are

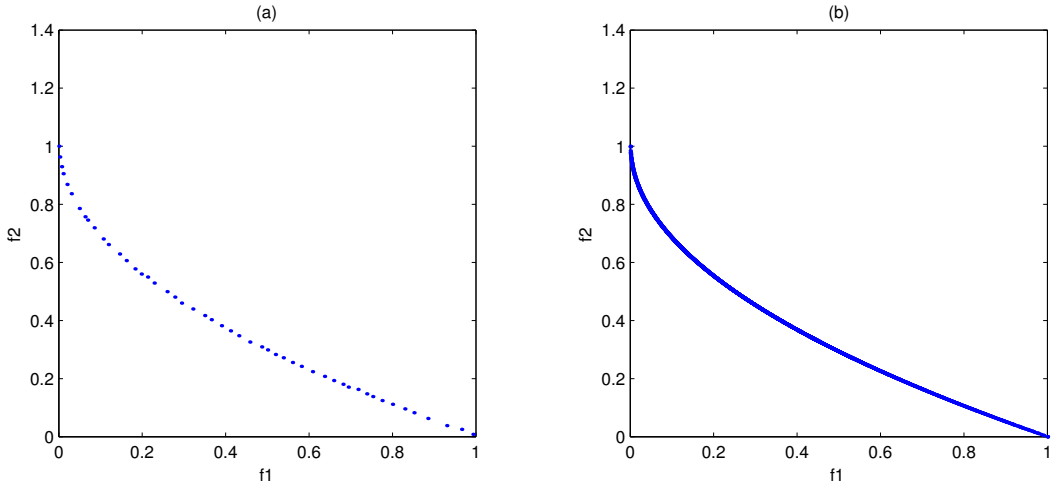


Figure 5.19: Solutions of the ZDT1 test function (a) with clustering (a = 50), (b) covered front

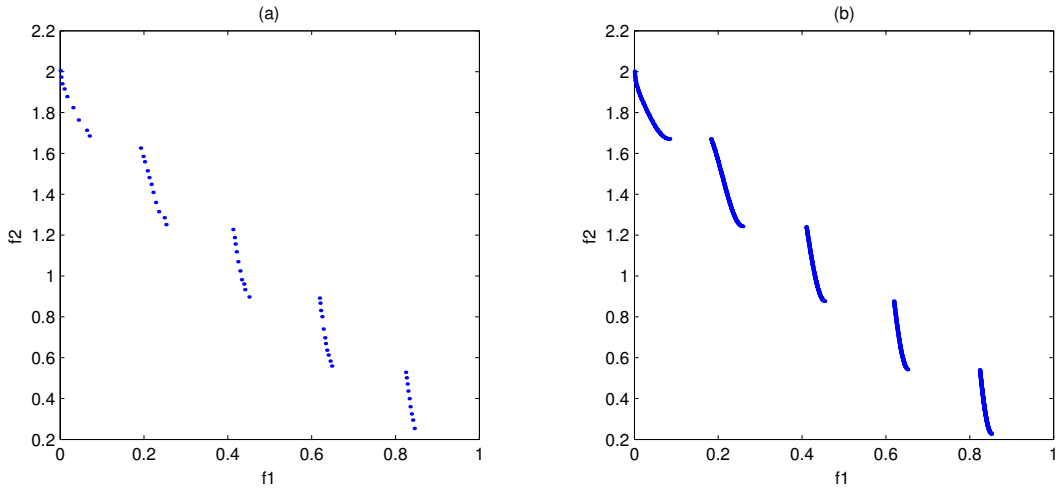


Figure 5.20: Solutions of the ZDT3 test function (a) with clustering (a = 50), (b) covered front

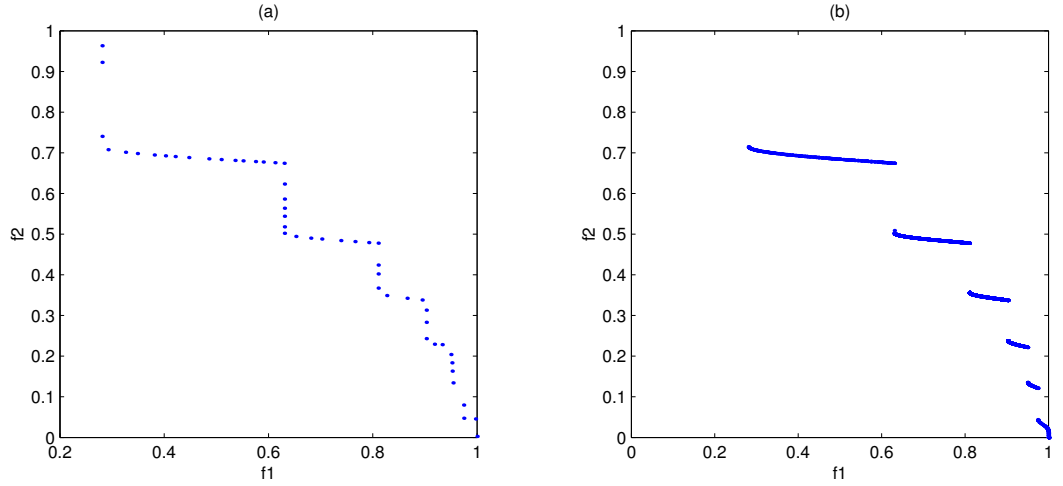


Figure 5.21: Solutions of the ZDT6 test function (a) with clustering (a = 50), (b) covered front

not so close to the true Pareto-optimal front.
For evaluation of the covered solutions, the Euclidian distance between two neighbor solutions is computed. Here, the covering means to find a set of finite solutions very close to each other. Therefore, one good metric is the distances between two neighbor solutions. Table 5.11 shows the minimum and maximum obtained distance between any pair of the covered solutions in the objective space. The computational times are also recorded in this table.

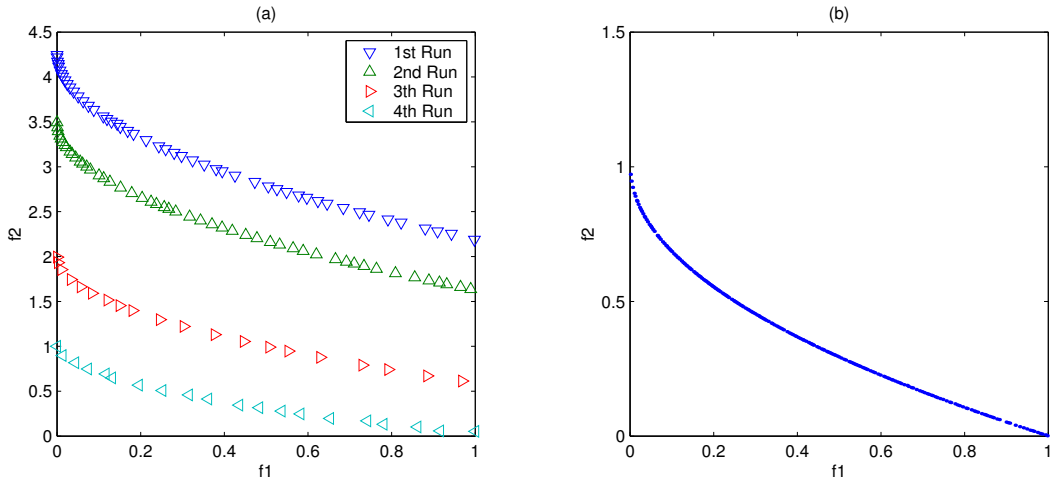


Figure 5.22: Solutions of the ZDT4 test function (a) with clustering, obtaining the converged front needs several runs, (b) covered front

Very low values of D_{max} show that the solutions are very close to each other. However, if we run the MOPSO for a larger number of generations than the recorded value, we obtain more solutions, and closer to each other. For obtaining faster results we can also apply the $\epsilon MOPSO$ algorithm (see Chapter 6). ϵ can be larger for the initial run, than for the covering process.

Discussion

Covering the Pareto-optimal front in low computational time is possible by using the covering MOPSO technique. Here, covering means to find a finite set of optimal solutions very close to each other. The proposed covering MOPSO method uses the property of moving particles in MOPSO and divides the population of the covering

Table 5.11: Computational time of initial (T_{init}) and covering (T_{cover}) runs of MOPSO, in seconds. D_{min} and D_{max} are the minimum and maximum distances between the non-dominated solutions of the approximated Pareto-front

test	T_{init}	T_{cover}	D_{min}	D_{max}
ZDT1	103.41	117.93	0.000	0.014
ZDT3	20.70	35.71	0.000	0.009
ZDT4	71.15	300.50	0.0024	0.025
ZDT6	17.08	500.37	0.000	0.007

MOPSO into subswarms by using the concept of the Sigma method. Each subswarm must contain a good amount of particles. Therefore, the population of the covering MOPSO must be considerably larger than that of the initial run, particularly for covering high dimensional approximated Pareto-optimal fronts. The subswarms try to cover the gaps between the non-dominated solutions found in the initial run. Here, the covering MOPSO method is tested with different test functions. However, it must be emphasized that this method is not only applicable to 2-objective multi-objective optimization problems, but also can be used for any desired number of objectives.

5.9 Conclusion

In this chapter, MOPSO methods as alternatives to MOEAs are studied. These methods are able to solve MOPs with high convergence and diversity of solutions. This is tested on different test functions and compared with the results of MOEAs. The following topics are investigated in this chapter:

- Finding local best guides in MOPSO is studied and a new strategy, called Sigma method, is developed. In comparison to its previously proposed MOPSO approaches, the Sigma method finds solutions with good convergence and diversity. This is examined on different 2-, 3-, and 4-objective test functions. The Sigma method is easy to implement and can be carried out on different MOPs with desired number of objectives and parameters.

- New diversity measures, namely the Sigma diversity metric and the median Sigma value, are introduced. These measurements provide information about the distribution of the solutions in the objective space. The output of the Sigma diversity metric is a percentage of the objective space, which is occupied by the set of the non-dominated solutions. The median Sigma value complements the information about the space and tells us where the solutions are. This measurement is very worthwhile for higher dimensional objective spaces.

- Three variants of MOPSO methods and also the SPEA2 (MOEA) method are tested on 2-, 3-, and 4-objective test functions with different numbers of parameters. The MOPSO methods are the Sigma method, Dtree method (best local guides are selected by using the method of the Fieldsend and Singh), and the Random method (best local guides are selected at random).

The results show that the Sigma method can find solutions with higher diversity and convergence than the other methods. The SPEA2 method finds good solutions in most cases with good convergence, but not a good diversity. In some cases, the Dtree method behaves the same as the Sigma method, but with a less diversity of solutions. The Random method needs a large number of generations to converge.

These results hold for all the recorded number of generations and population sizes, however, all of these methods can find solutions with good convergence and diversity after a large number of generations. We have to note that since the Sigma method can only work on positive values of the objective space, all the test functions must have positive objective values. Thus, a shift must be done and it requires an approximate prior knowledge of the test functions.

- The influence of different parameters are studied on the MOPSO methods:
  - Population size: Large population size increases the diversity of MOPSO methods. However, it increases the computational time.
  - Archive size: Elitism (existence of archive) in MOPSO, like in MOEAs, increases the convergence of the solutions. Fixed-sized archives drive the MOPSO to obtain solutions with a good diversity.
  - Turbulence factor: It avoids stoking in a local minimum.
  - Inertia weight: It controls the solutions to follow their previous paths.
  - Number of generations: For higher number of generations, we obtain better convergence of solutions.
- Covering the approximated Pareto-front is achieved by applying the covering MOPSO strategy. This method can only be used by a MOPSO algorithm. The obtained results show that the approximated Pareto-optimal front can easily be covered by MOPSO (here, covering means to find a set of finite solutions, where the solutions are to a certain value very close to each other).

Chapter 6

ϵ-MOPSO

In the previous chapters, archiving is used as an important part of MO methods. The archive must be kept domination-free and is therefore updated in each generation. The time needed to update the archive depends on several parameters, i.e., the archive size, population size, and the number of objectives and dramatically increases when increasing the values of these three factors (see Chapter 3). One possibility for reducing the computational time is to restrict the archive size and to use other available data structures to update the archive. Various data structures for the archive are studied in Chapter 3. However, these data structures also take lots of computational time for a large archive size. Therefore, it is required to restrict the archive size. Indeed, the existence of a restricted size archive in MOPSO methods causes a good diversity of solutions. On the other hand, from the decision maker's point of view, presenting the whole non-dominated set is useless when the size exceeds a certain bound. Therefore, it is demanded to fix the size of the archive.
There are several methods like clustering [Zit99], truncation [ZLT02], and crowding [Deb01] techniques to fix the archive size. Also, there is a recent work on using the Lebesgue measure to bound the archive [KCF03]. In order to keep a good diversity, these methods become the most time-consuming part during the update procedure.
In this chapter, the idea of ϵ-dominance as proposed in [PY00, LTDZ02] is used to fix the size of the archive to a certain amount. Using the ϵ-dominance in the context of MO is not new: Laumanns et al. [LTDZ02] have used it to increase the convergence of solutions. In this chapter, ϵ-dominance is used to bound the archive size. However, it also has influences on the convergence and diversity of solutions and the question is: what is the trade-off between computational time, diversity, and convergence of the solutions? Here, the concept of ϵ-dominance is integrated into MOPSO methods and compared with the clustering [Zit99] method.

6.1 Definitions

This section is dedicated to the definitions concerning domination and ϵ-domination. Required basic definitions are given in Section 1.1.

Definition 6.1 (ϵ-domination) *A decision vector $\vec{x}_1 \in S$ is said to ϵ-dominate a decision vector $\vec{x}_2 \in S$ for some $\epsilon > 0$ (denoted $\vec{x}_1 \prec_\epsilon \vec{x}_2$) if:*

- $f_i(\vec{x}_1)/(1+\epsilon) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m.$
- $f_i(\vec{x}_1)/(1+\epsilon) < f_i(\vec{x}_2)$ *for at least one* $i = 1, \ldots, m.$

Figure 6.1 shows the concept of ϵ-domination.

Definition 6.2 (ϵ-approximate Pareto-front) *Let $F \subseteq \mathbb{R}^m$ be a set of vectors and $\epsilon > 0$. The ϵ-approximate Pareto-front $F_\epsilon \subseteq F$ contains all vectors $\vec{x}_1 \in F$ which are not ϵ-dominated by any vector $\vec{x}_2 \in F$:*

$$\forall \vec{x}_2 \in F : \exists\ \vec{x}_1 \text{ such that } \vec{x}_1 \prec_\epsilon \vec{x}_2 \tag{6.1}$$

We have to note that the set F_ϵ is not unique, but contains just a certain amount of vectors, depending on the ϵ-value. This has been studied in [PY00, LTDZ02]. For any finite ϵ and any set F with objective vectors $1 \leq f_i \leq K,\ \forall i \in \{1, \ldots, m\}$, there exists a set F_ϵ containing *at most*:

$$|F_\epsilon| = O[(\frac{logK}{log(1+\epsilon)})^{m-1}] \tag{6.2}$$

vectors. Here, we consider that ϵ is the same for all objectives. Figure 6.2 illustrates an example of an ϵ-approximate Pareto-front in the objective space ($m = 2$). In this figure, the number of non-dominated solutions is bounded. Here, if $K1 = K2 = 2$ and $\epsilon = 0.08$, then the ϵ-approximate Pareto-front can have at most 9 solutions.

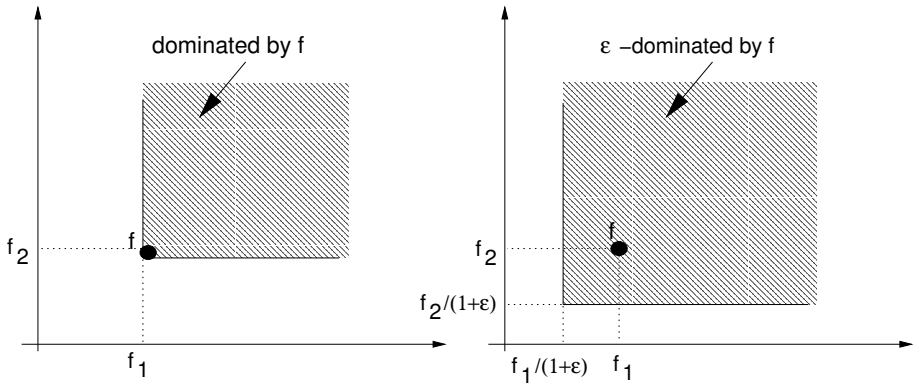


Figure 6.1: Domination and ϵ-domination in the objective space

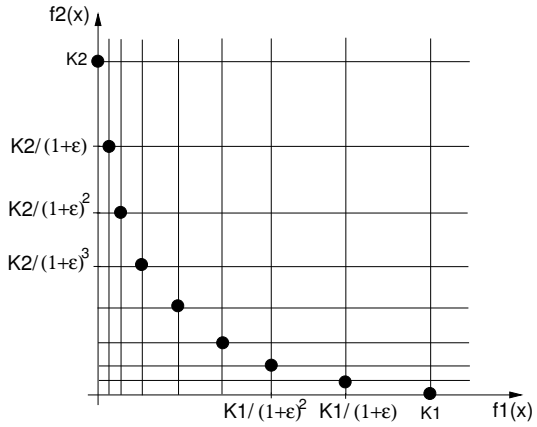


Figure 6.2: An example of ϵ-dominance

6.2 Bounding the Archive Size

The archive of most of the MO methods should contain a certain number of solutions, while keeping a good diversity of solutions, due to the following reasons:

- From the decision maker's point of view, presenting the entire non-dominated set is useless when the size increases a certain bound.
- Diversity of solutions increases when the archive size is fixed, particularly in MOPSO.
- In MOEA methods, non-restricted archives increase the selection pressure. In MOPSO methods, it is useless to store non-restricted solutions in the archive. In this case finding best local guides takes a long computational time. Indeed, a few well-distributed solutions in the archive are more useful than an unrestricted number of solutions.

Therefore, the archive of most of the MO methods is restricted, or fixed to a certain size. In the following, three variants of bounding methods are briefly reviewed.

6.2.1 Clustering

The main aim of the clustering method is to prune a non-dominated set and generate a representative subset, which maintains the characteristics of the original set. The clustering technique [Zit99] used in SPEA is a hierarchical clustering method, which works iteratively by joining adjacent clusters until the required size of the set is obtained. This method is first introduced by Morse [Mor80] and is named *average linkage method*. At the beginning, each solution in the set is a cluster. Then the two

clusters, which are closest to each other, are joined to make a bigger cluster. This is done iteratively, until the required number of clusters is achieved. The two clusters are selected according to the nearest neighbor criterion to make a new cluster. At the end, the solution with minimal average distance to other solutions inside each cluster is kept and the others are removed. This is shown in Figure 6.3.
As soon as the number of non-dominated solutions in the archive increases the fixed archive size, this clustering technique is used. However, we have to note that by increasing the size of the archive, the computational time increases. It has been shown in Chapter 3 that by increasing the number of objectives and population size, and particularly the archive size, the computational time of the MO methods increases dramatically.
In some MO methods (e.g., SPEA2 in Section 2.2.1), the archive has a fixed size and in the case that the number of non-dominated solutions is less than the fixed size, some particles of the population are selected at random to be inserted into the archive. In the case that the size of the set of non-dominated solutions becomes higher than the fixed size, an improved truncation technique is applied.

6.2.2 Lebesgue Archiving Hill Climber (LAHC)

In Section 2.5, Fleischer's algorithm is explained as a method to calculate the hyper-volume contribution of a single solution of a set of non-dominated solutions. Knowles et al. have used this idea and have proposed a method to bound the archive size [KCF03]. The principle behind their approach, called LAHC, is to remove the solution that has the least contribution to the hyper-volume. This is done whenever the archive is full and there is a new solution available to be added. This method is recorded to outperform the clustering method in some cases. However, the computational time is also another criteria which should be considered. Fleischer's algorithm is an efficient method of computing changes to the Lebesgue measure when one point

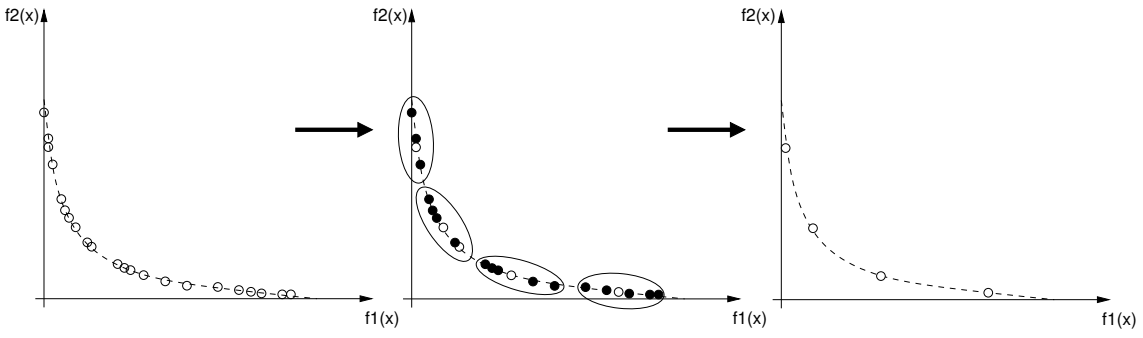


Figure 6.3: Clustering method [Zit99]

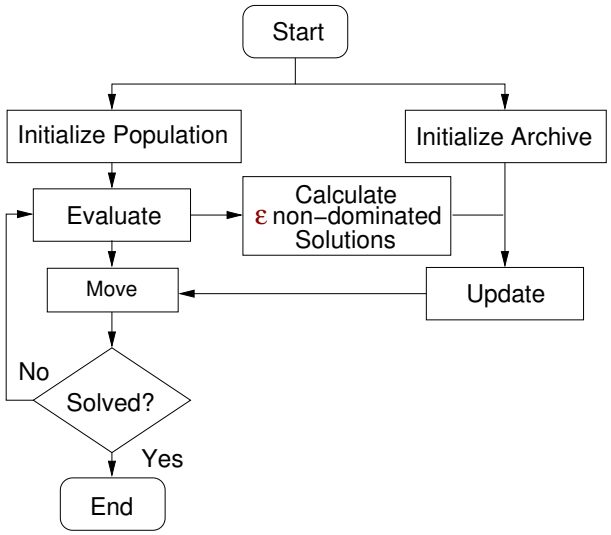


Figure 6.4: A typical structure of ϵMOPSO

is added or removed from a set. However, it is expensive to find the solution that has the least contribution to the hyper-volume, before inclusion of each new solution. Therefore, as is also mentioned in [KCF03], this method would be impractical in most of the applications for high dimensional ($m > 3$) objective spaces.

Discussion

Bounding the archive size by clustering, truncation, LAHC, and also other methods is implemented as a block consuming a large computational time in MOEA and MOPSO methods. In Sections 2.2.2 and 5.5, this has been shown as the *Clustering* block. However, another possible way is to omit this block completely and use the concept of ϵ-dominance in the block *Calculate Non-dominated Solutions*. This is shown in Figure 6.4 and is explained in the next section.

6.2.3 ϵ-dominance

From the definition of ϵ-dominance, we know that the size of the ϵ-approximate Pareto-front has an upper bound. Hence, if we use the concept of ϵ-domination instead of domination when updating the archive, the size of the archive is restricted to an upper bound. It means instead of comparing the particles using the standard domination criterion, we compare them using the ϵ-dominance criterion. Therefore, the size of the archive will have an upper bound of $O[(\frac{logK}{log(1+\epsilon)})^{m-1}]$ non-dominated solutions, where K is the upper bound of the objective values. It is obvious that

the size of the archive depends on the ϵ-value. Hence by using this ϵ-dominance, we can keep the size of the archive limited and reduce the computational time. The algorithm using ϵ-dominance is outlined in Algorithm 16. In this algorithm, it is checked if $\vec{a}$ ϵ-dominates $\vec{b}$. Indeed, the domination criterion is also used here.

Algorithm 16 : ϵ-domination

Input: $\vec{a}$, $\vec{b}$
Output: ϵ-non-dominated solution(s)
  if $\vec{a} \prec \vec{b}$ **then**
    return $\vec{a}$
  else if $\vec{b} \prec \vec{a}$ **then**
    return $\vec{b}$
  else if $\vec{a} \prec_\epsilon \vec{b}$ **then**
    return $\vec{a}$
  else
    return $\vec{a}$ and $\vec{b}$
  end if

The reason is shown within an example in Figure 6.5. In this figure, the simple domination criterion is used to check the dominated solutions in the *dashed areas*. The solution $\vec{a}$ dominates $\vec{b}1$ and $\vec{b}2$ dominates $\vec{a}$. Let us not consider $\vec{b}2$ anymore. We can say that $\vec{a}$ ϵ-dominates $\vec{b}3$ and the solution $\vec{b}4$ is indifferent to $\vec{a}$, therefore it must remain in the archive. We have to remark here that if we do not use the first domination criterion, $\vec{a}$ ϵ-dominates $\vec{b}2$ and $\vec{b}2$ will be discarded. Applying the ϵ-

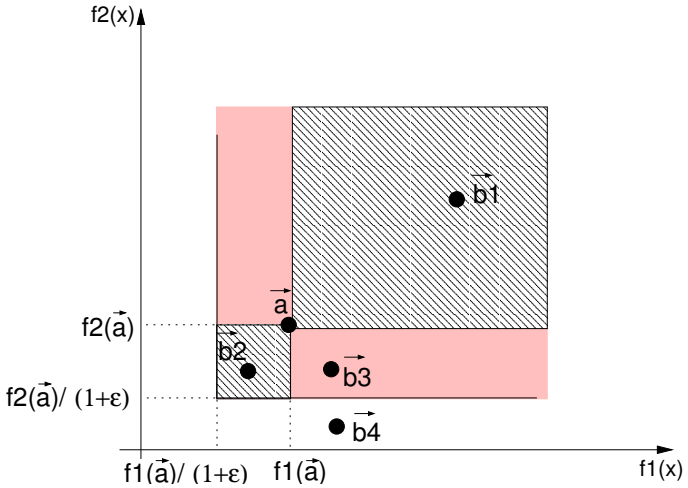


Figure 6.5: ϵ-dominance in MO

dominance in MOPSO techniques also has influence on the convergence and diversity of the solutions. This will be discussed throughout the following experiments.

6.3 Experiments

In the following experiments, two MOPSO methods are compared. For each the archive size is bounded. Both of these methods use the Sigma method (see Section 5.2) for finding the best local guides.

- ϵMOPSO: MOPSO using the ϵ-dominance
- CMOPSO: MOPSO using the clustering technique [Zit99][1]

Here, an initial archive (see Section 5.5.1) is also used for each test function. The initial archives are the results of a *short* MOPSO using the Sigma method. The *short* MOPSOs have a larger population size than the usual MOPSO and are run for a few generations. In this section, we study the influence of ϵ-dominance on the computational time, convergence, and diversity of solutions and compare the convergence and diversity of solutions.
The experiments are performed on 2- and 3-objective test functions from Table 2.1. The ϵMOPSO is run for different ϵ values. Then the maximum archive size in CMOPSO is set to the archive size obtained by the ϵMOPSO. The results are compared in terms of computational time, convergence (by using the C metric), and diversity of solutions (by using the Sigma diversity metric, Section 5.6).

6.3.1 Results

The 2- and 3-objective test functions ZDT1, ZDT3, GSP3, and DLZT are selected from Table 2.1. The Sigma method is run with the following parameters:

- Inertia weight: 0.4,
- Turbulence factor: 0.01,
- Population size: 120,
- Number of generations: 300,
- Archive size: 50.

[1] It is recorded in [KCF03, ZLT02] that the clustering method requires less computational time than the truncation and LAHC methods. Therefore, here the clustering method is selected for comparisons.

The initial archives are obtained by running a MOPSO with population size 200, 100 generations for both of the 2-objective test functions, with population size 500, and 10 generations for the 3-objective test functions GSP3 and DLZT.
Table 6.1 shows the results obtained from the test functions. In this table, A and B refer to the results of ϵMOPSO and CMOPSO, respectively. Size denotes the archive size, t_A and t_B are the required CPU-times for running each MOPSO on a 500 MHz SUN Ultra-SPARC-IIe workstation, N_{AB} refers to the number of solutions in B that are weakly dominated by A, D is the Sigma diversity metric value in percent, and $\tilde{\sigma}$ indicates the median Sigma value for comparing the diversity of the solutions. All the values recorded in Table 6.1 are average values from five different runs with different initial populations.

6.3.2 Influence on Computational Time

Figure 6.6 shows the computational times of the two methods (from Table 6.1) graphically, where size is the archive size and the computational time is shown in logarithmic milliseconds values. For all the test functions, the computational time increases for the increasing values of the archive size. We have to note that when the limit of the archive size is larger than the number of non-dominated solutions, clustering is not applied to the archive. This can be observed particularly for both of the 2-objective test functions. The computational time of the CMOPSO decreases for large archive sizes. In both the 2- and 3-objective test functions the computational time of the ϵMOPSO is much less than the CMOPSO; the CMOPSO takes in some cases more than 100 times the ϵMOPSO to find the same number of solutions. In Table 6.1, $*$ refers to the number of solutions obtained by CMOPSO. The ϵMOPSO method finds 941 solutions for the test function ZDT1, when $\epsilon = 0.0001$. But if we apply the method using the clustering technique (CMOPSO), we see that we never reach the number of 941 as the archive size in order to apply clustering on it, therefore it will take less time than the ϵMOPSO method.

6.3.3 Influence on Convergence

In Table 6.1, the factor N_{AB} shows the number of solutions in set B that are weakly dominated by the solutions in set A, i.e.:

$$N_{AB} := |\{b \in B | \exists a \in A : a \preceq b\}| \tag{6.3}$$

By comparing N_{AB} and N_{BA}, where A is the ϵMOPSO and B is the CMOPSO method, we can conclude that for the same archive sizes the results of ϵMOPSO dominate more solutions of the results of CMOPSO. This also depends on the archive

Table 6.1: A: ϵMOPSO, B: CMOPSO (times in milliseconds)

test	ϵ	size	t_A	t_B	N_{AB}	N_{BA}	D_A	D_B
ZDT1	0.1	16	3041	33429	4	0	87	87
	0.05	26	3312	37006	4	0	80	80
	0.025	48	3305	144151	1	1	93	91
	0.01	109	3754	204412	13	12	77	74
	0.005	204	4401	318432	80	13	93	87
	0.001	517	6331	23294	398	13	93	93
	0.0001	941	8083	6096*	828	7	97	93
	0	616	6096	-	-	-	-	-
ZDT3	0.025	20	3127	86329	0	9	50	45
	0.01	40	3229	11807	3	5	30	28
	0.0075	52	3300	12270	0	11	27	30
	0.005	71	3358	15102	6	14	34	48
	0.0025	123	3635	11509	80	2	43	53
	0.001	170	3909	55498	33	54	51	60
	0.0005	220	4353	78289	32	65	59	58
	0.00025	249	4416	40306	179	7	61	61
	0.0001	507	5868	41203	460	2	55	51
	0	730	7560	-	-	-	-	-
GSP3	0.1	68	2968	164954	16	0	77	91
	0.07	113	3368	254867	17	0	83	83
	0.06	130	3740	321244	12	0	86	93
	0.05	176	4188	555989	19	2	91	93
	0.04	219	5191	510482	25	1	98	98
	0.03	351	6979	1161747	34	8	70	83
	0.02	660	12126	3277912	73	7	92	93
	0.015	956	16694	6154627	135	11	98	97
	0	10692	172782	-	-	-	-	-
DLZT	0.1	30	2839	165275	3	0	32	28
	0.05	76	3275	298039	1	0	23	29
	0.04	84	3505	353349	1	1	23	28
	0.03	133	4194	552847	1	0	22	25
	0.025	157	4735	715612	1	1	21	24
	0.02	216	5788	1001681	7	1	24	27
	0.015	331	8103	1907411	2	0	18	19
	0.01	610	13574	5152398	10	1	19	21
	0	21351	373641	-	-	-	-	-

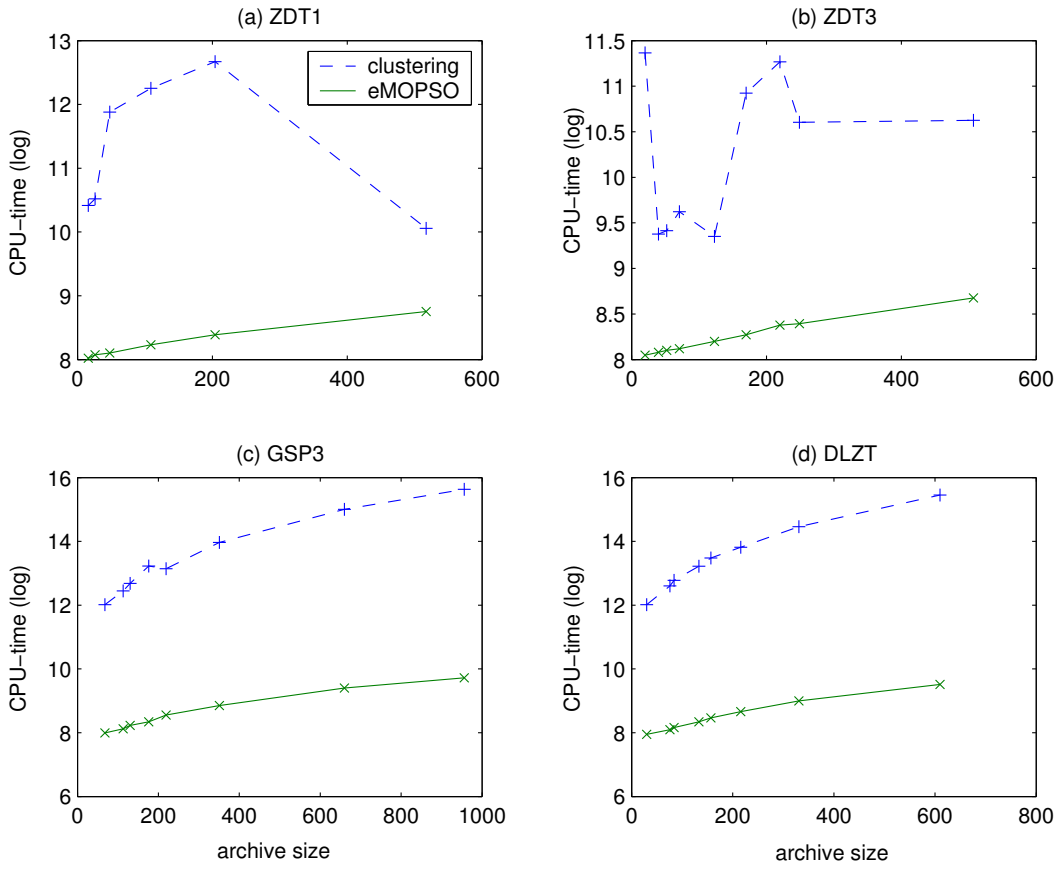


Figure 6.6: Comparison of computational time (CPU-time) of the CMOPSO and ϵMOPSO methods in milliseconds

size and the number of objectives. For test functions ZDT1, ZDT3, and GSP3, the values of N_{AB} are much higher than N_{BA}, from which we can conclude a better convergence. However, for the 3-objective test function DLZT, they are comparable.

6.3.4 Influence on Diversity

The diversity of solutions are measured by the Sigma diversity metric D and the median Sigma value $\tilde{\sigma}$. In Table 6.1, the D values are shown in percentages. Here, we study the results of 2- and 3-objective test functions separately:

2-objective Test Functions

Comparing the diversity of the results of the ϵMOPSO method with the results of the CMOPSO method by using D values, we conclude that for larger archive sizes the results of the ϵMOPSO have larger D values, which means a better diversity. In some

cases the results of the CMOPSO obtains higher D values than ϵMOPSO. However, the diversity of solutions is comparable. Figure 6.7 show the results graphically. The

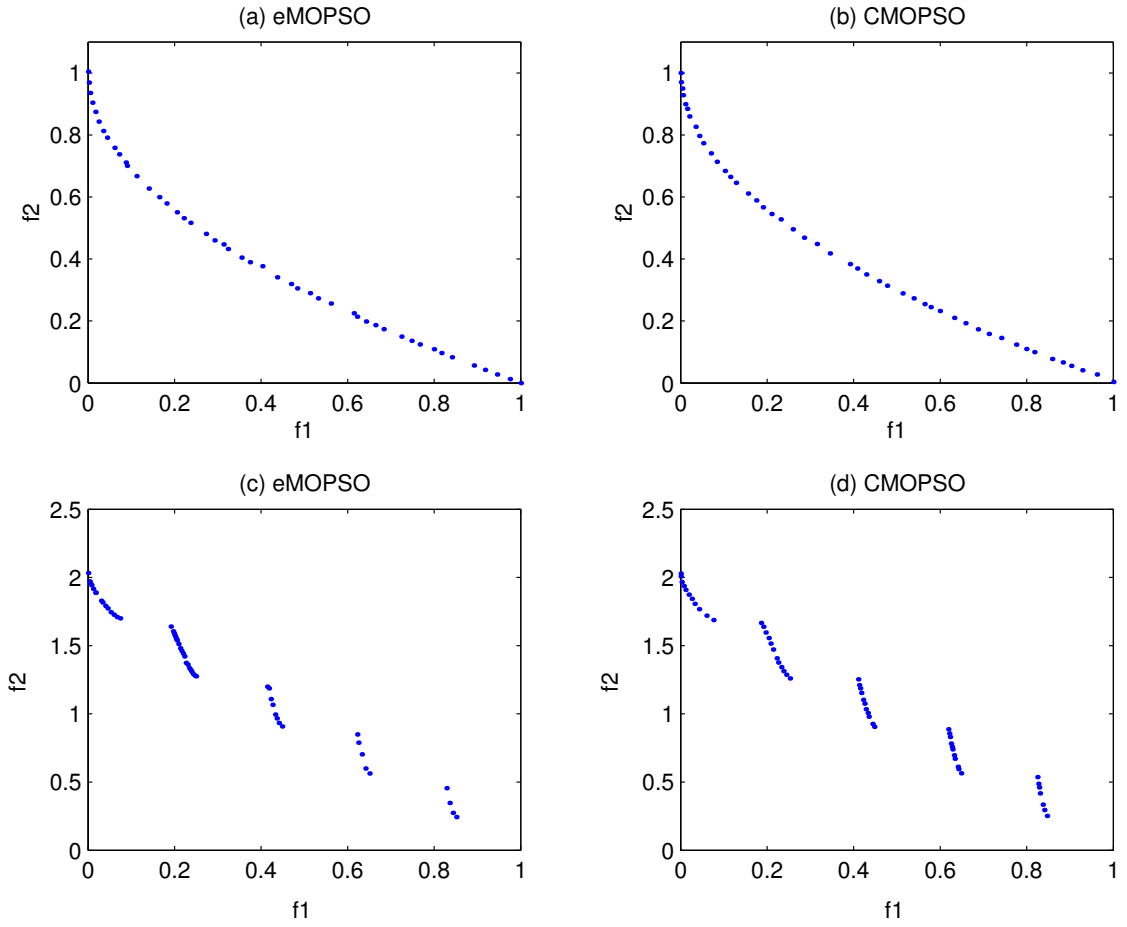


Figure 6.7: (a),(b) Results of ZDT1 ($\epsilon = 0.025$, $|A| = 48$); (c),(d) Results of ZDT3 ($\epsilon = 0.0075$, $|A| = 52$)

archive sizes are 48 and 52 for the ZDT1 and the ZDT3 test functions, respectively. It can be observed that the ϵMOPSO cannot find solutions with comparable diversity to CMOPSO. This is more obvious for the test function ZDT3.
Quantitatively, Table 6.2 shows the median Sigma values of the solutions. The median Sigma value shows how the solutions are symmetrically distributed along the non-dominated front. These results show that, although the D values shown in Table 6.1 are high, by increasing the number of solutions in the archive, the solutions tend to move toward the left upper part of the non-dominated front. Indeed, this is valid for both the ϵMOPSO and CMOPSO methods. However, CMOPSO obtains solutions with very good symmetry on the approximated Pareto-optimal front. For

Table 6.2: Median Sigma values applied on the results of the 2-objective test function (A: ϵMOPSO, B: CMOPSO)

ZDT1	ϵ	0.1	0.05	0.025	0.01	0.005	0.001	0.0001	0.0
	$\tilde{\sigma}_A$	-0.42	+0.28	-0.08	-0.02	+0.03	+0.61	+0.81	+0.71
	$\tilde{\sigma}_B$	+0.052	+0.12	-0.084	-0.14	-0.05	+0.42	+0.71	–
ZDT3	ϵ	0.025	0.01	0.005	0.0025	0.001	0.00025	0.0001	0.0
	$\tilde{\sigma}_A$	-0.83	-0.86	-0.81	-0.76	-0.36	-0.24	-0.039	+0.67
	$\tilde{\sigma}_B$	-0.37	-0.34	-0.29	-0.28	-0.29	-0.25	+0.47	–

example, consider the results of the ZDT1 test function in Table 6.2. In most of the cases, the results of the CMOPSO have lower absolute values of the median Sigma than the results of ϵMOPSO. It means that the solutions of CMOPSO are better distributed than the ϵMOPSO.

3-objective Test Functions

Considering the 3-objective test functions, the CMOPSO method obtains a better diversity of solutions than the ϵMOPSO method. One of the reasons may be the shape of the approximated Pareto-front. Figures 6.8 and 6.9 show the results of the GSP3 and DLZT test functions and also the θ-ϕ axis of the solutions (spherical coordinates), for having a better observation on the diversity of solutions. We can observe that the ϵMOPSO method cannot obtain some solutions, therefore the diversity of solutions, especially for test function GSP3, is not as good as the CMOPSO method. Indeed, this is due to the convex shape of the GSP3 test function.
Table 6.3 shows the median Sigma vectors of the solutions for different values of ϵ. Together with the D values in Table 6.1, the position of the solutions on the front can be studied in more detail. The absolute values of each element of the median Sigma vector must be zero for both the GSP3 and DLZT test functions. Therefore, it can also be quantitatively concluded that the solutions of the CMOPSO method are better symmetrically well-distributed than the results of ϵMOPSO. However, the ϵMOPSO also finds solutions with (acceptable) good diversity.

6.4 Conclusion

In this chapter, methods for bounding the archive size are studied and compared. Restricting the archive is performed by using clustering or similar techniques developed in the past. However, clustering takes a lot of computational time, since it

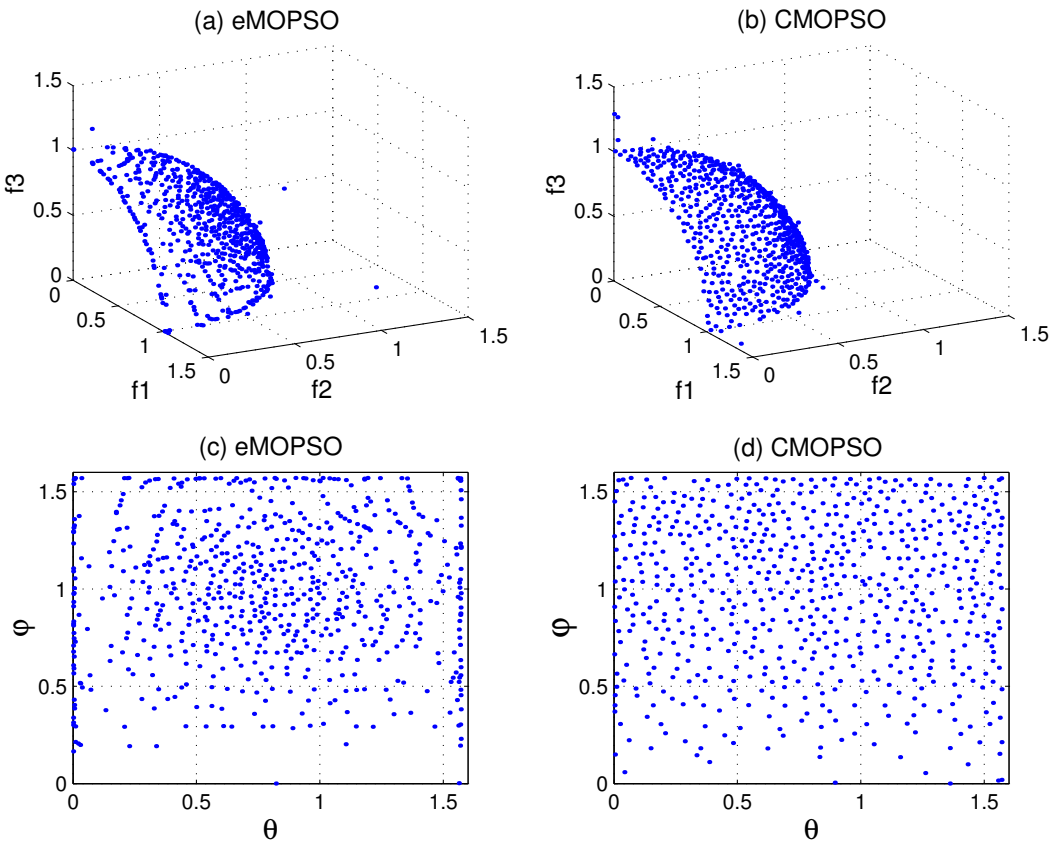


Figure 6.8: GSP3 test function ($\epsilon = 0.02$). (a),(b) objective space, and (c),(d) $\theta - \phi$ axis of the spherical coordinate

must keep a good diversity of solutions. Here, a method called ϵ-domination is studied as an alternative solution for guaranteeing bounded archive size. By extending the domination test to ϵ-dominance according to [PY00, LTDZ02], we can guarantee that only a bounded number of solutions according to Equation (6.2) may be non-dominated. Therefore, the archive is bounded. Indeed, the size of the archive has an upper bound. This method is called ϵMOPSO and compared to the MOPSO using the clustering technique called CMOPSO. The attractive property of ϵMOPSO is that it maintains a good diversity of solutions, without increasing the computational time. The computational time, diversity, and convergence of solutions are tested on different test functions:

- The computational time of ϵMOPSO is much less than of CMOPSO. In some cases 10 to 100 time less than CMOPSO.
- The obtained solutions from ϵMOPSO have comparable convergence and diver-

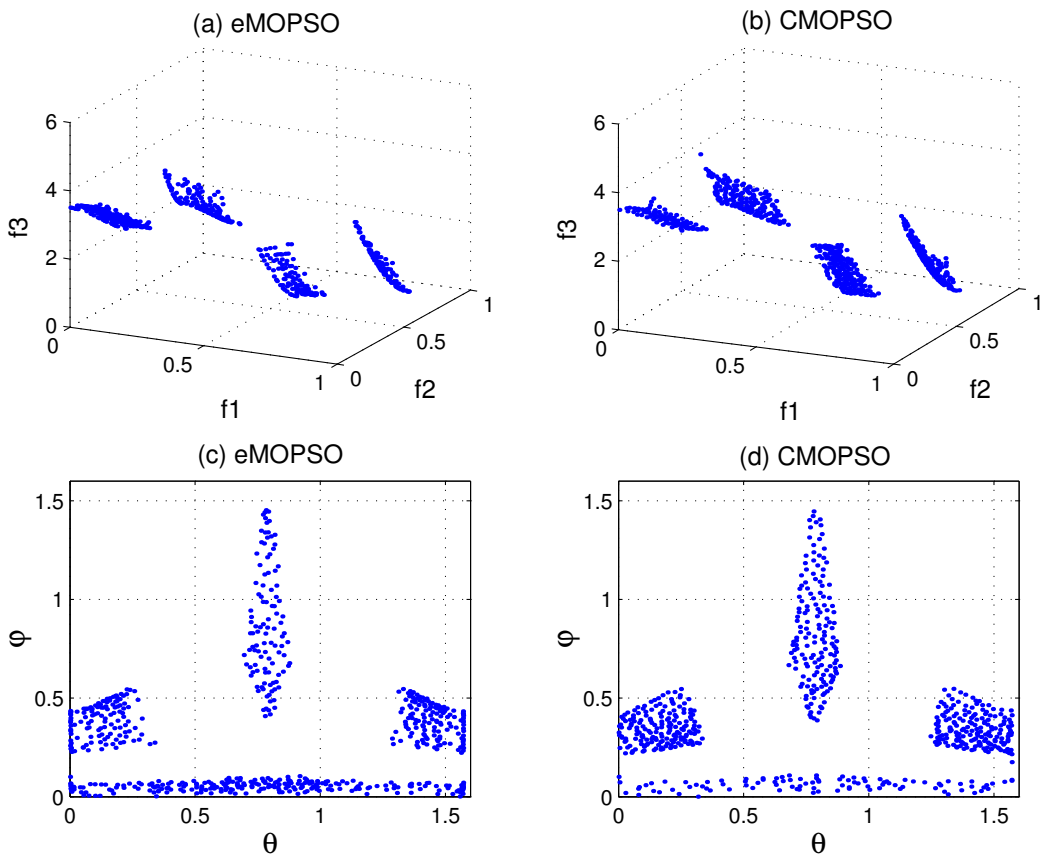


Figure 6.9: DLZT test function ($\epsilon = 0.01$). (a),(b) objective space, and (c),(d) $\theta - \phi$ axis of the spherical coordinate

sity to the solutions of CMOPSO and in some cases have better convergence, especially for 2-objective test functions.

- The diversities of the solutions are compared with the Sigma diversity metric and median Sigma value. According to these metrics, the diversity of the solutions obtained by ϵMOPSO is getting worse than the CMOPSO for higher number of objectives. But they keep an acceptable symmetry on the front.

Indeed, there is a trade-off between computational time and diversity of solutions. In the cases that the computational time is very important, the ϵMOPSO is suggested. However, we have to consider that the results are just for the recorded number of generations. Also, ϵMOPSO is suggested to be used when several solutions with high convergence are required, for example in the covering of the Pareto-optimal front using MOPSO (in Section 5.8).

Table 6.3: Median Sigma values applied on the results of the 3-objective test function (A: ϵMOPSO, B: CMOPSO)

GSP3						
ϵ	0.1	0.06	0.04	0.02	0.015	0.0
$\tilde{\sigma}_A$	$\begin{pmatrix} +0.11 \\ +0.00 \\ -0.11 \end{pmatrix}$	$\begin{pmatrix} +0.00 \\ +0.00 \\ +0.01 \end{pmatrix}$	$\begin{pmatrix} -0.02 \\ +0.03 \\ +0.00 \end{pmatrix}$	$\begin{pmatrix} +0.00 \\ -0.00 \\ +0.01 \end{pmatrix}$	$\begin{pmatrix} -0.01 \\ -0.02 \\ +0.02 \end{pmatrix}$	$\begin{pmatrix} -0.00 \\ -0.18 \\ +0.16 \end{pmatrix}$
$\tilde{\sigma}_B$	$\begin{pmatrix} -0.05 \\ +0.02 \\ +0.01 \end{pmatrix}$	$\begin{pmatrix} -0.00 \\ +0.00 \\ -0.00 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.00 \\ -0.01 \end{pmatrix}$	$\begin{pmatrix} -0.01 \\ -0.01 \\ +0.01 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.00 \\ +0.01 \end{pmatrix}$	–
DLZT						
ϵ	0.05	0.03	0.02	0.015	0.01	0.0
$\tilde{\sigma}_A$	$\begin{pmatrix} -0.00 \\ -0.06 \\ +0.06 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.06 \\ +0.06 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.06 \\ +0.06 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.06 \\ +0.06 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ -0.05 \\ +0.05 \end{pmatrix}$	$\begin{pmatrix} -0.00 \\ -0.05 \\ +0.05 \end{pmatrix}$
$\tilde{\sigma}_B$	$\begin{pmatrix} +0.00 \\ +0.02 \\ -0.01 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ +0.02 \\ -0.00 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ +0.01 \\ -0.02 \end{pmatrix}$	$\begin{pmatrix} 0.0 \\ +0.02 \\ -0.02 \end{pmatrix}$	$\begin{pmatrix} -0.00 \\ +0.01 \\ -0.01 \end{pmatrix}$	–

Chapter 7

Applications

In this chapter, two real-world multi-objective problems are studied. These problems are optimized using the methods investigated in the thesis. The first application is an antenna design problem which is also studied in Chapter 4. The second application, which is the main application of this study, is a real-world problem in computational chemistry. The solutions of these problems are compared with the solutions of the existing MOEA method.

7.1 Application in Antenna Design

It is a basic problem in antenna design to construct the shape or choose the *feeding* of the antenna to optimize the performance of the antenna. There are different types of problems in designing the antenna. Here, the geometry of the antenna is kept constant and the feeding of the antenna is varying. In other words, *the optimization problem is to direct the power of the antenna in a specific direction.* The bicriterial problem for the fixed geometry of the antenna and the wave propagation of the fields generated by currents on the antenna are studied by [JJK97]. The antenna is considered as a hollow infinite cylinder in the z direction[1] with constant cross section. The optimization problem is formulated as follows:

1. Maximize the *radiation efficiency* in a particular direction $\hat{\theta}$. The radiation efficiency is defined as the ratio of the power radiated in the particular direction to the total power fed to the antenna (more details are explained in [JJK97]).

2. Minimize the power radiated in other directions.

Figure 7.1 shows an example of radiation characteristics of points on the non-dominated front. This figure is shown just as an example for clarifying the multi-

[1] We consider the 3 dimensional space of x, y, and z directions

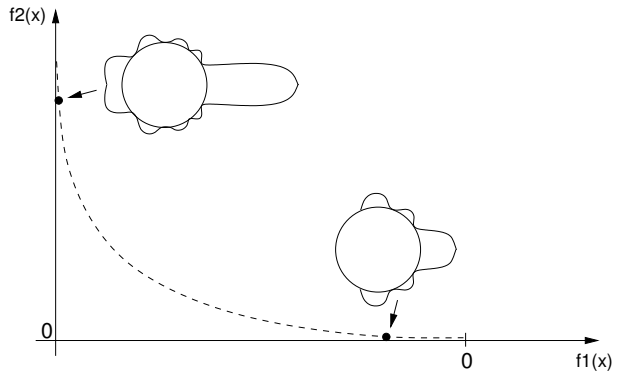


Figure 7.1: Radiation characteristics of the points on the approximated front

objective problem. The figure consists of a non-dominated front of the optimization problem, the horizontal intersection of the antenna, and the wave radiation in different directions. As shown in the figure, the first objective maximizes the radiation in one direction and the second objective minimizes the radiation in other directions. This antenna design problem is studied in Chapter 4. There, covering the Pareto-optimal front and controllable exploration of the search space is studied by HMOEA (hybrid MOEA). The test functions and parameters are also explained in Section 4.3. The number of parameters is considered to be 12. Here, this example is studied again and the MOPSO covering method is applied to it.

7.1.1 Experiments

In the experiments, the MOPSO method is applied and compared with the presented results in Chapter 4. Also, the approximated front is covered by the covering MOPSO technique (see Chapter 5). The selected parameters are set as follows:

- Inertia weight: initial run: 0.75, covering: 0.5,
- Turbulence factor: 0.7,
- Population size: 500,
- Number of generations: initial run: 500, covering: 1000,
- Archive size: initial run: 50, covering: unrestricted.

For the initial run, the inertia weight is selected 0.75, which is larger than for the covering process. The reason is as follows: The high value of the inertia weight leads the solutions to high convergence, and together with clustering (restricted archive) to a good diversity of solutions. But in the covering process, we need to explore the area around each non-dominated solution. Therefore, the inertia value should be

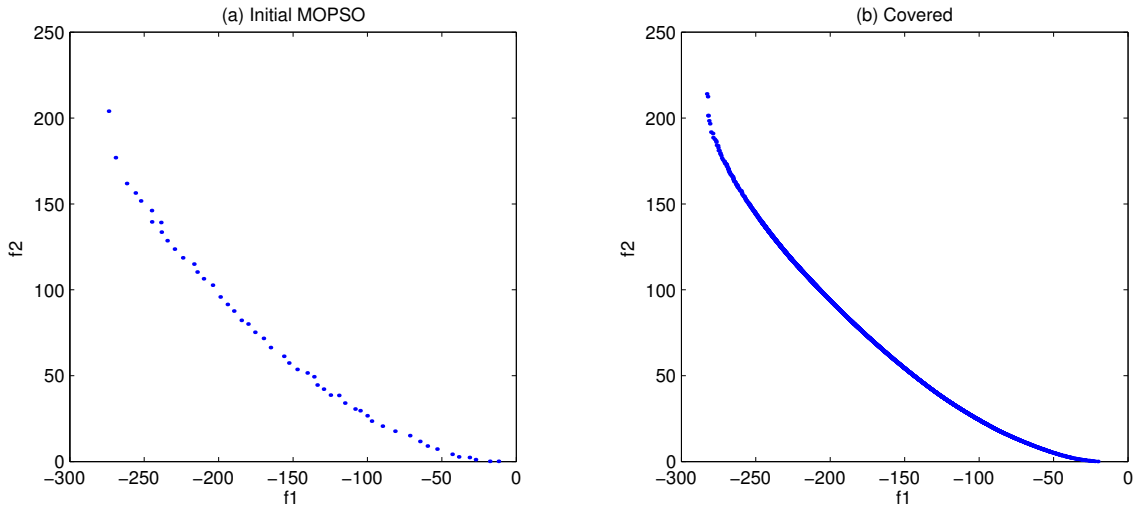


Figure 7.2: Antenna design problem, results of MOPSO (a) archive size: 50 (b) covered approximated Pareto-optimal front

decreased and the particles may have the chance to search their neighborhood areas. Figure 7.2 shows the results of the Sigma method and the covered approximated Pareto-optimal front.

The experiments are also compared with the same results of the MOEA (SPEA2) in Section 4.3. The selected parameters for the MOEA method is the same as in the second run in Section 4.3. Figure 7.3 (b) shows the results in a selected part

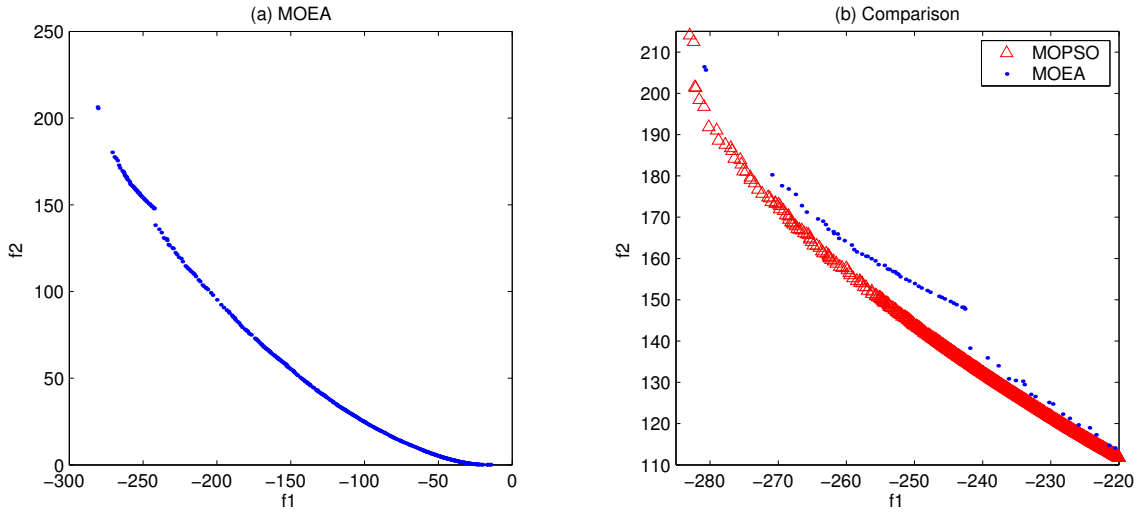


Figure 7.3: Antenna design problem (a) Result of MOEA (b) Comparison of MOEA and MOPSO in the selected space

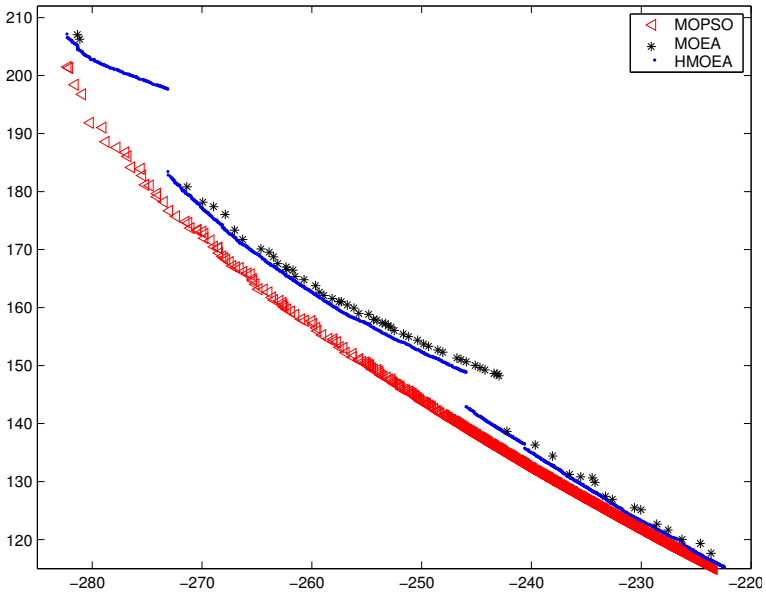


Figure 7.4: Comparison of HMOEA, MOEA and MOPSO for the selected part of the objective space of the antenna design problem

of the objective space for better observable comparison. The computational times for the initial run and the covering are 70.52 and 385 seconds, respectively. The quantitative comparison between the solutions is done by applying the C metric:

- $C(\text{MOPSO},\ \text{MOEA}) = 0.97$
- $C(\text{MOPSO},\ \text{MOEA}) = 0.00$

It clearly shows that the MOPSO obtains solutions with better convergence than the MOEA.
For more detailed comparison, Figure 7.4 shows the results of the covered approximated Pareto-optimal front using the HMOEA in Chapter 4, MOEA and covered MOPSO. The total running time for HMOEA is 20 minutes for the MOEA result and another 15 minutes for the recovering process. The computational time of the method is much higher than the covering MOPSO technique. Also, it can be observed that for the selected part of the objective space, the MOPSO method can find solutions with better convergence than the other methods. The minimum, maximum, and average distances between the covered solutions by MOPSO are $D_{min} = 1.6808e-04$, $D_{max} = 4.9609$, and $D_{av} = 0.0816$, respectively.
The distances between the solutions on the approximated front can be fixed to a lower bound, in the case of using ϵMOPSO. However, the MOPSO covering process can be terminated by assigning a threshold to the maximum distance between the neighbor solutions.

7.2 Application in Computational Chemistry

In this section, a real world application in *parameterization of molecular force fields* with three objectives is studied. In the following, a brief introduction is dedicated to this application. For more details please refer to [Jen99, PC03].

7.2.1 Parameterization of Molecular Force Fields

The notion of describing molecules by letters symbolizing atoms and lines symbolizing bonds can be declared as the most common way among the chemists and also others familiar with basic concepts of chemistry. An obvious choice for describing molecules is hence inspired by this *graphical* view. In so-called ball-and-springs models, atoms are represented by more or less elastic balls interconnected by springs representing the deformability of the structure. Also, in terms of the molecular force fields, these models can be used to explore, explain and predict a large variety of molecular properties. Common properties are for instance, the most favorable geometrical arrangement of atoms that correspond to a stable molecule (*conformations*), the relative energies of different arrangements, properties of these, or the models of the dynamical behavior of molecular structures such as the probability of transition between different conformations. Molecular force fields can be used for modeling a variety of chemical systems, ranging from the metal compounds used in industrial catalysts to biomolecular systems.
One crucial characteristic of molecular force field methods is the imperative of parameterizing the underlying functions, which is a tedious task since force fields involve a large number of parameters and a number of objectives that describe specific properties of the system.
One important aim for the force field parameterization is to obtain the best description of the reference data. Genetic algorithms are well suited for finding solutions close to the global best solution for this optimization problem.
When using conventional optimization techniques, either iterative techniques have to be employed – where each of the objectives is optimized one by one until a self-consistent state has been reached – or suitable weighting functions have to be introduced for reducing the dimensionality of the optimization problem [WK01]. Also, different approaches have been made using genetic algorithms. Busold and Strassner [Bus01, SBR01] describe an approach for parameterizing metalloorganic compounds in the framework of the Molecular Mechanics version 3 (MM3) force field [AYL89] using genetic algorithms. In similar (earlier) publications of Huttner et al., natural parameters for the MM2 force field for molybdenum compounds were parameterized using the *rmsd* values to a large number of reference structures as

the objective function [HH99, HBH+98]. The authors optimize the parameters that can directly be deduced from the objective function. Wang and Kollmann [WK01] describe the parameterization using genetic algorithms and a combination of different objectives with weighting functions. This approach however, has the intrinsic problem outlined above.
So far, there is no research on the application of MOEA or MOPSO for force field parameterization. Therefore, the application of MOEA on this problem is studied here.

7.2.2 Molecular Force Fields

The fundamental characteristic in describing a molecule or properties of a molecule is the energy of the system as a function of the nuclear coordinates. In force field methods this is realized by a parametric function that connects coordinates with energies. The work presented here is focused on the parameterization of a class II force field, namely the CHARMM [BBO+83] force field.
The molecular force field is of the functional form denoted in Equations (7.1)-(7.3). In these equations interactions in molecules are classified into interactions mediated by bonds, termed intramolecular Equation (7.2), and interactions between atoms separated by three or more bonds, termed intermolecular Equation (7.3). The total energy term describing a molecule is the sum of inter- and intramolecular energy contributions:

$$E = E_{intra} + E_{inter} \tag{7.1}$$

A natural choice of coordinates for the intramolecular interactions is one that is directly derived from the topology of the molecule using bond lengths, angles, and dihedral angles (see Figure 7.5). The intramolecular potential describes the variation of energy connected to deviations from given geometrical parameters (natural parameters).

$$\begin{aligned} E_{intra} &= \sum_{bonds} \frac{1}{2} k_b (b - b_0)^2 + \sum_{angles} \frac{1}{2} k_\theta (\theta - \theta_0)^2 + \\ &\quad + \sum_{torsions} \sum_{n} k_{n\phi} (\, 1 + cos(n\phi - \delta_n)) \end{aligned} \tag{7.2}$$

$$E_{inter} = \sum_{\substack{nonbonded, \\ i<j}} \frac{q_i q_j}{r_{ij}} + \sum_{\substack{nonbonded, \\ i<j}} \epsilon_{ij} \left[\left(\frac{R_{min,ij}}{r_{ij}} \right)^6 + \left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} \right] \tag{7.3}$$

The first term in (Equation 7.2) describes the necessary energy for lengthening and shortening bond lengths b, relative to a natural bond length b_0. The potential is harmonic with a force constant k_b. In the same fashion, deviations of bond angles θ from a natural bond angle θ_0 are described. The description of the dihedral coordinates ϕ differs as the function is required to satisfy periodicity. Force field parameterization includes the determination of suitable values for the force constants of bonds k_b, angles k_θ, and dihedral angles $k_{n\phi}$, as well as their associated natural values b_0, θ_0, and the phase for the dihedral angles δ_n. For the description of non-bonded interactions the partial charges q for the electrostatic energy and the Lenard-Jones Parameters ϵ_{ij} and $R_{min,ij}$ for the description of van der Waals interactions must be determined (see [Jen99, PC03] for more details).
To obtain a reasonable description, all of these parameters must be adapted for different *atom types*, i.e., different chemical elements and bonding situations. Hence, the force field parameters are specific for each term summed over. For instance, a carbon-carbon bond in a benzene ring and a carbon-carbon bond in an alcohol represent different bonding Situations, and different values for both k_b and b_0 are required. Simultaneously, force field parameters must be as transferable as possible, meaning that the parameters must be applicable to a different molecule showing a similar bonding situation. In many cases parameters can be adapted directly to other molecules (parameterization by analogy), while in some cases the description can be significantly improved by re-parameterization [Mac03].

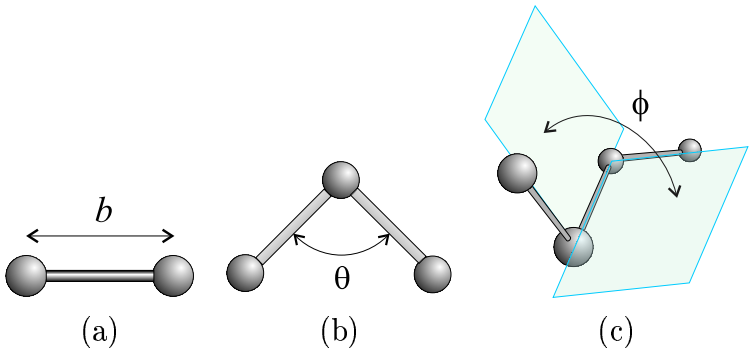


Figure 7.5: Relevant intramolecular geometry measures used in molecular force fields: a) bond lengths b) bond angles c) dihedral angles.

7.2.3 Description of Objective Functions

For the parameterization of force field terms, both experimental and *ab initio*[2] data can be employed. Here, the reproduction of *ab initio* molecular geometries, molecular vibrations, and rotational barriers are studied. As components of the decision vector, the force field parameters k_b, b_0, k_θ, θ_0, and $k_{n\phi}$ are used. The number of components of the decision vector $\vec{x}$ is defined by the particular parameterization problem. As explained above, for each specific chemical bonding situation different parameters are required. In this study we consider three objective functions as follows.

Objective 1: Reproduction of Molecular Geometries

The first objective is to take the maximum component of the energy gradient in Cartesian coordinates for the reproduction of molecular geometries Equation (7.4)[3]. Indeed, the test parameters can be used to exert the forces on the atoms of the reference structure. If the potential generated by the force field has a minimum at the same place as the reference structure, the forces, i.e., the gradients, should be zero [DI89]. This argument is followed and it is suggested to take the maximum component of the energy gradient in Cartesian coordinates as the objective function as follows:

$$f_1 := \max_{1 \leq i \leq N} |\nabla_i E| \tag{7.4}$$

Here, N specifies the number of atoms in the molecule and ∇_i is the gradient vector with respect to the position of the i-th atom.

Objective 2: Reproduction of Molecular Vibrations

The interpretation of vibrational spectra suffers from the identification problem. It describes the task of assigning each calculated vibration to the corresponding reference vibration, i.e., an experimentally observed one or one calculated using ab-initio methods. Due to this problem a direct fitting, which only compares the vibrations directly sorted by frequency, can lead to unphysical results and may prevent correct optimization.
To circumvent this problem a better choice is to compare the force matrices in *internal coordinates*. Molecular vibrations in harmonic approximation, i.e., vibrational frequencies, and normal coordinates are obtained using the second derivatives of the energy. For more details about this objective please refer to [MHK+04].

[2] *ab initio* refers to quantum mechanical calculations where no a priori information is required

[3] The units for objective functions f_1 and f_3 are kcal (mol Å)$^{-1}$ and (kcal mol^{-1})^2, respectively.

The elements $F_{ij}^{(c)}$ of the Cartesian force matrix $\mathbf{F}^{(c)}$ in the direction of the Cartesian displacement coordinates $q_i^{(c)}$ are calculated according to Equation (7.5):

$$F_{ij}^{(c)} = \frac{\partial^2 E}{\partial q_i^{(c)} \partial q_j^{(c)}} \qquad 1 \leq i, j \leq 3N \tag{7.5}$$

In internal coordinates the deformations of a molecule are not described using displacements in terms of an external Cartesian coordinate system, but regarding changes of internal coordinates of the molecule, e.g., bond lengths, bond angles and dihedral angles. The definition of the internal coordinates used in this work is based on Pulay et al. in [PFPB79].
These internal coordinates $\mathbf{q}^{(i)}$ can be described as a function of the displacement in Cartesian coordinates $\mathbf{q}^{(c)}$:

$$\mathbf{q}^{(i)} = \mathbf{q}^{(i)}(\mathbf{q}^{(c)}) \tag{7.6}$$

This function is not linear in general. But under the assumption of small displacements the transformation from Cartesian to internal coordinates can be linearized.

$$\mathbf{q}^{(i)} = \mathbf{B}\mathbf{q}^{(c)} \tag{7.7}$$

The construction of the matrix $\mathbf{B}$ was first described by Wilson et al. [EBDC55]. In this work the $(3N-6,\ 3N)$-dimensional $\mathbf{B}$ matrix respective to its pseudo-inverse $\mathbf{B}^{\dagger}$ is used to transform the force matrix into internal coordinates [Reb98].

$$\mathbf{F}^{(i)} = \mathbf{B}^{T\dagger}\mathbf{F}^{(c)}\mathbf{B}^{\dagger} \tag{7.8}$$

One of the advantages of this approach is that the problem is decoupled from the external coordinate system. Therefore, a more natural description of the system can be achieved at the cost of choosing a proper set of non-redundant internal coordinates and transforming the calculated matrices into internal coordinates. In order to minimize the identification problem, the relative deviation in Equation (7.9) of the diagonal elements of the force constant matrix in internal coordinates, with respect to a reference matrix, is used as our second objective. Comparing only the diagonal elements is possible, because after the transformation the off-diagonal elements are smaller than the diagonal elements by several orders of magnitude.

$$f_2 := \sum_{j=1}^{3N-6} \left(\frac{F_{jj}^{(i)} - F_{jj}^{(i),ref}}{F_{jj}^{(i),ref}} \right)^2 \tag{7.9}$$

The choice of absolute deviations leads to a good description of vibrations associated with the diagonal elements that are large, i.e., bond stretching, but a weak description of vibrations where the associated diagonal elements are small. Hence, we choose relative deviations.

Objective 3: Reproduction of Energetics for Different Conformations

A force field must be able to reproduce the relative energies for different conformations as well as the barriers between them. Therefore, the third objective function is defined by the sum over quadratic energy deviations for all relevant conformations and the barriers between them [3]:

$$f_3 := \sum_i (E_i - E_i^{ref})^2 \tag{7.10}$$

7.2.4 Experiments

The experiments are carried out using the MOPSO and MOEA methods, namely the Sigma and SPEA2 methods, respectively. The computational details of the physical point of view are selected as in [MHK+04]. The MOEA and MOPSO methods are run with the following parameters:

- Inertia weight: 0.4,
- Turbulence factor: 0.01,
- Mutation probability: 0.01, 0.1,
- Cross-over probability: 0.8,
- Population size: 300,
- Archive size: 200.

For validation, the force field parameters were determined for primary alcohols. While the force field parameters for these alcohols are available in common force fields, they are used for *validation* of the methods described here. They are of simple structure and allow the testing of the new suggested procedure.
Aliphatic alcohols whose parameters were fitted in this study have the general structure depicted in Figure 7.6. In the series k is a nonnegative integer describing the length of the alcohol.

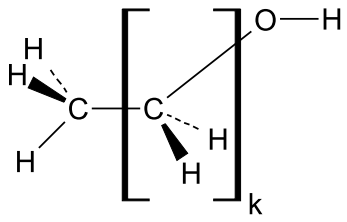


Figure 7.6: General structure of non-branched, primary, aliphatic alcohols. The index k indicates how often the unit contained within the brackets is repeated.

7.2.5 Case Study: Set of Two Alcohols

The case study is aimed to find the force field parameters for a set of two alcohols: methanol $\mathsf{k} = 0$ and ethanol $\mathsf{k} = 1$. The number of parameters and objectives are 36 and 3, respectively. The set of parameters is composed of 6 parameters for k_b and b_0, 9 parameters for k_θ and θ_0, and 6 parameters for $k_{n\phi}$. The objectives are the functions introduced in Equations (7.4), (7.9), and (7.10), where the values for the two molecules are summed up, e.g., $f_1^{\Sigma} = f_1^{methanol} + f_1^{ethanol}$. For the computation of the objective function f_2, 12 and 21 (for methanol and ethanol, respectively) internal coordinates are defined for calculating the force matrices in Equation (7.9) (for more details refer to [MHK+04]).

7.2.6 Comparison of Different Algorithms

The first experiments are carried out using the MOEA method with different probabilities of mutation (p_m), namely 0.01 and 0.1. The optimization process is terminated after 3000 generations. Figures 7.7 (a) and 7.7 (b) show the results of the MOEA with different mutation probabilities. It can be observed that the MOEA with higher p_m finds better solutions in terms of the first objective. However, the quantitative C metric values are as follows:

$$C(MOEA_{0.1}, MOEA_{0.01}) = 8\%$$
$$C(MOEA_{0.01}, MOEA_{0.1}) = 73\%$$

For additional comparisons we also compare the results of the 10 runs of MOPSO after 3000 generations, and the best run is further analyzed (the same as the MOEA results). Figure 7.7 (c) shows the non-dominated set in the objective space. The

values of the quantitative measures (C metric) are as follows:

$$C(MOPSO, MOEA_{0.1}) = 97\%$$
$$C(MOEA_{0.1}, MOPSO) = 0\%$$

$$C(MOPSO, MOEA_{0.01}) = 42\%$$
$$C(MOEA_{0.01}, MOPSO) = 2\%$$

These values show that the solutions of the MOPSO dominate most of the solutions of the MOEA with $p_m = 0.1$ and 42% of the solutions of the MOEA with $p_m = 0.01$. This can also be observed in Figure 7.7. Although, the MOEA with a lower p_m is able to obtain solutions with better convergence than with higher probability, 42 % of its solutions are still dominated by the solutions of the MOPSO. However,

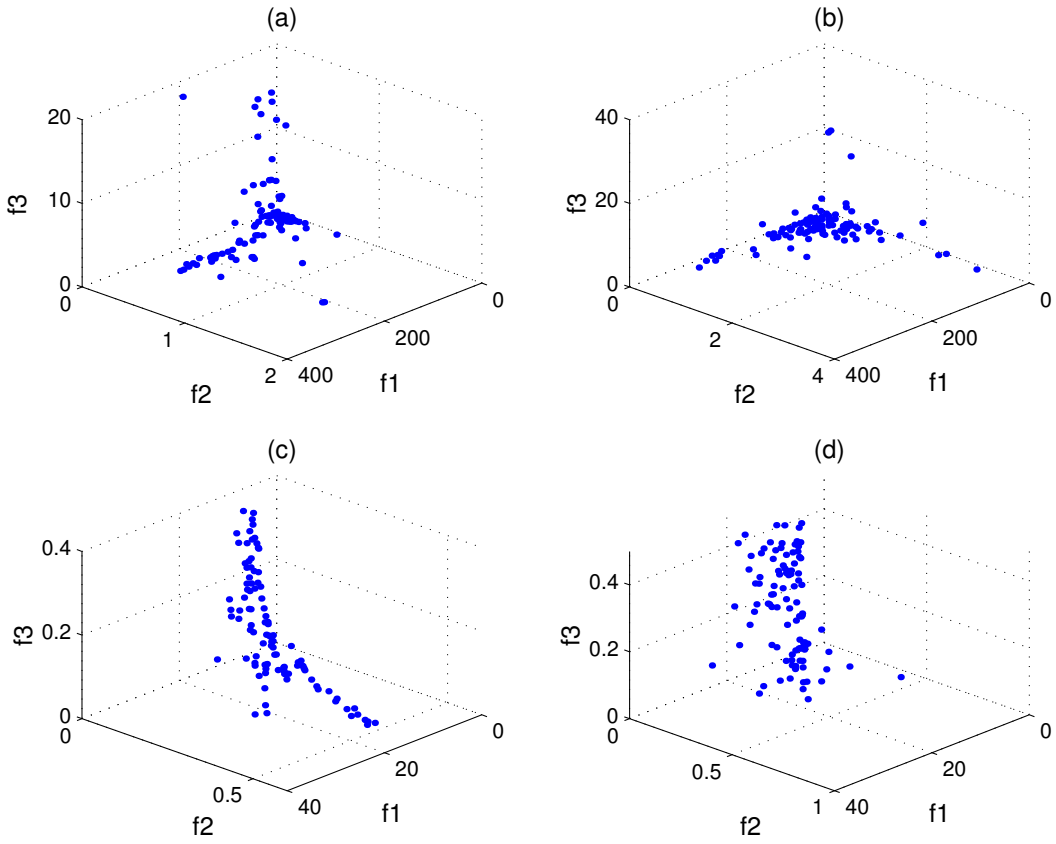


Figure 7.7: The non-dominated front in objective space obtained for the best three runs of (a) MOEA, p_m=0.01, (b) MOEA, p_m=0.1, (c) MOPSO after 3000 generations. (d) the result of MOPSO after 10000 generations

MOPSO is able to achieve solutions which show a higher convergence than those obtained with MOEA for the same number of generations.

Stopping Criteria

As the single evaluations of the objective functions are computationally expensive, the applicability for real-world parametrizations critically depends on the number of generations required. Furthermore, the results of the optimization are, in general, unknown. Hence, we require a reliable criterion for estimating whether further generations significantly improve the physical results obtained. However, the following issues must be considered in this application:

- A relatively small set of solutions is required, i.e., here it is not necessary to use unconstrained archives to find the whole set of Pareto-optimal solutions.
- Convergence of solutions is more important than a good distribution of solutions along the approximated Pareto-optimal front. Further convergence only makes sense up to a certain threshold, after which the improvements are no longer of physical significance because of other sources of errors (e.g., in the reference or the model).
- As calculating the objective functions is the most time consuming part in this application, the number of generations, and also the number of particles in the population should be selected as low as possible.

For obtaining a stopping criterion for this case study, we run the MOPSO for a large number of generations, e.g., 10000. Then, the solutions of every 500 generations are compared in terms of their convergence. The results of MOPSO after 10000 generations is shown in Figure 7.9. It can be observed that the MOPSO obtained lower objective values than when it is run for 3000 generations. This is also confirmed quantitatively:

$$\begin{aligned} C(MOPSO^{10000}, MOPSO^{3000}) &= 85\% \\ C(MOPSO^{3000}, MOPSO^{10000}) &= 0\% \end{aligned}$$

For finding the least number of generations, the following experiment was performed: After running 10 independent optimizations using MOPSO for each of the runs we compared the non-dominated set (A_t) of generation t with the non-dominated set of generation $t - \Delta t$ $(A_{t-\Delta t})$ using the C metric.
Figure 7.8 shows the trend of this measure, averaged over these 10 runs, together with its standard deviation. For further examinations in this chapter, one of the results (run1) is being selected.

We can observe that after a certain number of generations (approx. 6500) the measure is nearly constant and below 20%. From this, we assume that the differences between following non-dominated sets $[A_{t-\Delta t}, A_t]$ are small and the cost/gain of the accuracy ratio, i.e., the amount of generations required for further improvements, is high. Hence, we suggest to take a C metric threshold as termination criteria instead of running it for a fixed number of generations. Later, the results obtained after 6500 generations, and the final results after 10000 generations (for run1) are analyzed. When comparing the non-dominated sets in objective space, we can see that the differences between them are minor.

7.2.7 Analysis of Physical Properties

In the following, four solutions (A, B, C, D), taken from the approximated Pareto-fronts by MOPSO after 6500 and 10000, are further analyzed. The selected solutions A, B, and C have the best objective functions and D is manually picked out of the center of the surface defined by the non-dominated set. The solution D was chosen under the assumption that it represents a reasonable compromise for the different objective functions. The position of the parameter sets in objective space are shown in Figure 7.9. The corresponding objective values are listed in Table 7.1. The

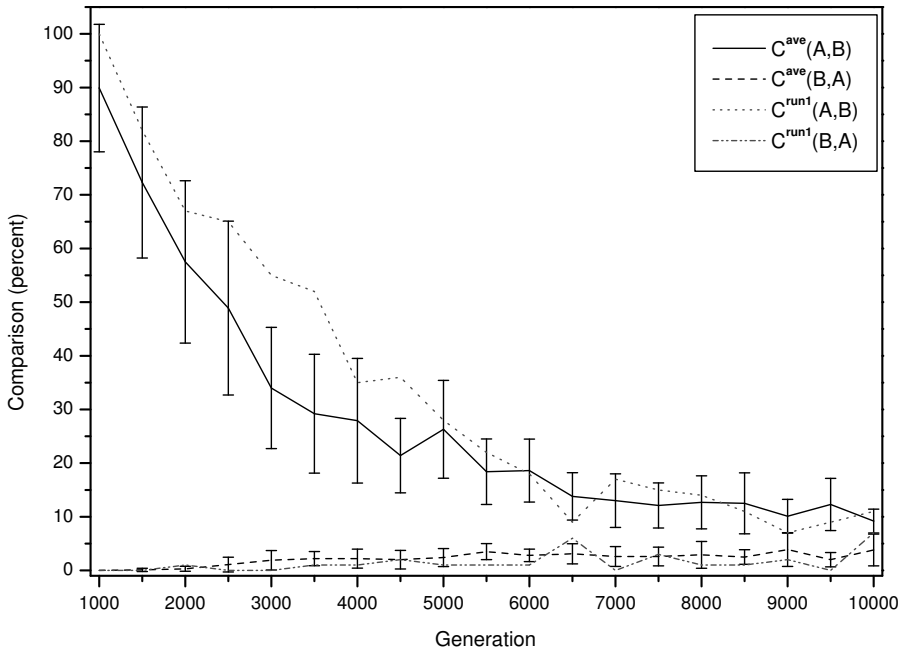


Figure 7.8: C metric comparison of the non-dominated set of generations t and $t - \Delta t$ (A = A_t, B = $A_{t-\Delta t}$, $\Delta t = 500$)

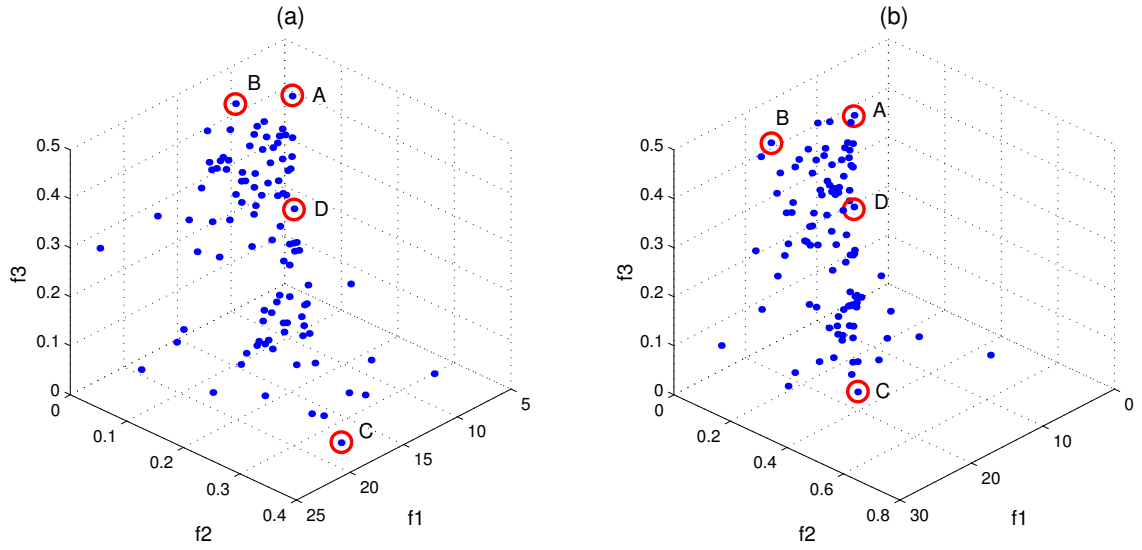


Figure 7.9: The non-dominated set obtained using MOPSO after (a) 6500 and (b) 10000 generations

Table 7.1: Objective values for the parameter sets picked from run 1 after 6500 and 10000 generations. The line max. denotes the maximum value of the respective objective within the non-dominated set.

	f_1	f_2	f_3
A^{6500}/A^{10000}	6.3 / 6.3	0.038 / 0.037	0.42 / 0.40
B^{6500}/B^{10000}	11.5 / 17.7	0.036 / 0.034	0.46 / 0.43
C^{6500}/C^{10000}	17.9 / 17.9	0.345 / 0.345	0.008 / 0.008
D^{6500}/D^{10000}	7.3 / 7.0	0.060 / 0.055	0.21 / 0.22
$max.^{6500}/max.^{10000}$	24.5 / 27.8	0.355 / 0.547	0.46 / 0.43

objectives values defined by f_1 (Equation (7.4)) and f_2 (Equation (7.9)) are not directly related to observable physical properties and hence it is difficult to judge the quality of the force field parameters obtained. Therefore, we analyze more descriptive properties in the following.

Structural Analysis

The molecules were optimized with respect to the energy function defined by the force field in Equations (7.1)-(7.3), using the parameter sets A-D after 6500 and

10000 steps. Some characteristic geometrical parameters, obtained using the parameter sets attained after 6500 generations, are shown in Table 7.2. For all of the sets, the geometrical data shows good agreement with the reference. The deviations from the reference are less than the usual error bar and hence not significant from a physical point of view. The data obtained with the parameter sets after 10000 steps of MOPSO show the same picture (data not shown). We conclude that the accuracy obtained here after 6500 generations is sufficient for application purposes no matter which of the parameter sets A-D is chosen.

Vibrational spectra

The Vibrational spectra were calculated for each of the parameter sets at the energetic minimum. A simple ordering of the vibrational frequencies by magnitude can lead to wrong assignments of vibrations. Hence, the vibrational modes of the different test sets were manually assigned those of the reference. The vibrational spectrum for the two alcohols calculated with parameter set D^{6500} in comparison to the reference is shown in Figure 7.10. For most vibrations of both alcohols, good correspondence is found, although various swappings of vibrational modes can be observed. This is common when comparing different methodologies and is not a problem *per se*, as relative shifts are more meaningful.
A few of the vibrational modes are significantly shifted, leading to the large maximal absolute deviations, as listed in Table 7.3. Closer observation of the vibrational spectrum shows that the modes responsible for the maximum deviations are the

Table 7.2: Characteristical geometry parameters for minimum geometries for parameter sets A-D. Distances d are in Å, angles a in °.

	A^{6500}	B^{6500}	C^{6500}	D^{6500}	Ref.
Methanol					
d(H-O)	0.96	0.96	0.96	0.96	0.96
d(O-C)	1.42	1.42	1.43	1.42	1.43
a(H-O-C)	108.3	108.3	108.5	108.3	108.9
Ethanol					
d(H-O)	0.96	0.96	0.96	0.96	0.96
d(O-C)	1.43	1.43	1.43	1.42	1.43
d(C-C)	1.51	1.51	1.50	1.51	1.52
a(H-O-C)	109.1	109.0	109.1	109.0	109.0
a(O-C-C)	107.3	107.6	107.9	107.5	108.0

same ones for all parameter sets and similar ones for both alcohols. In all of these descriptions, these modes are too high compared to the reference. We can also observe this problem with the CHARMM27 parametrization [MBB+98]. Not considering this problem, the maximum absolute deviations are about 100 cm^{-1} or less. The large maximum absolute deviation for ethanol, using parameter set B^{10000}, is therefore not significant for the overall quality of the fit.
Apart from this deviation, the comparison of the maximum absolute deviations clearly shows that vibrational spectra are generally better described with parameter set B or D than with parameter set C. Parameter set A describes the vibrational modes with comparable quality to the descriptions of sets B and D. This correlates well with the corresponding values of f_2. The same tendency can be observed for the *rmsd* and the mean absolute deviation. The parameter sets B and D again give the best results, whereas set C gives poor results. The quality of description obtained with set A is different for ethanol and methanol. The trends of the quality of description are well reflected in the objective values f_2.

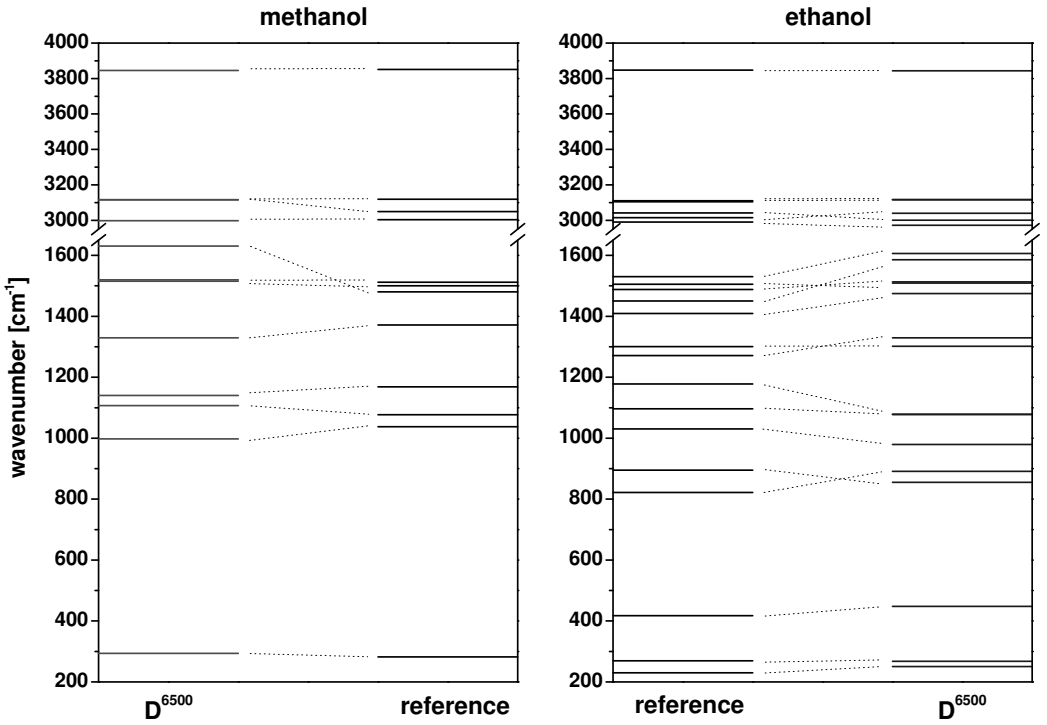


Figure 7.10: Reference vibrational spectra and vibrational spectra obtained with parameter set D^{6500}. The dotted lines connect corresponding modes.

Energetics of different conformations

From both the deviations from the rotational profile by f_3, and the individual rotational profiles, it is obvious that for all of the sets in the approximated Pareto-optimal front, the deviations are well below the accuracy of the reference calculations ($\Delta E \leq \sqrt{f_3}\ kcal/mol$). Also, the qualitative features are well represented for all of the parameter sets considered here. This is depicted in an example for one rotational profile in Figure 7.11 for one bond in ethanol and the parameters obtained after 6500 generations. This particular torsional angle was chosen for illustration, as it shows the most prominent deviations. Even those parameters which lead to the larger deviations in terms of f_3, show the required features of maxima and minima at the respective angles. Nevertheless, we observe that parameter sets C and D best reproduce the rotational profile. When taking a closer look at the curves at 60° we get the impression that the curve generated by parameter set D is superior to set C in its description of the rotational profile. This is counterintuitive on first glance with respect to the objective values f_3. However we have to keep in mind that f_3 measures the fit of *all* rotational profiles, in some of which the deviations of D are larger. Similar behavior as for this dihedral rotation is observed for the other

Table 7.3: Vibrational frequency deviations (in terms of wavenumbers cm^{-1}) for parameter sets A-D

	A^{6500} (A^{10000})	B^{6500} (B^{10000})	C^{6500} (C^{10000})	D^{6500} (D^{10000})
Methanol				
rmsd	51.4	52.1	64.8	52.2
	(51.5)	(52.7)	(64.8)	(52.2)
mean abs. dev.	32.2	32.9	44.9	33.8
	(32.3)	(33.8)	(44.9)	(33.5)
max abs. dev.	145.5	148.0	187.6	150.1
	(145.7)	(149.9)	(187.6)	(148.9)
Ethanol				
rmsd	51.1	51.6	62.0	51.6
	(51.1)	(55.2)	(62.0)	(51.6)
mean abs. dev.	38.6	39.3	49.4	38.2
	(38.6)	(39.7)	(49.4)	(38.3)
max abs. dev.	130.6	126.0	157.8	135.3
	(130.1)	(174.4)	(157.7)	(134.2)

rotational profiles (data not shown).

Discussion

In contrast to previously investigated methods by evolutionary algorithms [Bus01, Reb98, HH99, HBH+98, WK01], the approach presented here delivers a multitude of different solutions. A reasonable choice of a parameter set from the non-dominated set can lead to a good description of *all* physical properties concerned. Also, we introduced a set of objective functions for the purpose of parametrization. The results confirm that the choice of objective functions is suitable for this task.

7.3 Conclusion

This chapter is dedicated to the applications of MOPSO and MOEA on real-world optimization problems. Here, two real-world applications are studied and the results are compared with the results of existing methods.
The first application concerns the multi-objective optimization problem in antenna design. The antenna design problem arises in directing the power of the antenna in a specific direction, and it is assumed that the geometry of the antenna is fixed. This problem has already been studied in Chapter 4 by HMOEA methods. Here, the

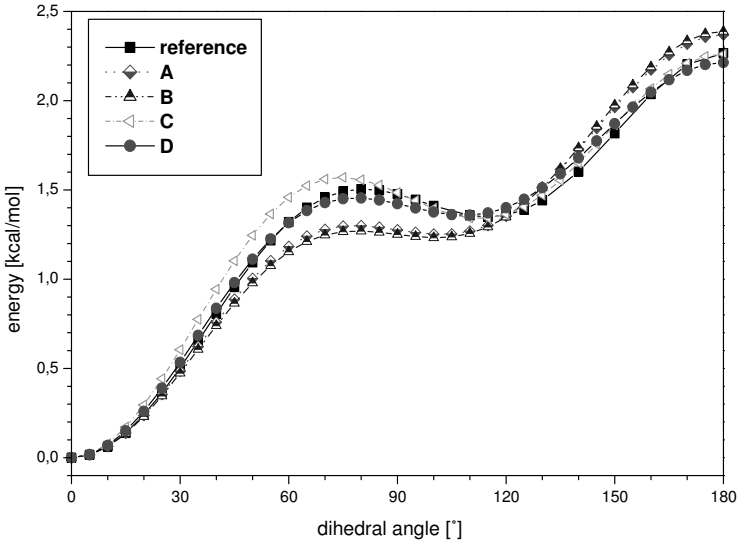


Figure 7.11: Comparison of the rotational profile around the C-C[–]O-H bond in ethanol. The rotational angles and energies are relative to the conformational minimum at 0 ° (staggered conformation).

MOPSO method is applied on it and compared with the existing results of HMOEA. The results show that MOPSO obtains more solutions than the MOEA and even HMOEA. The solutions also have better convergence. Covering the approximated front is also studied by the covering MOPSO method.
The second application is focused on the computational chemistry and the parameterization of the molecular force field. This application is studied for the first time by applying the multi-objective optimization methods namely, MOEA and MOPSO methods. The most important issue in solving this MOP is that the Pareto-optimal front is not known. Therefore, the physical analysis plays an important role. Here, the MOPSO method outperforms the MOEA method. Therefore, the results of the MOPSO method are further analyzed physically. The analysis shows that the results of MOPSO are very close to the desired solutions. This is also confirmed by applying the C metric convergence comparison on different generations.

Chapter 8

Conclusion and Outlook

8.1 Fundamental Results

This dissertation emphasizes on methodologies for achieving better convergence and diversity of solutions than existing methods in Multi-objective Evolutionary Algorithms (MOEAs). In detail, the major contributions are summarized as follows:

- Different data structures for the archive in elitist MOEAs are investigated. Quad-tree data structures are presented to potentially increase computational efficiency of MOEAs and proposed for continuous problems with large population sizes and high number of objectives with small archives.
- Controllable exploration of the search space by MOEA is investigated in order to achieve better convergence of solutions than in MOEA. This is achieved by proposing a Hybrid MOEA (HMOEA). This method is used to cover the Pareto-optimal front.
- Multi-objective Particle Swarm Optimization (MOPSO) methods are investigated and compared with MOEAs. A novel strategy in MOPSO is introduced in order to obtain good convergence and diversity of solutions. The method is compared with existing MOEA and MOPSO methods.
- Covering techniques for the Pareto-optimal front have been advanced through a new method using MOPSO. These are compared with HMOEA for a real-world example.
- The clustering and truncation in elitist MOEA and MOPSO often require a high computational time. Here, ϵMOPSO is proposed to omit the clustering process and reduce the computational time of the MOPSO. The influence of using ϵMOPSO on the convergence and diversity of solutions is studied.

- The performances of the above methods are validated on two real-world applications: a) Antenna design problem and b) Molecular force field parameterization in computational chemistry. In the second example, the results are also evaluated by physical analysis.

- A new quantitative measure for comparing and measuring the diversity of a set of non-dominated solutions is developed. This so-called Sigma diversity metric is used to compare different non-dominated sets. The Sigma diversity metric – in contrast to previous approaches – provides us with information about the non-dominated fronts and the distribution of solutions on them.

8.2 Future Directions in MOEA

The presented methods and the validated results illustrate good efficiency, effectiveness, and motivation for future research in the area of multi-objective optimization using particle swarm optimization techniques. However, the future of the multi-objective optimization techniques such as MOEAs and MOPSOs can be classified into a few areas outlined in the following:

- Techniques to deal with dynamic problems: There are several techniques using EA and PSO methods to solve single-objective dynamic problems, e.g., [Bra01, AS02]. However, the question is how to deal with multi-objective dynamic problems, and what are the performance metrics for evaluating the results of a dynamic problem?

- Algorithms that can handle many objectives: Recently, Purshouse and Fleming have analyzed the existing MOEAs for solving MOPs with a large number of objectives [PF03]. It has been shown that the convergence of solutions gets worse for high number of objectives. So the question is: what must be done when dealing with such problems, and how should the performance metrics evaluate the results in a high dimensional objective space?

- Real-world applications: Real-world applications have been the motivation for developing different techniques in MO. Handling large number of parameters and objectives, defining stopping criteria and performance metrics are the challenges, which differ for each application and should be investigated further. Some real-world problems require a large computational time to compute the objective values and therefore methods that demand less evaluation of objective values are desirable.

Altogether, in the future we can also be engaged in other directions, such as handling constraints as objectives, developing covering techniques, and approximating the Pareto-optimal front in less computational time.

Appendix A

Convergence of MOEA

A.1 Background

Definition A.1 (Partially Ordered Sets) *Let $\mathcal{F}$ be a set. A reflexive, antisymmetric, and transitive relation $\preceq$ on $\mathcal{F}$ is termed a partial order relation, whereas a strict partial order relation $\prec$ must be anti-reflexive, antisymmetric, and transitive. If the partial order relation is valid on a set $\mathcal{F}$, then the pair $(\mathcal{F}, \preceq)$ is called a* **partially ordered set** *(poset).*

If $x \prec y$ for some $x,\ y \in \mathcal{F}$ then x is said to dominate y. Distinct points $x,\ y \in \mathcal{F}$ are said to be comparable when either $x \prec y$ or $y \prec x$. Otherwise x and y are incomparable or indifferent to each other.

Definition A.2 (Chain) *If each pair of distinct points of a poset $(\mathcal{F}, \preceq)$ is comparable, then $(\mathcal{F}, \preceq)$ is called a totally ordered set or a* **chain**. *If each pair of distinct points of a poset $(\mathcal{F}, \preceq)$ is incomparable, then $(\mathcal{F}, \preceq)$ is called an* **antichain**.

Definition A.3 (Minimal element) *An element $x^* \in \mathcal{F}$ is called a minimal element of the poset $(\mathcal{F}, \preceq)$, if there is no $x \in \mathcal{F}$ such that $x \prec x^*$. The set of all minimal elements, denoted by $\mathcal{M}(\mathcal{F}, \preceq)$ is said to be complete if for each $x \in \mathcal{F}$ there is at least one $x^* \in \mathcal{M}(\mathcal{F}, \preceq)$, such that $x^* \preceq x$.*

Let $f\ :\ S \to \mathcal{F}$ be a mapping from some set S to the poset $(\mathcal{F}, \preceq)$. For some $A \subseteq S$ the set $\mathcal{M}_f(A, \preceq)$ contains those elements from A whose images are minimal elements in the image space $f(A)$.

$$\mathcal{M}_f(A, \preceq) = \{a \in A\ :\ f(a) \in \mathcal{M}(f(A), \preceq)\} \tag{A.1}$$

A.1.1 Markov Chains

Here, only the required definitions are studied. Please refer to [Häg02] for more details.

Definition A.4 (Homogeneous Markov chain) *Let* $\mathbf{P}$ *be a* $k \times k$ *matrix with elements* $\{\mathbf{P}_{i,j} : i,j = 1,\ldots,k\}$. *A random process* $(X_0, X_1, \ldots)$ *with finite state space* $S = \{s_1,\ldots,s_k\}$ *is said to be a* **homogeneous Markov chain with transition matrix** $\mathbf{P}$, *if for all* n, *all* $i,j \in \{1,\ldots,k\}$ *and all* $i_0,\ldots,i_{n-1} \in \{1,\ldots,k\}$ *we have*

$$\begin{aligned}&\mathbf{P}(X_{n+1} = s_j | X_0 = s_{i_0},\ X_1 = s_{i_1},\ \ldots,\ X_{n-1} = s_{i_{n-1}},\ X_n = s_{i_n})\\ &= \mathbf{P}(X_{n+1} = s_j | X_n = s_i)\\ &= \mathbf{P}_{i,j}. \end{aligned} \tag{A.2}$$

The elements of the transition matrix P are called transition probabilities. The transition probability $\mathbf{P}_{i,j}$ is the conditional probability of being in state s_j in time $t+1$, given that we are in state s_i in $t \geq 0$. Another important characteristic - besides the transition matrix - of a Markov chain $(X_0, X_1, \ldots)$, is the **initial distribution**, which tells us how the Markov chain starts. The initial distribution is represented as a row vector $\mu^{(0)}$, given by

$$\begin{aligned}\mu^{(0)} &= (\mu_1^{(0)}, \mu_2^{(0)}, \ldots, \mu_k^{(0)})\\ &= (\mathbf{P}(X_0 = s_1), \mathbf{P}(X_0 = s_2), \ldots, \mathbf{P}(X_0 = s_k))\end{aligned} \tag{A.3}$$

Since $\mu^{(0)}$ represents a probability distribution, we have

$$\sum_{i=1}^{k} \mu_i^{(0)} = 1 \tag{A.4}$$

It can be proved that the distribution $\mu^{(n)}$ at time n can be calculated by:

$$\mu^{(n)} = \mu^{(0)} P^n \tag{A.5}$$

Irreducibility is another definition, which loosely speaking, is the property that *all states of Markov chain can be reached from all others.* The chain is irreducible if for any $s_i, s_j \in S$ we can find an n such that $(\mathbf{P}^n)_{i,j} > 0$.

Definition A.5 (Stationary Distribution) *Let* $(X_0, X_1, \ldots)$ *be a Markov chain with state space* $S = \{s_1,\ldots,s_k\}$ *and transition matrix* $\mathbf{P}$. *A row vector* $\pi =$

$(\pi_1, \ldots, \pi_k)$ is said to be a **stationary distribution** *for the Markov chain, if it satisfies:*

$$\text{(i) } \pi_i \geq 0 \text{ for } i = 1, \cdots, k \text{ and } \sum_{i=1}^{k} \pi_i = 1$$

$$\text{(ii) } \pi \mathbf{P} = \pi, \text{ meaning that } \sum_{i=1}^{k} \pi_i \mathbf{P_{i,j}} = \pi_j \text{ for } j = 1, \cdots, k \tag{A.6}$$

The existence and uniqueness of stationary distributions have been studied in [Häg02]. It can be shown that *for any irreducible and aperiodic Markov chain, there is at least one stationary distribution.*

Theorem 1 (The Markov chain convergence theorem) *Let $(X_0, X_1, \ldots)$ be an irreducible aperiodic*[1] *Markov chain with state space S, transition matrix $\mathbf{P}$, and arbitrary initial distribution $\mu^{(0)}$. Then for each stationary distribution π for the transition matrix $\mathbf{P}$, μ converges to π as $n \to \infty$.*

A.2 Convergence

Let P_t be the population of an Evolutionary Algorithm (EA) at generation t and $F_t = f(P_t)$ its associated set of objective vectors. The EA is said to converge with probability 1 to the entire set of optimal solutions[2] $\mathcal{F}^*$ if:

$$d(F_t, \mathcal{F}^*) \to 0 \text{ with probability 1, as } t \to \infty \tag{A.7}$$

whereas it is said to converge with probability 1 to the set of optimal solutions $\mathcal{F}^*$ if:

$$\delta_{\mathcal{F}^*}(F_t) \to 0 \text{ with probability 1, as } t \to \infty \tag{A.8}$$

Here, $d(F_t, \mathcal{F}^*)$ refers to the Hamming distance between associated incidence vectors F_t and $\mathcal{F}^*$. It means that the population size will eventually grow at least to the size of the set $\mathcal{F}^*$. $\delta_{\mathcal{F}^*}(F_t)$ means the number of elements that are in F_t but not in $\mathcal{F}^*$.
If there is only a single objective, $(\mathcal{F}, \preceq)$ is a chain, then $|\mathcal{F}^*| = 1$. But if there are several objectives, $(\mathcal{F}, \preceq)$ is not a chain, then $|\mathcal{F}^*| > 1$.

[1] If period $d(s_i)$ of the state s_i is equal to 1, the state s_i is aperiodic. $d(s_i) = gcd\{n \geq 1 : (P^n)_{i,i} > 0\}$. A Markov chain is aperiodic when all its states are aperiodic.

[2] Here, optimal solutions are the minimal elements.

Let S be the finite search space and $f : S \to \mathcal{F} = \{f(x) : x \in S\}$ the fitness function, where $(\mathcal{F}, \preceq)$, is a partially ordered set. The target of the MOEA is to find the set of optimal solutions $\mathcal{M}(\mathcal{F}, \preceq)$. Algorithm 17 shows the base algorithm that is going to be discussed here[3]. In this algorithm, A_t and P_t are the archive and the population in generation t, respectively.

Algorithm 17 : Basic MOEA

$P(0)$ is drawn at random from S
$A(0) = \mathcal{M}_f(P(0), \preceq)$
$t = 0$ generation counter
repeat
  $P_{t+1} = generate(P_t)$
  $A_{t+1} = \mathcal{M}_f(A_t \cup P_{t+1}, \preceq)$
  $t = t + 1$
until stopping criteria fulfilled

The algorithm stores the generated non-dominated solutions of P_t in A_t. Here, the size of A_t can grow to the size of $\mathcal{F}^*$, i.e., the size of the archive is not bounded to a certain size.

A.2.1 Proof

If the sequence $(P_t)_{t\geq 0}$ is a homogeneous finite Markov chain with irreducible transition matrix then $d(f(A_t), \mathcal{F}^*) \to 0$ with probability one as $t \to \infty$.

Proof:

By construction of the algorithm it is guaranteed that the set of objective vectors $f(A_t)$ of A_t is an antichain, because incomparable solutions are stored in A_t. As soon as an element of $\mathcal{F}^* = \mathcal{M}(\mathcal{F}, \preceq)$ has entered $f(A_t)$ it will stay there forever.
It remains to show that all elements of $\mathcal{F}^*$ will be contained in $f(A_\tau)$ for some random time τ with $\mathbf{P}\{\tau < \infty\} = 1$.
Let $P^*(t) = \mathcal{M}_f(P_t, \preceq)$. Notice that $\mathcal{M}_f(A_t \cup P_{t+1}, \preceq) = \mathcal{M}_f(A_t \cup P^*(t+1), \preceq)$.
Let $a \in A(t_0)$ with $f(a) \notin \mathcal{F}^*$. Since $(\mathcal{F}, \preceq)$ is complete it is guaranteed that there exists an element $x \in S$ such that $f(x) \prec f(a)$. Since the Markov chain is irreducible, it ensures that every element in S will be often visited infinitely. This implies that the waiting time of the first occurrence, as well as between two consecutive occurrences of x, is finite with probability one. Therefore, non-optimal

[3]This algorithm is introduced by Rudolph and Agapie in [RA00].

solutions will be eliminated after a finite number of iterations with probability one. Moreover, each element that is incomparable to all elements in A_t will enter A_{t+1}. If it is optimal it remains in A forever, otherwise it will be replaced in finite time by an optimal solution. The appearance of an incomparable element ensures the irreducibility of the Markov chain $(P_t)_{t\geq 0}$.
All optimal solutions will enter A in finite time with probability one and the non-optimal solutions will be discarded. Since optimal solutions can not get lost one gets $d(f(A_t), \mathcal{F}^*) \to 0$ with probability one and due to boundedness of $d(\cdot, \mathcal{F}^*)$.

□

According to this proof we just need to check if the transition matrix is irreducible in order to get convergence results.
Since the transition matrix is a product of several other transition matrices; describing, mutation, cross over, and selection (in *generate* function in Algorithm 17), it is useful to find the irreducibility of the product of the matrices. It can be shown that the cross-over and selection do not possess irreducible matrices. Therefore, the mutation operator must establish irreducibility property. It can be shown that the transition matrix of the mutation for a single point mutation is irreducible. For more details refer to [RA00].

Bibliography

[AS02] E. Costa A. Simões. Using gas to deal with dynamic environments: a comparative study of several approaches based on promoting diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'02*, 2002.

[AYL89] Norman L. Allinger, Young H. Yuh, and Jenn Huei Lii. Molecular mechanics. the MM3 force field for hydrocarbons. 1. *Journal of the American Chemical Society*, 111(23):8551–66, 1989.

[BBO+83] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. Charmm: a program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4(2):187–217, 1983.

[Bäc96] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[BH03] V. Barichard and J. Hao. A population and interval constraint propagation algorithm. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 88–101, 2003.

[BMK03] D. Büche, S. Müller, and P. Koumoutsakos. Self-adaptation for multiobjective evolutionary algorithms. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 267–281, 2003.

[Bra01] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, New York, 2001.

[Bus01] M. Busold. *Quantenchemische Untersuchungen zur Oxidation von Mehrfachbindungen und die Entwicklung einer automatisierten Parametrisierung für Kraftfelder*. PhD thesis, Technische Universität München, Garching, Germany, 2001.

[CDG99] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization.* Mc Graw Hill, 1999.

[CL02] C. A. Coello Coello and M. S. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *IEEE Proceedings World Congress on Computational Intelligence (CEC'02)*, pages 1051–1056, 2002.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* The MIT Press - Mc Graw Hill, 1990.

[CVL02] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-objective Problems.* Kluwer, New York, 2002.

[DAPM00] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 849–858, 2000.

[Deb99] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. In *Evolutionary Computation Journal 7(3)*, pages 205–230, 1999.

[Deb01] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, 2001.

[DH97] M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, pages 293–317, 1997.

[DI89] S. Dasgupta and W. A. Goddard III. Hessian-biased force fields from combining theory and experiment. *Journal of Chemical Physics*, 90:7207–15, 1989.

[DJ98] M. Dellnitz and O. Junge. An adaptive subdivision technique for the approximation of attractors and invariant measures. *Comput. Visual. Sci.*, pages 63–68, 1998.

[DMM03a] K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread Pareto-optimal solutions. In *KanGAL Report No. 2003002*, Indian Institute Of Technology, Kanpur, 2003.

[DMM03b] K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread Pareto-optimal solutions. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 222–236, 2003.

[DO04] E. Dunn and G. Olague. Pareto optimal sensing strategies for an active vision system. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'04)*, Portland, USA, June 2004. to appear.

[DSH03] M. Dellnitz, O. Schütze, and T. Hestermeyer. Covering Pareto sets by multilevel subdivision techniques. *Optimization, Theory and Applications*, 2003.

[DTLZ02] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'02)*, pages 825–831, May 2002.

[EBDC55] Jr Wilson E. B., J. C. Decius, and P. C Cross. *Molecular Vibrations.* McGraw-Hill, 1955.

[Edg81] F. Y. Edgeworth. *Mathematical Physics.* C. Kegan Paul and Company, London, 1881.

[ETP03] M. Ehrgott and D. Tenfelde-Podehl. Nadir values: Computation and use in compromise programming. *European Journal of Operational Research*, 1:119–139, 2003.

[FES02] J.E. Fieldsend, R.M. Everson, and S. Singh. Using unconstrained elite archives for multi-objective optimisation. In *IEEE Transactions on Evolutionary Computation*, 2002.

[FF93] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[Fie03] J. E. Fieldsend. *Novel Algorithms for Multi-Objective Search and their application in Multi-Objective Evolutionary Neural Network Training.* PhD thesis, University of Exeter, 2003.

[Fle03] Mark Fleischer. The measure of Pareto optima: Application to multi-objective metaheuristics. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 519–533, 2003.

[FMA02] A. Farhang-Mehr and S. Azarm. Diversity assessment of Pareto optimal sets: An entropy approach. In *IEEE Proceedings World Congress on Computational Intelligence (CEC'02)*, May 2002.

[Fog95] D. B. Fogel. *Evolutionary Computation: toward a new philosophy of machine intelligence*. IEEE Press, New York, 1995.

[FS02] J. E. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In *The 2002 U.K. Workshop on Computational Intelligence*, pages 34–44, 2002.

[FTS+98] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, A. G. Baboul, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, and J. A. Pople. *Gaussian 98*. Gaussian, Inc., Pittsburgh PA, 1998.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[Hab83] W. Habenicht. Quad trees, a data structure for discrete vector optimization problems. In *Lecture Notes in Economic and Mathematical Systems*, volume 209, pages 136–145, Springer-Verlag, 1983.

[HBH+98] J. Hunger, S. Beyreuther, G. Huttner, K. Allinger, U. Radelof, and L. Zsolnai. How to derive force field parameters by genetic algorithms.

modeling tripod-mo(co)3 compounds as an example. *European Journal of Inorganic Chemistry*, pages 693–702, 1998.

[HE02] X. Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'02)*, pages 1677–1681, May 2002.

[HES03] X. Hu, R. Eberhart, and Y. Shi. Particle swarm with extended memory for multiobjective optimization. In *IEEE Swarm Intelligence Symposium*, pages 193–197, 2003.

[Häg02] O. Häggström. *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, 2002.

[HH99] J. Hunger and G. Huttner. Optimization and analysis of force field parameters by combination of genetic algorithms and neural networks. *Journal of Computational Chemistry*, 20:455–471, 1999.

[Hil01] C. Hillermeier. Nonlinear multiobjective optimization: A generalized homotopy approach. *International Series of Numerical Mathematics*, 135, 2001.

[HJ98] M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. In *IMM-REP-1998-7*, Technical report, Institut of Mathematical Modeling, Technical University of Denmark, 1998.

[HNG94] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *1st IEEE Conf on Evolutionary Computation*, volume 1, pages 82–87, 1994.

[Hug03] E. Hughes. Multiobjective binary search optimisation. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 102–117, 2003.

[Jen99] F. Jensen. *Introduction to computational chemistry*. John Wiley & Sons, 1999.

[JJK97] A. Jüschke, J. Jahn, and A. Kirsch. A bicriterial optimization problem of antenna design. *Computational Optimization and Applications*, 7:261–276, 1997.

[KC99] J. Knowles and D. Corne. The Pareto Archived Evolution Strategy: A new baseline algorithm for Pareto multiobjective optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Piscataway, NJ, 1999. IEEE Service Center.

[KC02] J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'02)*, pages 711–716, May 2002.

[KC03] J. Knowles and D. Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. 7(2):100–116, 2003.

[KCF03] J. Knowles, D. W. Corne, and M. Fleischer. Bounded archiving using the Lebesque measure. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'03)*, Canberra, Australia, December 2003.

[KE95] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, Perth, Australia, 1995.

[KE01] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[KTW02] K. Klamroth, J. Tind, and M. Wiecek. Unbiased approximation in multicriteria optimization. *Mathematical Methods of Operations Research*, 56:413–437, 2002.

[Kur91] F. Kursawe. A variant of evolution strategies for vector optimization. In *Parallel Problem Solving from Nature, PPSN I*, pages 193–197, 1991.

[KYD03] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 376–390, 2003.

[Li03] Xiaodong Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation Conference (GECCO'03)*, pages 37–48, 2003.

[LRS98] M. Laumanns, G. Rudolph, and H. P. Schwefel. A spatial predator-prey approach to multi-objective optimization: A preliminary study. In *Parallel Problem Solving From Nature - PPSN V, Berlin: Springer*, pages 241–249, 1998.

[LTDZ02] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Archiving with guaranteed convergence and diversity in multi-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO'02)*, pages 439–447, 2002.

[Mac03] A. D. MacKerell. Empirical force fields: Overview and parameter optimization. In *43th Sanibel Symposium*, 2003.

[MBB+98] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, III Reiher W. E., B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *Journal of Physical Chemistry B*, 102(18):3586–3616, 1998.

[MHK+04] S. Mostaghim, M. Hoffmann, P. König, T. Frauenheim, and J. Teich. Molecular force field parameterization using multi-objective evolutionary algorithms. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'04)*, Portland, USA, June 2004.

[Mie99] K. M. Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, 1999.

[Mor80] J. N. Morse. Reducing the size of the non-dominated set: pruning by clustering. *Computers and Operaton Research*, 7(1-2):55–66, 1980.

[MT03a] S. Mostaghim and J. Teich. The role of e-dominance in multi-objective particle swarm optimization. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'03)*, pages 1764–1771, Canberra, Australia, December 2003.

[MT03b] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 26–33, Indianapolis, USA, 2003.

[MT04a] S. Mostaghim and J. Teich. Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'04)*, Portland, USA, June 2004.

[MT04b] S. Mostaghim and J. Teich. Quad-trees: A data structure for storing Pareto-sets in multi-objective evolutionary algorithms with elitism. In *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, 2004. to appear.

[MTT02] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of data structures for storing Pareto-sets in MOEAs. In *IEEE Proceedings World Congress on Computational Intelligence (CEC'02)*, pages 843–849, May 2002.

[ND03] P. K. S. Nain and K. Deb. A computationally effective multi-objective search and optimization technique using coarse-to-fine grain modeling. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'01)*, pages 1281–1289, 2003.

[OJS03] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'03)*, pages 878–885, Canberra, Australia, December 2003.

[Par06] V. Pareto. *Manuale di Economica Politica, Translated into English by A. S. Schwier as Manual of Political Economy, Edited by A. S. Schwier and A. N. Page, A. M. Kelley, New York, 1971.* Societa Editrice Libaria, Milan, 1906.

[PC03] J. W. Ponder and D. A. Case. Force fields for protein simulations. *Advances in protein chemistry*, 66:27–85, 2003.

[PF03] R. C. Purshouse and P. J. Fleming. Evolutionary multi-objective optimisation: An exploratory analysis. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'01)*, pages 2066–2073, 2003.

[PFPB79] P. Pulay, G. Fogarasi, F. Pang, and J. E. Boggs. Systematic ab initio gradient calculation of molecular geometries, force constants, and dipole moment derivatives. *Journal of the American Chemical Society*, 101(10):2550–2560, 1979.

[PV02a] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[PV02b] K. E. Parsopoulos and M. N. C. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings on the 2002 ACM Symposium on Applied Computing (SAC'02)*, pages 603–607, 2002.

[PY00] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, 2000.

[RA00] G. Rudolph and A. Agapie. Convergence properties of some Multi-Objective Evolutionary Algorithms. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 1010–1016, Piscataway, NJ, USA, 2000. IEEE Service Center.

[Reb98] R. Rebentisch. *Theoretische Methoden zur Interpretation von Molekülspektren.* PhD thesis, Universität zu Köln, Germany, 1998.

[Rud94] G. Rudolph. Convergence of non-elitist strategies. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 63–66, 1994.

[Rud98] G. Rudolph. On a Multi-Objective Evolutionary Algorithm and its convergence to the Pareto set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, IEEE Press, 1998.

[SBR01] T. Strassner, M. Busold, and H. Radrich. FFGenerAtor 2.0 - an automated tool for the generation of MM3 force field parameters. *Journal of Molecular Modeling*, 7(1):374–377, 2001.

[Sch85] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and Thei Applications*, pages 93–100, 1985.

[Sch95] Jason R. Schott. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization.* Master's Thesis, Air Force Institute of Technology, Ohio, USA, 1995.

[Sch03] O. Schütze. A new data structure for nondominance problem in multi-objective optimization. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 509–518, 2003.

[Sch04] O. Schütze. *Set Oriented methods for global optimization.* PhD Thesis, University of Paderborn, 2004. to appear.

[SD94] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. In *Evolutionary Computation*, pages 221–248, 1994.

[SE98] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. *Evolutionary Programming*, pages 591–600, 1998.

[SKW01] B. Schandl, K. Klamroth, and M. M. Wiecek. Norm-based approximation in bicriteria programming. *Computational Optimization and Applications, Kluwer Academic Publishers*, 1(20):23–42, 2001.

[SMDT03] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multilevel evolutionary subdivision techniques. In *Proceedings of Second International Conference on Evolutionary Multi-Criterion Optimization*, pages 118–132, 2003.

[Spe98] W. M. Spears. *Evolutionary Algorithms: The Role of Mutation and Recombination.* Springer-Verlag, 1998.

[SS96a] M. Sun and R. E. Steuer. Interquad: An interactive quad tree based procedure for solving the discrete multiple criteria problem. *European Journal of Operational Research*, 89:462–472, 1996.

[SS96b] M. Sun and R.E. Steuer. Quad trees and linear list for identifying nondominated criterion vectors. *INFORMS Journal on Computing*, 8(4):367–375, 1996.

[SSW02] S. Schäffler, R. Schultz, and K. Weinzierl. A stochastic method for the solution of unconstrained vector optimization problems. *Optimization, Theory and Application*, 2002.

[Tei01] J. Teich. Pareto-front exploration with uncertain objectives. *Lecture Notes in Computer Science (LNCS)*, 1993:314–328, March 2001.

[TLK01] K. C. Tan, T. H. Lee, and E. F. Khor. Evolutionary algorithms for multi-objective optimization: Performance assesment and comparisons. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC'01)*, 2001.

[Vel99] D. A. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classification, Analyses and New Innovations.* PhD Thesis, Air Force Institute of Technology, Ohio, USA, 1999.

[VL00] D. A. Van Veldhuizen and G. B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proc. CEC'00, the Congress on Evolutionary Computation*, 2000.

[WK01] J. Wang and P. A. Kollman. Automatic parameterization of force field by systematic search and genetic algorithms. *Journal of Computational Chemistry*, 22(12):1219–1228, 2001.

[WZ99] A. H. Wright and Y. Zhao. Markov chain models of genetic algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 734–741. Morgan Kaufmann, 1999.

[Zad63] L. Zadeh. Optimality and non-scaler-valued performance criteria. *IEEE Transaction on Automatic Control*, 8, 1963.

[ZDT00] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Journal of Evolutionary Computation*, 8(2):173–195, 2000.

[Zit99] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* Shaker Verlag, Germany, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Swiss Federal Institute of Technology (ETH) Zurich, 1999.

[ZLT02] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. In *Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, pages 95–100, Barcelona, Spain, 2002.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.

[ZTL+02] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. In *TIK Report Nr. 139*, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2002.

[Zyd03] J. B. Zydallis. *Explicit Building Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development.* PhD Thesis, Air Force Institute of Technology, Ohio, USA, 2003.

List of Figures

List of Tables

Index