Universidade do Minho
Escola de Engenharia

Roman Denysiuk

Evolutionary Multiobjective Optimization: Review, Algorithms, and Applications

outubro de 2013

Evolutionary Multiobjective Optimization:
Review, Algorithms, and Applications
Roman Denysiuk
UMinho | 2013

Universidade do Minho
Escola de Engenharia

Roman Denysiuk

Evolutionary Multiobjective Optimization: Review, Algorithms, and Applications

Programa Doutoral em Engenharia Industrial e Sistemas

Trabalho realizado sob a orientação do
Professor Doutor Lino Costa
e da
Professora Doutora Isabel Espírito Santo

outubro de 2013

Abstract

Many mathematical problems arising from diverse fields of human activity can be formulated as optimization problems. The majority of real-world optimization problems involve several and conflicting objectives. Such problems are called multiobjective optimization problems (MOPs). The presence of multiple conflicting objectives that have to be simultaneously optimized gives rise to a set of trade-off solutions, known as the Pareto optimal set. Since this set of solutions is crucial for effective decision-making, which generally aims to improve the human condition, the availability of efficient optimization methods becomes indispensable.

Recently, evolutionary algorithms (EAs) have become popular and successful in approximating the Pareto set. The population-based nature is the main feature that makes them especially attractive for dealing with MOPs. Due to the presence of two search spaces, operators able to efficiently perform the search in both the decision and objective spaces are required. Despite the wide variety of existing methods, a lot of open research issues in the design of multiobjective evolutionary algorithms (MOEAs) remains.

This thesis investigates the use of evolutionary algorithms for solving multiobjective optimization problems. Innovative algorithms are developed studying new techniques for performing the search either in the decision or the objective space. Concerning the search in the decision space, the focus is on the combinations of traditional and evolutionary optimization methods. An issue related to the search in the objective space is studied in the context of many-objective optimization.

Application of evolutionary algorithms is addressed solving two different real-world problems, which are modeled using multiobjective approaches. The problems arise from the mathematical modelling of the dengue disease transmission and a wastewater treatment plant design. The obtained results clearly show that multiobjective modelling is an effective approach. The success in solving these challenging optimization problems highlights the practical relevance and robustness of the developed algorithms.

Resumo

Muitos problemas matemáticos que surgem nas diversas áreas da actividade humana podem ser formulados como problemas de otimização. A maioria dos problemas do mundo real envolve vários objetivos conflituosos. Tais problemas chamam-se problemas de otimização multiobjetivo. A presença de vários objetivos conflituosos, que têm de ser otimizados em simultâneo, dá origem a um conjunto de soluções de compromisso, conhecido como conjunto de soluções ótimas de Pareto. Uma vez que este conjunto de soluções é fundamental para uma tomada de decisão eficaz, cujo objetivo em geral é melhorar a condição humana, o desenvolvimento de métodos de otimização eficientes torna-se indispensável.

Recentemente, os algoritmos evolucionários tornaram-se populares e bem sucedidos na aproximação do conjunto de Pareto. A natureza populacional é a principal característica que os torna especialmente atraentes para lidar com problemas de otimização multiobjetivo. Devido à presença de dois espaços de procura, operadores capazes de realizar a procura de forma eficiente, tanto no espaço de decisão como no espaço dos objetivos, são necessários. Apesar da grande variedade de métodos existentes, várias questões de investigação permanecem em aberto na área do desenvolvimento de algoritmos evolucionários multiobjetivo.

Esta tese investiga o uso de algoritmos evolucionários para a resolução de problemas de otimização multiobjetivo. São desenvolvidos algoritmos inovadores que estudam novas técnicas de procura, quer no espaço de decisão, quer no espaço dos objetivos. No que diz respeito à procura no espaço de decisão, o foco está na combinação de métodos de otimização tradicionais com algoritmos evolucionários. A questão relacionada com a procura no espaço dos objetivos é desenvolvida no contexto da otimização com muitos objetivos.

A aplicação dos algoritmos evolucionários é abordada resolvendo dois problemas reais, que são modelados utilizando abordagens multiobjetivo. Os problemas resultam da modelação matemática da transmissão da doença do dengue e do desenho ótimo de estações de tratamento de águas residuais. O sucesso na resolução destes problemas de otimização constitui um desafio e destaca a relevância prática e robustez dos algoritmos desenvolvidos.

Acknowledgments

I would like to thank my advisors Lino Costa and Isabel Espírito Santo for introducing me to multiobjective optimization, giving me the opportunity to do postgraduate research, and helping me through this time. I am also thankful to Teresa Monteiro for believing in me and helping me. I am grateful to my parents Tetyana and Vitaliy, who made my education possible. Finally, I would like to thank my sisters Hanna and Oksana for encouragement and support.

To my family.

Contents

List of Tables

List of Figures

List of Acronyms

AbYSS archive-based hybrid scatter search

AMOSA archived multiobjective simulated annealing

ASM activated sludge model

BSM benchmark simulation model

CEC congress on evolutionary computation

CMA-ES covariance matrix adaptation evolution strategy

CMX center of mass crossover

C-PCA correntropy principal component analysis

CSTR completely stirred tank reactor

DDMOA descent directions-guided multiobjective algorithm

DDMOA2 generalized descent directions-guided multiobjective algorithm

DE differential evolution

DEMR differential evolution with mutation restriction

DF dengue fever

DHF dengue hemorrhagic fever

DRA dynamical resource allocation

DTLZ Deb-Thiele-Laumanns-Zitzler

EA evolutionary algorithm

EDA estimation of distribution algorithm

EMO evolutionary multiobjective optimization

EMyO evolutionary many-objective optimization

EMyO-C evolutionary many-objective optimization algorithm with clustering-based selection

EMyO-IGD evolutionary many-objective optimization algorithm with inverted generational distance-based selection

ER external repository

ES evolution strategy

ESP evolution strategy with probabilistic mutation

GA genetic algorithm

GA_{AL} genetic algorithm with augmented Lagrangian

GD generational distance

GDE generalized differential evolution

HGPSAL hybrid genetic pattern search augmented Lagrangian algorithm

HJ Hooke and Jeeves

HJ_{AL} Hooke and Jeeves with augmented Lagrangian

HypE hypervolume estimation algorithm

IBEA indicator-based evolutionary algorithm

IGD inverted generational distance

jMetal metaheuristic algorithms in Java

LH Latin hypercube

MEES multiobjective elitist evolution strategy

MMEA probabilistic model-based multiobjective evolutionary algorithm

MO multiobjective optimization

MO-CMA-ES multiobjective covariance matrix adaptation evolution strategy

MOEA multiobjective evolutionary algorithm

MOEA/D multiobjective evolutionary algorithm based on decomposition

MOGA multiobjective genetic algorithm

MO-HGPSAL multiobjective hybrid genetic pattern search augmented Lagrangian algorithm

MO-$HGPSAL_{NBI}$ multiobjective hybrid genetic pattern search augmented Lagrangian

algorithm with normal boundary intersection method

MO-HGPSAL$_{\mathbf{NC}}$ multiobjective hybrid genetic pattern search augmented Lagrangian algorithm with normal constraint method

MO-NSGA-II reference-point based many-objective nondominated sorting genetic algorithm

MOP multiobjective optimization problem

MOPSO multiobjective particle swarm optimization

MOSA multiobjective simulated annealing

MSOPS multiple single objective Pareto sampling

MVU maximum variance unfolding

NBI normal boundary intersection

NC normal constraint

NSDE nondominated sorting differential evolution

NSDEMR nondominated sorting differential evolution with mutation restriction

NSGA nondominated sorting genetic algorithm

NSGA-II elitist nondominated sorting genetic algorithm

OC optimal control

ODE SEIR+ASEI ordinary differential equations susceptible exposed infected resistant + aquatic susceptible exposed infected

PAES Pareto-archived evolution strategy

PCA principal component analysis

PCSEA Pareto corner search evolutionary algorithm

PDE Pareto differential evolution

PF Pareto optimal front

PISA a platform and programming language independent interface for search algorithms

PS Pareto optimal set

PSO particle swarm optimization

PWWF peak wet weather flow

RM-MEDA regularity model-based multiobjective estimation of distribution algorithm

SA simulated annealing

SBX simulated binary crossover

SMPSO speed-constrained multiobjective particle swarm optimization

SMS-EMOA S metric selection evolutionary multiobjective algorithm

SOD-CNT sub-objective dominance count procedure

SOP single-objective optimization problem

SP secondary population

SPEA strength Pareto evolutionary algorithm

SPX simplex crossover

SS scatter search

VEGA vector evaluation genetic algorithm

WFG Walking Fish Group

WHO World Health Organization

WWTP wastewater treatment plant

ZDT Zitzler-Deb-Thiele

Chapter 1

Introduction

1.1 Motivation

An optimization problem can be simply described as the problem of finding the best solution from all feasible solutions. Such problems do not only occur in science and engineering fields but also in the decision-making, which is an integral part of daily life. As an example, consider a task of buying a car. The buyer seeks to purchase the car that best meets his preferences, given a set of restrictions (e.g. a limited budget).

If only one criterion is considered, the given decision-making problem can be formulated as a single-objective optimization problem (e.g., the only concern is the car price, so the buyer, like any customer, seeks to minimize the cost of purchase). In this case, the existence of a single solution (car) that meets buyer requirements is quite evident (it is possible to find a car at the lowest price). However, in many real-world problems there are several and often conflicting objectives that have to be simultaneously considered. The optimization problems with more than one objective function are commonly known as multiobjective optimization problems.

In our example, suppose the buyer likes rapid sport cars with powerful engines, but also wants to minimize the cost of his purchase. So the buyer's objectives can be determined as follows: on one hand, the buyer seeks to maximize the car engine power when selecting the

car; on the other hand, the buyer seeks to make a purchase at the lowest possible cost. In his search for a new car, the buyer remarks that a rapid sport car can be bought at a high cost, but when sacrificing engine power he can spend much less money for the purchase. Moreover, considering two cars with the same characteristics it is obvious that a car at the lower cost is preferable; or, from the equally priced cars, the one with a more powerful engine is preferable. Discarding unpreferable options the buyer reduces the set of possible alternatives. Eventually, he ends up with a set of cars where a gain in price does not occur without a loss in engine power. Since none of these found alternatives can be considered to be superior than other in this set, they all are optimal. This set represents different trade-offs between the two criteria (price and engine power). To select a final single alternative from the obtained optimal set, further preference information is required.

Although a multiobjective optimization is basically an optimization process, there are fundamental differences between single-objective and multiobjective optimization. In the case of single-objective optimization, one seeks to find a single optimal solution. Since minimization is often assumed, an optimal solution means a solution with the minimum value of the given objective function. On the other hand, in multiobjective optimization (MO) there is no generally accepted definition of optimum as it is in single-objective optimization.

When several objectives are optimized at the same time, the search space becomes partially ordered. As a consequence, there is no longer a single optimal solution but a set of optimal solutions. This set contains equally important solutions that represent different trade-offs between the given objectives. This set is generally known as the Pareto optimal set, or Pareto set (PS) for short. Approximating the Pareto set is the main goal in multiobjective optimization. However, achieving this goal is not an easy task. The obtained solutions that aim to approximate the Pareto set must be as close as possible to the true Pareto set. On the other hand, as many as possible solutions must be found. Since from the practical point of view usually it is not possible to generate the whole Pareto set, the generated set of solutions must cover the entire Pareto set and be as diverse as possible. Due to the existence of two different and somewhat conflicting goals – the convergence and

diversity – the task of approximating the Pareto set is multiobjective in nature. The first goal is similar to the optimality goal in single-objective optimization. The second goal is specific to multiobjective optimization.

Furthermore, since a multiobjective optimization problem consists of different objectives, it inherits all properties of its single-objective functions. There are several properties of objective functions that can make even single-objective optimization difficult, namely, multimodality, high-dimensionality, non-separability, deceptiveness, etc. Thus, a multiobjective problem possesses all these difficulties in the decision space. At the same time, the presence of multiple conflicting objectives adds additional complexity requiring to perform the search in the objective space as well. Two search spaces and two goals of approximating optimal solutions constitute the fundamental differences between single-objective and multiobjective optimization. In general, they make multiobjective optimization more difficult than single-objective optimization.

Algorithms for solving multiobjective optimization problems must be able to successfully deal with all of the aforementioned difficulties. They must be able to efficiently explore the search space and find a set of optimal solutions. Mechanisms for providing the convergence to the Pareto optimal region and for maintaining a diversity of obtained solutions must be implemented to an algorithm.

The present thesis addresses multiobjective optimization using evolutionary algorithms. As their single-objective counterparts, multiobjective evolutionary algorithms mimic the principles of natural evolution to perform the search. Due to their ability to simultaneously deal with a set of candidate solutions and to approximate the Pareto set in a single simulation run, MOEAs became especially attractive for solving multiobjective problems. During the last two decades, they have been successfully applied to solve many real-world multiobjective optimization problems, thereby highlighting their robustness. However, growing complexity of problems emerging in different fields of human activity such as science, engineering, business and medicine, among others, provides new challenges and requires powerful tools for solving these problems. This thesis explores the methodology of evolutionary multiobjective optimization (EMO) and aims to promote this area of research.

1.2 Outline of the Thesis

As the title suggests, the present thesis consists of three major parts: review, algorithms, and applications. Each part comprises a specific aspect of evolutionary multiobjective optimization addressed in this thesis.

Chapter 2 provides the definitions and essential background necessary for the work on multiobjective optimization presented in the following chapters.

Chapter 3 reviews a number of approaches developed for dealing with multiobjective optimization problems. The review includes classical methods and evolutionary algorithms. Additionally, EMO algorithms designed specifically for handling many-objective problems are discussed.

Chapter 4 addresses the issue of performance assessment of multiobjective optimization algorithms. In particular, discussions of the existing test suites, unary quality indicators, and statistical comparison methods used for the performance assessment of multiobjective optimizers are presented. Some of presented techniques are used throughout the thesis for the performance evaluation of the herein proposed algorithms.

Chapter 5 introduces a hybrid approach for constrained single-objective optimization. This approach combines a genetic algorithm with a pattern search method, within an augmented Lagrangian framework for constraint handling. After evaluating its performance on a set of constrained single-objective problems, this technique is extended to deal with multiobjective problems.

Chapter 6 suggests a hybrid multiobjective evolutionary algorithm termed descent directions-guided multiobjective algorithm (DDMOA). DDMOA borrows the idea of generating new candidate solutions from Timmel's method. The proposed framework combines paradigms of classical methods and evolutionary algorithms.

Chapter 7 presents a generalized descent directions-guided multiobjective algorithm (DDMOA2). The convergence property of the original algorithm is improved by allowing all the solutions to participate in the reproduction process. Also, the algorithm is extended for handling many-objective problems by introducing the scalarizing fitness assignment into

the selection process.

Chapter 8 focuses on many-objective optimization. Differential evolution with modified reproduction operator is used as the basis. Two different selection schemes able to guide the search in a high dimensional objective space are proposed.

Chapter 9 reports on applications of EMO algorithms to solve the problem arising from a mathematical modelling of the dengue disease transmission. The performance comparison of different algorithms is carried out on this problem. The obtained trade-off solutions are presented, and different scenarios of the dengue epidemic are discussed.

Chapter 10 considers the problem of finding optimal values of the state variables in a wastewater treatment plant design. Different modelling methodologies, consisting in defining and simultaneously optimizing several objectives, are addressed. The discussion of the obtained results and the analysis of the used approaches to find optimal values are presented.

Chapter 11 concludes the work and discusses some possible future research.

1.3 Contributions

The main contributions of this thesis are:

- A contemporary state-of-the-art of multiobjective optimization. The review considers different issues related to multiobjective optimization, including theoretical background, classical methods, evolutionary algorithms, and performance assessment of multiobjective optimizers. The main focus is on multiobjective evolutionary algorithms, being their discussion is organized according to reproduction operators. Additional emphasis is put on algorithms designed to deal with many-objective problems.

- A new hybrid evolutionary algorithm [35, 150]. The proposed approach combines a genetic algorithm with a pattern search method, within an augmented Lagrangian

framework for handling constrained problems. The approach is extended to multiobjective optimization adopting frameworks of classical methods for multiobjective optimization.

- A new local search based approach for multiobjective optimization [52, 53, 55]. The approach adopts the idea of generating new candidate solutions from Timmel's method. A local search procedure is introduced to overcome some limitations and to extend its applicability to multiobjective problems with nondifferentiable objective functions. There are two versions of the proposed approach. The first proposal is simpler, yet a viable one. Inspired by promising results of the first version, a generalized approach has been developed, resulting in a highly competitive multiobjective optimization algorithm.

- New selection schemes for evolutionary many-objective optimization [54]. Two different selection schemes are studied using differential evolution with an improved variation operator. The first scheme is indicator-based. It incorporates the inverted generational distance indicator in the selection process. The second scheme aims to provide more flexible and self-adaptive approach. It uses clustering procedure and calculates distances to a reference point to select a representative within each cluster.

- The application of MOEAs to find an optimal control in the mathematical model of the dengue disease transmission. The problem is modeled using a multiobjective approach, which after discretization of the involved system of eight differential equations results in a challenging large-scale optimization problem. The performance of several state-of-the-art EMO algorithms on this real-world problem is also studied.

- The application of the developed algorithms to solve a multiobjective problem arising from a wastewater treatment plant design optimization. Different modelling methodologies to find optimal values of the decision variables in the WWTP design are considered.

Part I

Review

Chapter 2

Multiobjective Optimization Background

2.1 Introduction

As the name suggests, a multiobjective optimization problem is an optimization problem that deals with more than one objective. It is quite evident that the majority of practical decision-making and optimization problems in science and engineering possess a number of different usually conflicting objectives. In the past, because of a lack of suitable approaches these problems have been mostly solved as single-objective optimization problems (SOPs).

The present chapter introduces the basic concepts of multiobjective optimization and the notation used throughout this thesis. The concepts provided in this chapter are necessary for understanding the principles and particularities of multiobjective optimization. Furthermore, the established background is essential for developing efficient approaches to deal with multiobjective optimization problems, which will be described later in this thesis.

Thus, a formal definition and some basic concepts in multiobjective optimization are presented in Section 2.2. Furthermore, optimality conditions for any solution to be optimal in the presence of multiple objectives are discussed in Section 2.3.

2.2 General Concepts

A multiobjective optimization problem with m objectives and n decision variables can be formulated mathematically as follows:

$$\begin{aligned} \text{minimize:} \quad & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^{\mathrm{T}} \\ \text{subject to:} \quad & g_i(\boldsymbol{x}) \leq 0 && i \in \{1, \ldots, p\} \\ & h_j(\boldsymbol{x}) = 0 && j \in \{1, \ldots, q\} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{2.2.1}$$

where $\boldsymbol{x}$ is the decision vector, $\Omega \subseteq \mathbb{R}^n$ and $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, $\boldsymbol{g}(\boldsymbol{x})$ is the vector of inequality constraints, $\boldsymbol{h}(\boldsymbol{x})$ is the vector of equality constraints, $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower and upper bounds of the decision variables, respectively, and $\boldsymbol{f}(\boldsymbol{x})$ is the objective vector defined in the objective space $\mathbb{R}^m$. Minimization is assumed throughout the thesis without loss of generality. From (2.2.1), it can be seen that there are two different spaces. Each solution in the decision space maps uniquely to the objective space. However, the inverse mapping may be non-unique. The mapping, which is performed by a function $\boldsymbol{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, takes place between the n-dimensional decision space and the m-dimensional objective space. Figure 2.1 illustrates the representation of these two spaces and a mapping between them.

In (2.2.1), the statement "minimize" means that the goal is to minimize all objective functions simultaneously. If there is no conflict between objectives, then a solution can be found where every objective function achieves its optimum. However, it is a trivial case which is unlikely to be encountered in practice. Another particular case arises when $m = 1$. In this case, the problem formulated in (2.2.1) is a single-objective optimization problem.

In the following, it is assumed that a problem defined in (2.2.1) does not have a single solution and the number of objectives is not less than two. This means that the objectives are conflicting or at least partly conflicting. Therefore, the presence of multiple conflicting objectives gives rise to a set of optimal solutions, instead of a single optimal solution. This set represents different trade-offs between objectives, and, in the absence of any further information, none of these solutions can be said to be better than other.

Since the objective space is partially ordered solutions are compared on the basis of the

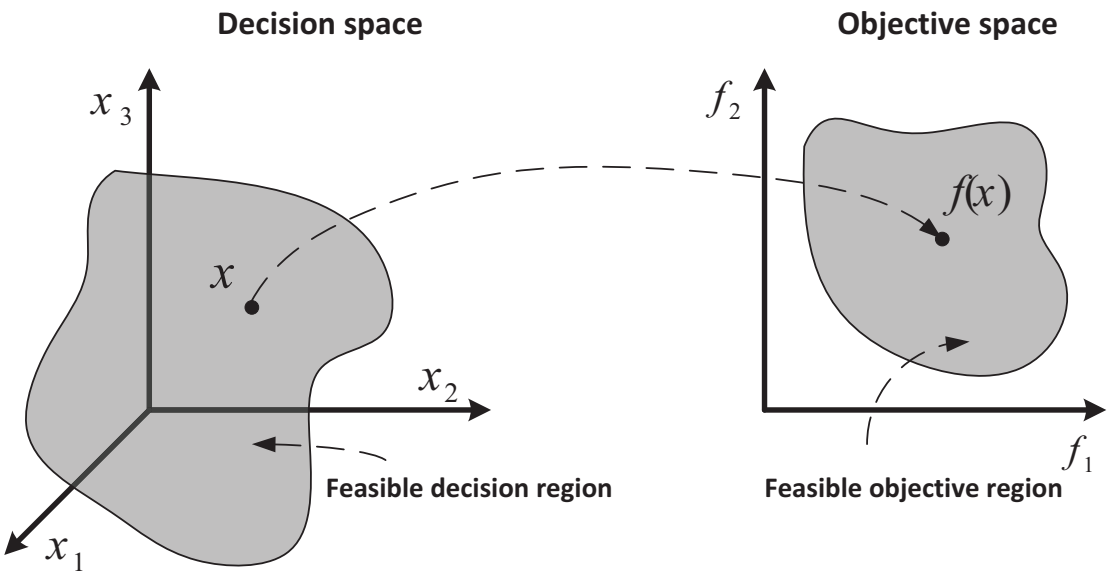


Figure 2.1: Representation of the decision space and the corresponding objective space.

concept of the Pareto dominance.

Definition 2.2.1 (Pareto dominance)**.** *A vector* $\boldsymbol{u} = (u_1, \ldots, u_k)$ *is said to dominate a vector* $\boldsymbol{v} = (v_1, \ldots, v_k)$ *if and only if:*

$$\forall i \in \{1, \ldots, k\} : u_i \leq v_i \wedge \exists j \in \{1, \ldots, k\} : u_j < v_j.$$

Thus, in the context of multiobjective optimization, the following preference relations on the feasible set in the decision space are defined on the basis of the associated objective vectors.

Definition 2.2.2 (weak dominance)**.** *Given two solutions* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *from* Ω*, a solution* $\boldsymbol{a}$ *is said to weakly dominate a solution* $\boldsymbol{b}$ *(denoted by* $\boldsymbol{a} \preceq \boldsymbol{b}$*) if:*

$$\forall i \in \{1, \ldots, m\} : f_i(\boldsymbol{a}) \leq f_i(\boldsymbol{b}).$$

Definition 2.2.3 (dominance)**.** *Given two solutions* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *from* Ω*, a solution* $\boldsymbol{a}$ *is said to dominate a solution* $\boldsymbol{b}$ *(denoted by* $\boldsymbol{a} \prec \boldsymbol{b}$*) if:*

$$\forall i \in \{1, \ldots, m\} : f_i(\boldsymbol{a}) \leq f_i(\boldsymbol{b}) \wedge \exists j \in \{1, \ldots, m\} : f_j(\boldsymbol{a}) < f_j(\boldsymbol{b}).$$

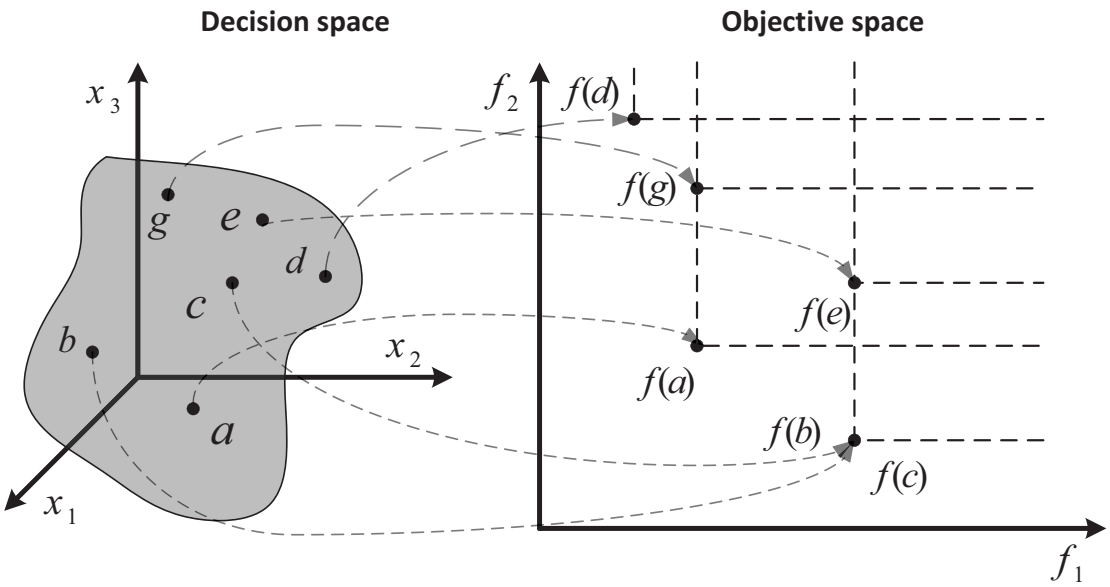


Figure 2.2: Representation of solutions in two different spaces.

Definition 2.2.4 (strict dominance)**.** *Given two solutions* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *from* Ω*, a solution* $\boldsymbol{a}$ *is said to strictly dominate a solution* $\boldsymbol{b}$ *(denoted by* $\boldsymbol{a} \prec\prec \boldsymbol{b}$*) if:*

$$\forall i \in \{1, \ldots, m\} : f_i(\boldsymbol{a}) < f_i(\boldsymbol{b}).$$

Definition 2.2.5 (incomparability)**.** *Given two solutions* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *from* Ω*, a solution* $\boldsymbol{a}$ *is said to be incomparable to a solution* $\boldsymbol{b}$ *(denoted by* $\boldsymbol{a} \| \boldsymbol{b}$*) if:*

$$\boldsymbol{a} \not\preceq \boldsymbol{b} \wedge \boldsymbol{b} \not\preceq \boldsymbol{a}.$$

Definition 2.2.6 (indifference)**.** *Given two solutions* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *from* Ω*, a solution* $\boldsymbol{a}$ *is said to be indifferent (or equivalent) to a solution* $\boldsymbol{b}$ *(denoted by* $\boldsymbol{a} \equiv \boldsymbol{b}$*) if:*

$$\boldsymbol{a} \preceq \boldsymbol{b} \wedge \boldsymbol{b} \preceq \boldsymbol{a}.$$

Figure 2.2 shows a set of solutions $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{g}\}$ and their image in the objective space. From the figure, it can be seen that $\boldsymbol{a} \prec\prec \boldsymbol{e}$, $\boldsymbol{b} \prec \boldsymbol{e}$, $\boldsymbol{b} \equiv \boldsymbol{c}$, $\boldsymbol{a} \| \boldsymbol{d}$, and so on.

The above presented preference relations are also defined on the objective space. However, the indifference relation actually only makes sense with regard to the decision space,

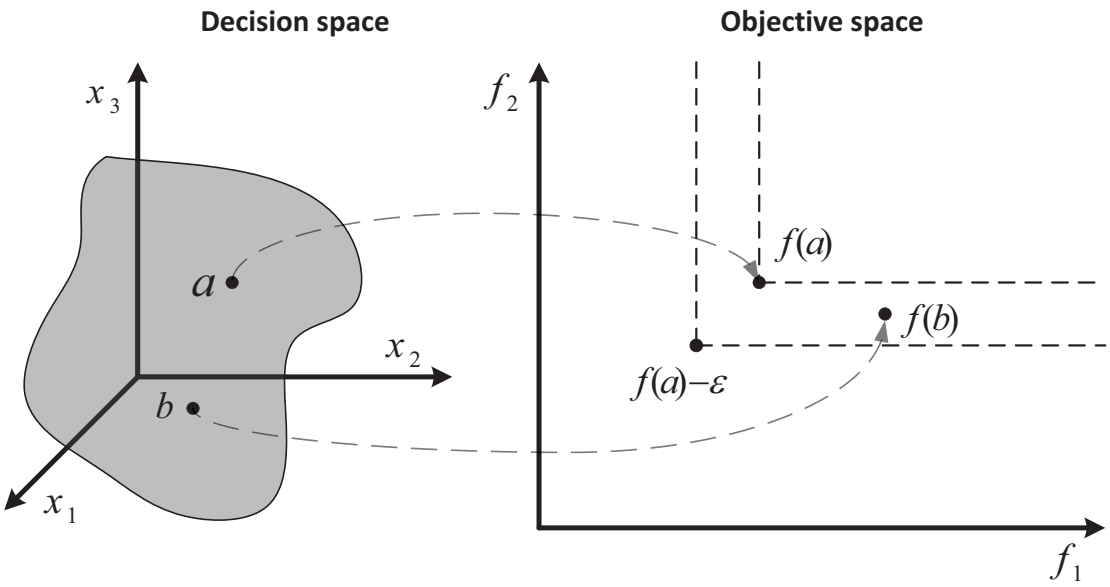


Figure 2.3: Representation of the additive ϵ-dominance.

as in the objective space, it simply means equality. In addition, another widely used preference relation, called ϵ-dominance, is defined. Under ϵ-dominance, the conditions required for one solution to dominate another are relaxed.

Definition 2.2.7 (multiplicative ϵ-dominance)**.** *Given two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$ from Ω, a solution $\boldsymbol{a}$ is said to ϵ-dominate a solution $\boldsymbol{b}$ (denoted by $\boldsymbol{a} \preceq_\epsilon \boldsymbol{b}$) if for a given ϵ:*

$$\forall i \in \{1, \ldots, m\} : (1 - \epsilon) f_i(\boldsymbol{a}) \leq f_i(\boldsymbol{b}).$$

Definition 2.2.8 (additive ϵ-dominance)**.** *Given two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$ from Ω, a solution $\boldsymbol{a}$ is said to ϵ-dominate a solution $\boldsymbol{b}$ (denoted by $\boldsymbol{a} \preceq_\epsilon \boldsymbol{b}$) if for a given ϵ:*

$$\forall i \in \{1, \ldots, m\} : f_i(\boldsymbol{a}) - \epsilon \leq f_i(\boldsymbol{b}).$$

An illustration of the additive ϵ-dominance in the biobjective case is shown in Figure 2.3. As we can see, solution $\boldsymbol{a}$ and solution $\boldsymbol{b}$ are incomparable ($\boldsymbol{a} \| \boldsymbol{b}$). The region that is ϵ-dominated by solution $\boldsymbol{a}$ is composed of the area that would normally be dominated by $\boldsymbol{a}$ plus the areas that would otherwise be nondominated with respect to $\boldsymbol{a}$ or would in fact

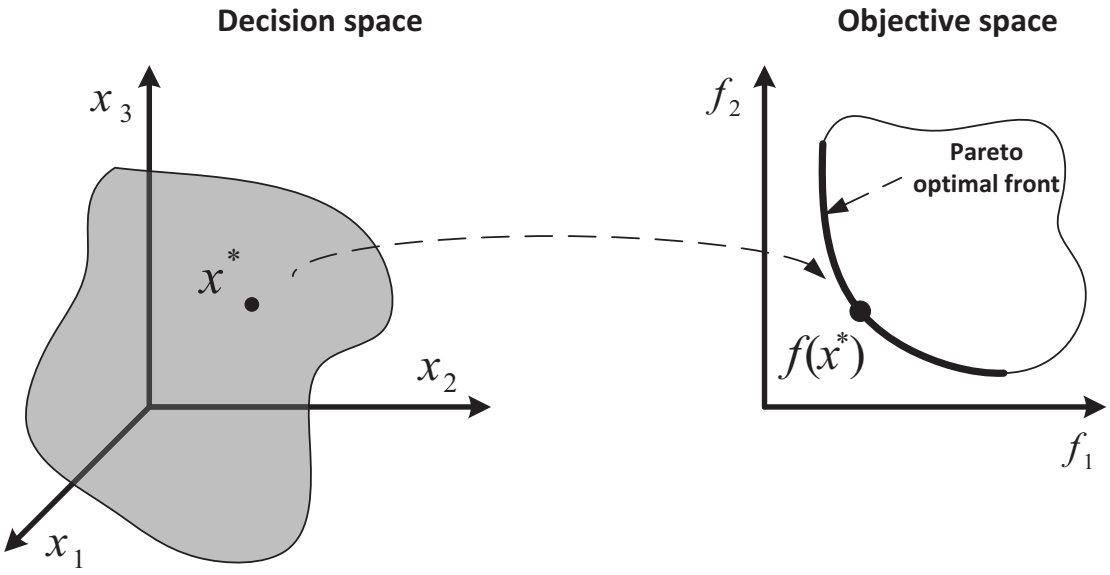


Figure 2.4: Representation of the Pareto optimal front.

dominate $\boldsymbol{a}$. Thus, solution $\boldsymbol{b}$ that is nondominated with respect to $\boldsymbol{a}$ is also ϵ-dominated by $\boldsymbol{a}$ ($\boldsymbol{a} \preceq_\epsilon \boldsymbol{b}$).

The concepts of optimality for multiobjective optimization are defined as follows.

Definition 2.2.9 (Pareto optimality)**.** *A solution $\boldsymbol{x}^* \in \Omega$ is Pareto optimal if and only if:*

$$\nexists \boldsymbol{y} \in \Omega : \boldsymbol{y} \prec \boldsymbol{x}^*.$$

Definition 2.2.10 (Pareto optimal set)**.** *For a given multiobjective optimization problem $\boldsymbol{f}(\boldsymbol{x})$, the Pareto optimal set (or Pareto set for short) is defined as:*

$$\mathcal{PS}^* = \{\boldsymbol{x}^* \in \Omega \,|\, \nexists \boldsymbol{y} \in \Omega : \boldsymbol{y} \prec \boldsymbol{x}^*\}.$$

Definition 2.2.11 (Pareto optimal front)**.** *For a given multiobjective optimization problem $\boldsymbol{f}(\boldsymbol{x})$ and Pareto optimal set $\mathcal{PS}^*$, the Pareto optimal front, or Pareto front (PF) for short, is defined as:*

$$\mathcal{PF}^* = \{\boldsymbol{f}(\boldsymbol{x}^*) \in \mathbb{R}^m \,|\, \boldsymbol{x}^* \in \mathcal{PS}^*\}.$$

Figure 2.4 illustrates the representation of the Pareto optimal front and the Pareto optimal solution $\boldsymbol{x}^*$.

In the following, some special points often used in multiobjective optimization are discussed. These points define the range of the entire Pareto optimal front and are widely used in the decision making process.

Definition 2.2.12 (ideal objective vector)**.** *An objective vector minimizing each of the objective functions is called an ideal objective vector* $\boldsymbol{z}^* \in \mathbb{R}^m$.

The components of the ideal objective vector can be obtained by minimizing each of the objective functions individually. The elements of the ideal objective vector are the lower bounds of all objectives. Thus, for every objective function there is at least one solution in the feasible region sharing an identical value with the corresponding element of the ideal objective vector. However, sometimes there may be required an objective vector that is strictly better than any vector in the feasible objective space. For this purpose, the utopian objective vector is defined as follows.

Definition 2.2.13 (utopian objective vector)**.** *A utopian objective vector* $\boldsymbol{z}^{**}$ *is an objective vector whose components are formed by:*

$$z_i^{**} = z^* - \epsilon_i, \quad \forall i \in \{1, \ldots, m\}$$

where z_i^* *is a component of the ideal objective vector and* $\epsilon_i > 0$.

Unlike the ideal objective vector $\boldsymbol{z}^*$, which represents the lower bound of each objective in the entire feasible objective space, the nadir objective vector $\boldsymbol{z}^{\text{nad}}$ represents the upper bound of each objective in the entire Pareto optimal front. A nadir objective is often confused with a worst objective vector found by using the worst feasible function values.

First a critical point [45] is defined (sometimes called as anchor or corner point), as follows.

Definition 2.2.14 (critical point)**.** *A point* $\boldsymbol{z}^{(j)c}$ *is a critical point with respect to the* j*-th objective function, if it corresponds to the worst value of* f_j *among all Pareto optimal solutions:*

$$\boldsymbol{z}^{(j)c} = \{\boldsymbol{f}(\boldsymbol{y}) \,|\, \boldsymbol{y} = \underset{\boldsymbol{x}^* \in \mathcal{PS}^*}{\operatorname{argmax}} f_j(\boldsymbol{x})\}$$

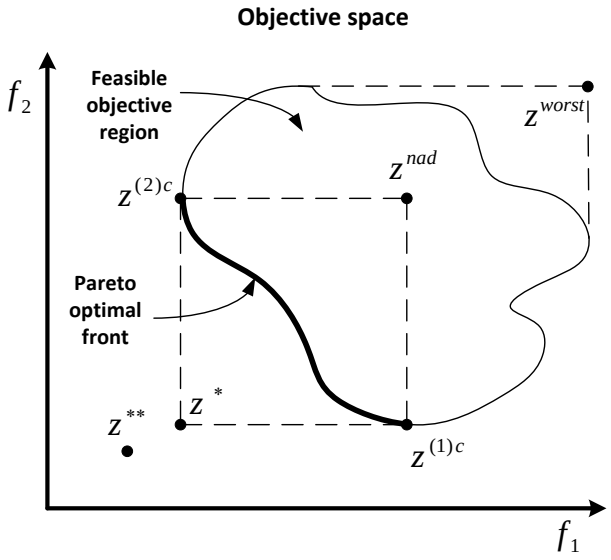


Figure 2.5: Representation of some special points in multiobjective optimization.

The nadir objective vector can be defined as follows.

Definition 2.2.15 (nadir objective vector)**.** *An objective vector whose j-th element is taken from the j-th component of the corresponding critical Pareto point is called a nadir objective vector $\boldsymbol{z}^{\text{nad}} = (z_1^{\text{nad}}, \ldots, z_m^{\text{nad}})^{\text{T}}$ where ($z_j^{\text{nad}} = z_j^{(j)c}$).*

Figure 2.5 illustrates the ideal objective vector $\boldsymbol{z}^*$, the nadir objective vector $\boldsymbol{z}^{\text{nad}}$, the critical points $\boldsymbol{z}^{(1,2)c}$, and the worst objective vector $\boldsymbol{z}^{\text{worst}}$.

The ideal and nadir objective vectors represent the lower and upper bound of the Pareto front for a given multiobjective optimization problem. The reliable estimation of both these vectors is an important issue in multiobjective optimization. However, the estimation of the nadir objective vector may be a quite difficult task. Usually, this involves computing individual optimal solutions for objectives, constructing a payoff table by evaluating other objective values at these optimal solutions, and estimating the nadir point from the worst objective values from the table. This procedure may not guarantee a true estimation of the nadir point for more than two objectives. Thus, in the majority of cases an estimation of the nadir objective vector requires information about the whole Pareto optimal front.

The ideal objective vector and the nadir objective vector are used to normalize objec-

tive functions thereby mapping their values onto the interval $[0, 1]$. The normalization is performed as follows:

$$\overline{f}_i = \frac{f_i - z_i^*}{z_i^{\text{nad}} - z_i^*}, \quad \forall i \in \{1, \ldots, m\},$$

where $\overline{f}_i$ is the normalized objective value.

2.3 Optimality Conditions

Optimality conditions are an important issue in optimization. Therefore, this section presents a set of theoretical optimality conditions for a multiobjective optimization problem. As in single-objective optimization, there are first- and second-order optimality conditions for multiobjective optimization. In the following, a multiobjective optimization problem under consideration is of the form:

$$\begin{aligned} \text{minimize:} \quad & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^{\mathrm{T}} \\ \text{subject to:} \quad & \boldsymbol{x} \in \Omega, \end{aligned} \tag{2.3.1}$$

where $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x}))^{\mathrm{T}} \leq \boldsymbol{0}\}$.

Furthermore, the set of active constraints at a point $\boldsymbol{x}^*$ is denoted by:

$$J(\boldsymbol{x}^*) = \{j \in \{1, \ldots, p\} \,|\, g_j(\boldsymbol{x}^*) = 0\}.$$

All optimality conditions provided in this section assume that all objectives and constraint functions are continuously differentiable. Thus, the following definitions need to be established first.

Definition 2.3.1 (nondifferentiable MOP)**.** *The multiobjective optimization problem is nondifferentiable if some of the objectives or the constraints forming the feasible region are nondifferentiable.*

Definition 2.3.2 (continuous MOP)**.** *If the feasible region Ω is a closed and connected region in $\mathbb{R}^n$ and all the objectives are continuous of $\boldsymbol{x}$ then the multiobjective optimization problem is called continuous.*

Before a convex multiobjective problem is discussed, the definition of a convex function is presented.

Definition 2.3.3 (convex function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if for any two pairs of solutions $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, the following condition is true:*

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha)f(\boldsymbol{y}),$$

for all $0 \leq \alpha \leq 1$.

Definition 2.3.4 (convex MOP)**.** *The multiobjective optimization problem is convex if all objective functions and the feasible region are convex.*

The convexity is an important matter to multiobjective optimization algorithms. Similarly to the property of a convex function, the following theorem establishes a relation between a local and a global Pareto optimal solution for a multiobjective optimization problem.

Theorem 2.3.1. Let the multiobjective optimization problem be convex. Then every locally Pareto optimal solution is also globally Pareto optimal solution.

For a proof, see Miettinen [133].

2.3.1 First-Order Conditions

The following condition is known as the necessary condition for Pareto optimality.

Theorem 2.3.2 (Fritz-John necessary condition for Pareto optimality)**.** Let the objective and the constraint functions of the problem shown in (2.3.1) be continuously differentiable at a decision vector $\boldsymbol{x}^* \in \Omega$. A necessary condition for $\boldsymbol{x}^*$ to be Pareto optimal is that there exist vectors $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$ (where $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\lambda}, \boldsymbol{\mu} \neq \mathbf{0}$) such that:

1. $\sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla g_j(\boldsymbol{x}^*) = 0$
2. $\mu_j g_j(\boldsymbol{x}^*) = 0, \ \forall j \in \{1, \ldots, p\}$.

For a proof, see Da Cunha and Polak [38].

If the multiobjective optimization problem is convex, then there can be stated a sufficient condition for Pareto optimality. Thus, the following theorem offers sufficient conditions for a solution to be Pareto optimal for convex functions.

Theorem 2.3.3 (Karush-Kuhn-Tucker sufficient condition for Pareto optimality)**.** Let the objective and the constraint functions of problem shown in (2.3.1) be convex and continuously differentiable at a decision vector $\boldsymbol{x}^* \in \Omega$. A sufficient condition for $\boldsymbol{x}^*$ to be Pareto optimal is that there exist vectors $\boldsymbol{\lambda} > \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$ (where $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\mu} \in \mathbb{R}^p$) such that:

1. $\sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla g_j(\boldsymbol{x}^*) = 0$
2. $\mu_j g_j(\boldsymbol{x}^*) = 0, \ \forall j \in \{1, \ldots, p\}$.

For a proof, see Miettinen [133].

2.3.2 Second-Order Conditions

Second-order optimality conditions reduce the set of candidate solutions produced by the first-order conditions. However, they tighten the assumptions set to the regularity of the problem.

Furthermore, a definition of the regularity of decision vector follows.

Definition 2.3.5. *A point $\boldsymbol{x}^* \in \Omega$ is said to be a regular point if the gradients of the active constraints at $\boldsymbol{x}^*$ are linearly independent.*

The following theorem provides second-order necessary optimality conditions.

Theorem 2.3.4 (second-order necessary condition for Pareto optimality)**.** Let the objective and the constraint functions of the problem shown in (2.3.1) be twice continuously differentiable at a regular decision vector $\boldsymbol{x}^* \in \Omega$. A necessary condition for $\boldsymbol{x}^*$ to be Pareto optimal is that there exist vectors $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$ (where $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\lambda} \neq \mathbf{0}$) such that:

1. $\sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla g_j(\boldsymbol{x}^*) = 0$
2. $\mu_j g_j(\boldsymbol{x}^*) = 0, \ \forall j \in \{1, \ldots, p\}$
3. $\boldsymbol{d}^T \left(\sum_{i=1}^{m} \lambda_i \nabla^2 f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla^2 g_j(\boldsymbol{x}^*) \right) \boldsymbol{d} \geq 0,$

$\forall \boldsymbol{d} \in \{\boldsymbol{d} \in \mathbb{R}^n \,|\, \boldsymbol{d} \neq \boldsymbol{0}, \forall i \in \{1, \ldots, m\} : \nabla f_i(\boldsymbol{x}^*)^T \boldsymbol{d} \leq 0, \forall j \in J(\boldsymbol{x}^*) : \nabla g_j(\boldsymbol{x}^*)^T \boldsymbol{d} = 0\}$.

For a proof, see Wang [185].

The following theorem gives second-order sufficient optimality conditions.

Theorem 2.3.5 (second-order sufficient condition for Pareto optimality)**.** Let the objective and the constraint functions of the problem shown in (2.3.1) be twice continuously differentiable at a decision vector $\boldsymbol{x}^* \in \Omega$. A sufficient condition for $\boldsymbol{x}^*$ to be Pareto optimal is that there exist vectors $\boldsymbol{\lambda} \geq \boldsymbol{0}$ and $\boldsymbol{\mu} \geq \boldsymbol{0}$ (where $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\lambda}, \boldsymbol{\mu} \neq \boldsymbol{0}$) such that:

1. $\sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla g_j(\boldsymbol{x}^*) = 0$
2. $\mu_j g_j(\boldsymbol{x}^*) = 0, \ \forall j \in \{1, \ldots, p\}$
3. $\boldsymbol{d}^T \left(\sum_{i=1}^{m} \lambda_i \nabla^2 f_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \mu_j \nabla^2 g_j(\boldsymbol{x}^*) \right) \boldsymbol{d} > 0,$

for either all $\boldsymbol{d} \in \{\boldsymbol{d} \in \mathbb{R}^n \,|\, \boldsymbol{d} \neq \boldsymbol{0}, \forall i \in \{1, \ldots, m\} : \nabla f_i(\boldsymbol{x}^*)^T \boldsymbol{d} \leq 0, \forall j \in J(\boldsymbol{x}^*) : \nabla g_j(\boldsymbol{x}^*)^T \boldsymbol{d} \leq 0\}$ or all $\boldsymbol{d} \in \{\boldsymbol{d} \in \mathbb{R}^n \,|\, \boldsymbol{d} \neq \boldsymbol{0}, \forall j \in J^+(\boldsymbol{x}^*) : \nabla g_j(\boldsymbol{x}^*)^T \boldsymbol{d} = 0, \forall j \in J(\boldsymbol{x}^*) \backslash J^+(\boldsymbol{x}^*) : \nabla g_j(\boldsymbol{x}^*)^T \boldsymbol{d} \leq 0\}$, where $J^+(\boldsymbol{x}^*) = \{j \in J(\boldsymbol{x}^*) \,|\, \mu_j > 0\}$.

For a proof, see Wang [185].

2.4 Summary

Many real-world optimization problems involve the simultaneous optimization of several conflicting objectives. These problems are called multiobjective optimization problems.

Since the objectives are in conflict, there is no single solution to the problems, but a set of compromise solutions representing the different trade-offs with respect to the given objective functions. This set is generally known as the set of Pareto optimal solutions.

The main goal in multiobjective optimization is to find a set of solutions that approximates the set of Pareto optimal solutions as well as possible. Since without further preference information none of these solutions can be said superior than other, it is important to find as many Pareto optimal solutions as possible. Therefore, there are two goals in multiobjective optimization: (i) to find a set of solutions as close as possible to the true Pareto optimal front, and (ii) to find a set of solutions as diverse as possible.

The first goal is identical to the goal of convergence in single-objective optimization. However, the second goal is specific to multiobjective optimization. Additionally, all well-distributed Pareto optimal solutions should cover the entire Pareto optimal region. A diverse set of solutions assures a good set of trade-off solutions. The concept of diversity can be defined either in the decision space or in the objective space. However, the diversity in one space does not guarantee the diversity in the other space.

Since both goals are important, a multiobjective optimization algorithm must satisfy both of them. When designing an efficient multiobjective algorithm, it should be realized that the achievement of one goal does not necessarily achieve the other goal. Therefore, explicit or implicit mechanisms of ensuring the convergence to the Pareto optimal region as well as the maintenance of a diverse set of solutions must be implemented in an algorithm. Due to these dual tasks, multiobjective optimization is more difficult than single-objective optimization.

Chapter 3

Multiobjective Optimization Algorithms

3.1 Introduction

The previous chapter presents the essential background necessary for dealing with multiobjective optimization. The present chapter discusses some of the most prominent approaches developed for solving multiobjective optimization problems. In general, there is a large variety of such methods, and all of these methods can be classified in according to different criteria.

A general and probably most commonly used way for categorizing the methods is by differentiating into the so-called classical methods and evolutionary algorithms. Such classification is mainly based on the working principles for finding Pareto optimal solutions. Additionally, according to the participation of the decision maker in the solution process, all methods can be classified as [133]:

- No-preference methods (no articulation of preference information is used),
- A posteriori methods (a posteriori articulation of preference information is used),
- A priori methods (a priori articulation of preference information is used),

- Interactive methods (progressive articulation of preference information is used).

As the name suggests, no-preference methods do not use any preference information and do not rely on any assumptions about the importance of objectives. These methods do not make an attempt to find multiple Pareto optimal solutions. Instead, the distance between some reference point and the feasible objective region is minimized to find a single optimal solution. A posteriori methods aim at finding multiple Pareto optimal solutions. After the Pareto optimal set has been generated, it is presented to the decision maker, who selects the most preferred solution among the alternatives. In the case of a priori methods, the decision maker specifies his preferences before the search process. Then one preferred solution or a subset of Pareto optimal solutions that satisfies these preferences is found and presented to the decision maker. In interactive methods the preference information is used progressively during the search process. The decision maker works together with a computer program to answer some questions or provide additional information after a certain number of iterations. The focus of the present thesis is on a posteriori methods.

Multiobjective evolutionary algorithms became very popular due to their ability to simultaneously deal with a set of solutions and to approximate the Pareto set in a single run. They can be also classified in various ways. For example, in the first book on evolutionary multiobjective optimization [42] they are classified into non-elitist and elitist approaches. On the other hand, MOEAs can be differentiated according to the principles of performing the search either in the decision or objective space. Concerning the objective space, the selection of promising solutions is made based on the fitness assigned to the population members. Therefore, one can categorize MOEAs according to the fitness assignment mechanisms as: dominance-based, scalarizing-based, and indicator-based. Dominance-based approaches calculate an individual's fitness on the basis of the Pareto dominance. Scalarizing-based approaches use traditional mathematical techniques based on the aggregation of multiple objectives into a single parameterized objective to assign a scalar fitness value to each individual in the population. In turn, indicator-based approaches, which are a relatively recent trend in EMO, employ performance indicators to

assign fitness to individuals in the current population.

Another way to classify MOEAs is based on how the search is performed in the decision space. In other words, the classification is based on how new candidate solutions are generated (i.e., based on the variation operators of EAs). This classification is adopted in this thesis to discuss different state-of-the-art MOEAs, except for MOEAs based on decomposition that are put in a distinct category. Furthermore, due to the growing attention to the field of many-objective optimization, evolutionary algorithms designed specifically to deal with many-objective problems are discussed separately. The fitness assignment and selection process are the main features under consideration in these algorithms. As a final remark, it should be noted that the herein presented classifications are not absolute, overlapping and combinations of different categories are possible. Some methods can be put in more than one category. All classifications are for guidance only.

3.2 Classical Methods

Classical methods have been studied in literature for nearly last six decades. Each of them has its own strengths and weaknesses. However, the proofs of convergence to the Pareto optimal set are their main strength. In the following, a few classical methods for multiobjective optimization are discussed.

3.2.1 Weighted Sum Method

The weighted sum method [75] associates each objective function with a weighting coefficient and minimizes the weighted sum of the objectives. In this way, the multiple objective functions are transformed into a single objective function. Thus, the original MOP results

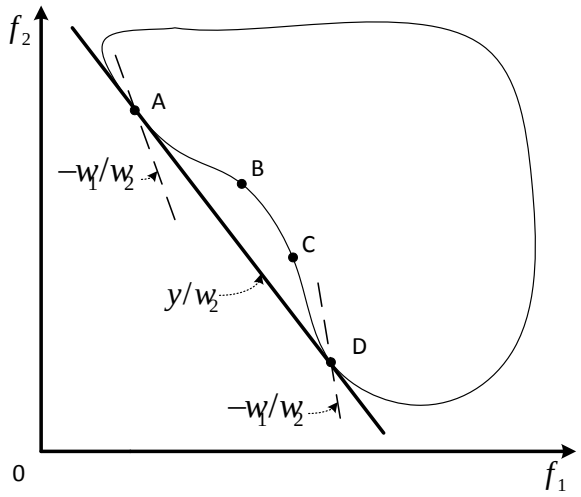


Figure 3.1: Representation of the weighted sum method.

in SOP by forming a linear combination of the objectives as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} w_i f_i(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.1}$$

where $\forall i \in \{1, \ldots, m\} : w_i \geq 0 \wedge \sum_{i=1}^{m} w_i = 1$.

The main disadvantage of this method is that it cannot find Pareto optimal solutions in nonconvex regions of the Pareto optimal front. For two objectives, this is illustrated in Figure 3.1. For fixed weights w_1, w_2, solution $\boldsymbol{x}$ is sought to minimize $y = w_1 \cdot f_1(\boldsymbol{x}) + w_2 \cdot f_2(\boldsymbol{x})$. This equation can be reformulated as $f_2(\boldsymbol{x}) = -\frac{w_1}{w_2} \cdot f_1(\boldsymbol{x}) + \frac{y}{w_2}$, which defines a line with slope $-\frac{w_1}{w_2}$ and intercept $\frac{y}{w_2}$ in objective space (solid line in Figure 3.1). Graphically, the optimization process corresponds to moving this line downwards until no feasible objective vector is below it and at least one feasible objective vector (here A and D) is on it. However, the points B and C will never minimize y . If the slope is decreased, D achieves a lower value of y than B and D (lower dotted line); if the slope is increased, A has a lower value of y (upper dotted line).

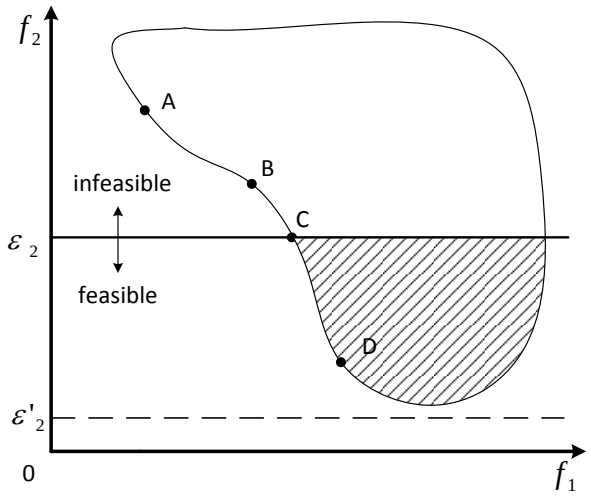


Figure 3.2: Representation of the ϵ-constraint method.

3.2.2 ϵ-Constraint Method

In the ϵ-constraint method, introduced in Haimes et al. [78], one of the objective functions is selected to be optimized and all the other objective functions are converted into constraints by setting an upper bound to each of them. The problem to be solved is now of the form:

$$\begin{aligned} \text{minimize} \quad & f_l(\boldsymbol{x}) \\ \text{subject to} \quad & f_i(\boldsymbol{x}) \leq \epsilon_i, \quad \forall i \in \{1, \ldots, m\} \wedge i \neq l \\ & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.2}$$

In the above formulation, the parameter ϵ_i represents an upper bound of the value of f_i. The working principle of the ϵ-constraint method for two objectives is shown in Figure 3.2. The function $f_1(\boldsymbol{x})$ is retained and optimized whereas $f_2(\boldsymbol{x})$ is treated as a constraint: $f_2(\boldsymbol{x}) \leq \epsilon_2$. The optimal solution of this problem is the point C. It can be easily seen that the method is able to obtain solutions in convex as well as nonconvex regions of the Pareto optimal front. However, if the upper bounds are not chosen appropriately (ϵ'_2), the obtained new feasible set might be empty, i.e., there is no solution to the corresponding

SOP. In order to avoid this situation, a suitable range of values for the ϵ_i has to been known beforehand.

3.2.3 Weighted Metric Methods

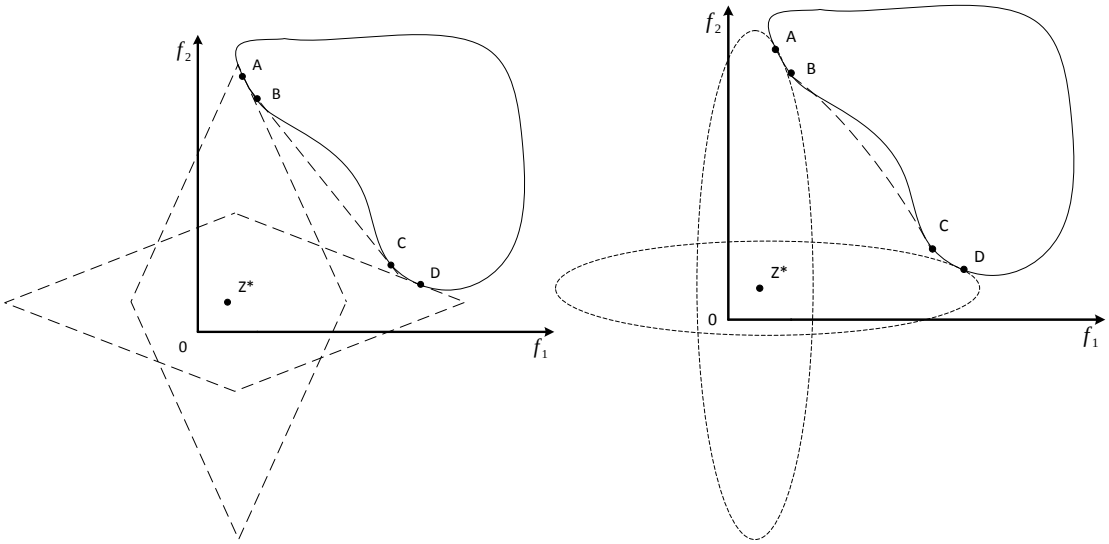
In the weighted metric methods, the distance between some reference point and the feasible objective region is minimized. For this purpose, the weighted L_p metrics are used for measuring the distance of any solution from the reference point. The ideal objective vector is often used as the reference point. Thus, the weighted L_p-problem is of the form:

$$\begin{aligned} \text{minimize} \quad & \left(\sum_{i=1}^{m} w_i |f_i(\boldsymbol{x}) - z_i^*|^p \right)^{1/p} \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.3}$$

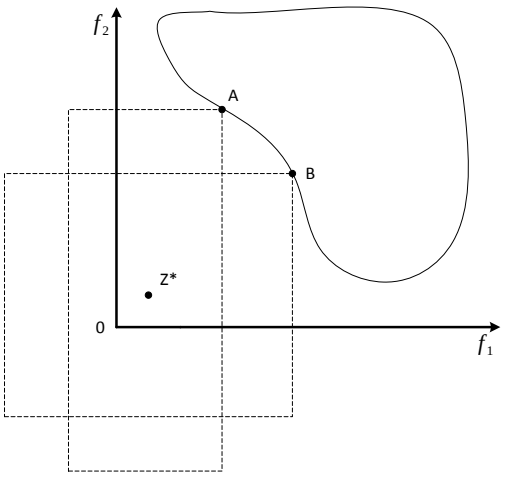
The parameter p can take any value between and 1 and ∞. When $p = 1$ is used, the resulting problem is equivalent to the weighted sum approach (if $z^* = 0$). When $p = 2$ is used, a weighted Euclidean distance of any point in the objective space from the ideal point is minimized. When p gets larger, the minimization of the largest deviation becomes more and more important. When $p = \infty$, the only thing that matters is the weighted relative deviation of one objective function, and the above problem reduces to the following problem:

$$\begin{aligned} \text{minimize:} \quad & \max_{1 \leq i \leq m} \quad w_i |f_i(\boldsymbol{x}) - z_i^*| \\ \text{subject to:} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.4}$$

Problem shown in (3.2.4) was originally introduced in Bowman [106], and it is called the weighted Chebyshev problem. In literature, the name of the above problem may vary due to the different ways of spelling.



(a) The weighted metric method with $p = 1$ (b) The weighted metric method with $p = 2$



(c) The weighted metric method with $p = \infty$

Figure 3.3: Representation of the weighted metric method.

However, the resulting optimal solution obtained by the chosen L_p depends on the parameter p. The working principle of this method for two objectives is illustrated in Figures 3.3(a), 3.3(b) and 3.3(c) for $p = 1, 2$ and ∞, respectively.

In all these figures, optimal solutions for two different weight vectors are shown. It is clear that with $p = 1$ or 2, not all Pareto optimal solutions can be obtained. In these cases, the figures show that no solution in the region BC can be found by using $p = 1$ or 2. However, when the weighted Chebyshev metric is used (Figure 3.3(c)), any Pareto optimal solution can be found. The theorem which states that every Pareto optimal solution can be obtained by the weighted Chebyshev method and its prove can be found in Miettinen [133].

It is important to note that even differentiable multiobjective problem becomes nondifferentiable using the weighted Chebyshev method. On the other hand, for the low values of p if the original MOP is differentiable the resulting SOP remains also differentiable and it can be solved using gradient-based methods. Another difficulty may arise from dealing with the objectives of different orders of magnitude. In this case, it is advisable to normalize the objective functions. In turn, this requires a knowledge of minimum and maximum function values of each objective. Moreover, this method also requires the ideal solution $\boldsymbol{z}^*$. Therefore, all m objectives need to be independently optimized before optimizing the L_p metric.

3.2.4 Normal Boundary Intersection Method

Das and Denis [39] proposed the normal boundary intersection (NBI) method, which attempts to find multiple Pareto optimal solutions of a given multiobjective problem by converting it into a number of single objective constrained problems. In the NBI method, it is assumed that the vector of global minima of the objectives ($\boldsymbol{f}^*$) is available. The convex hull is obtained using the individual minimum of the functions. Then, the simplex is constructed by the convex hull of the individual minimum and is expressed as $\boldsymbol{\Phi\beta}$. The NBI scalarization scheme takes a point on the simplex and then searches for the maximum distance along the normal pointing toward the origin. This obtained point may or may

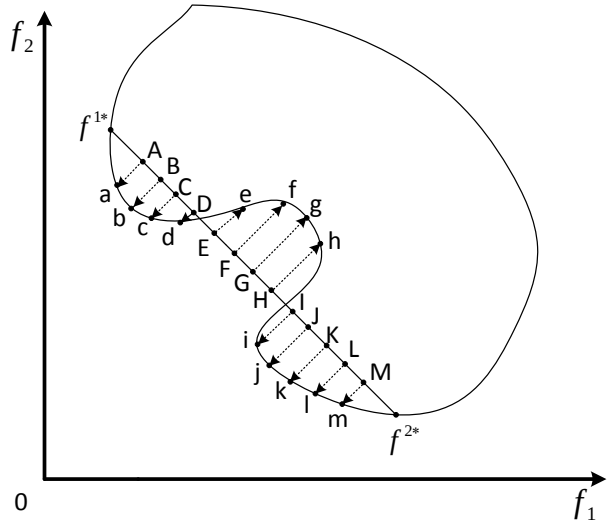


Figure 3.4: Representation of the normal boundary intersection method.

not be a Pareto optimal point. The NBI subproblem for a given vector $\boldsymbol{\beta}$ is of the form:

$$\begin{aligned} \text{maximize :} \quad & t \\ \text{subject to} \quad & \boldsymbol{\Phi\beta} + t\hat{\boldsymbol{n}} = \boldsymbol{f}(\boldsymbol{x}) \\ & \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.5}$$

where $\forall i \in \{1, \ldots, m\} : \beta_i \geq 0 \wedge \sum_{i=1}^{m} \beta_i = 1$, $\boldsymbol{\Phi} = [\boldsymbol{f}(\boldsymbol{x}^{1*}), \boldsymbol{f}(\boldsymbol{x}^{2*}), \ldots, \boldsymbol{f}(\boldsymbol{x}^{m*})]$ is a $m \times m$ matrix, $\boldsymbol{x}^{i*}$ is the minimizer of the i-th individual objective function ($\boldsymbol{x}^{i*} = \arg\min_{\boldsymbol{x}} f_i(\boldsymbol{x})$), $\hat{\boldsymbol{n}}$ is the normal direction at the point $\boldsymbol{\Phi\beta}$ pointing towards the origin. The solution of the above problem gives the maximum t and also the corresponding approximation to the Pareto optimal solution $\boldsymbol{x}$. The method works even when the normal direction is not an exact one, but a quasi-normal direction. The following quasi-normal direction vector is suggested in Das and Denis [39]: $\hat{\boldsymbol{n}} = -\boldsymbol{\Phi e}$, where $\boldsymbol{e} = (1, 1, \ldots, 1)^{\mathrm{T}}$ is a $m \times 1$ vector. The above quasi-normal direction has the property that NBI_β is independent of the relative scales of the objective functions.

Figure 3.4 illustrates the working principle of the NBI method. It shows the solutions $\{a, b, c, d, e, f, g, h, i, j, k, l, m\}$ obtained from the points $\{A, B, C, D, E, F, G, H, I, J, K, L, M\}$

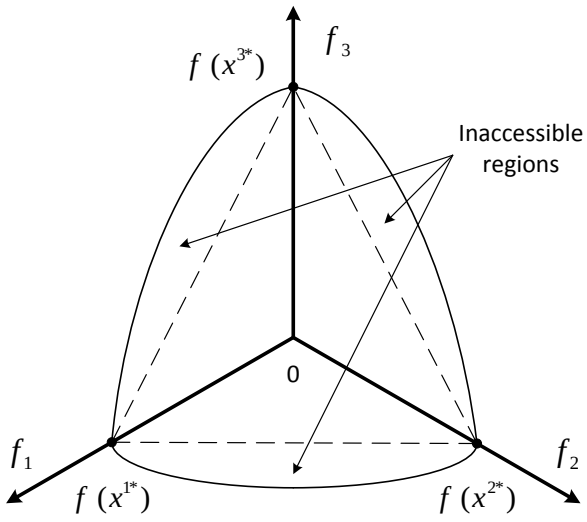


Figure 3.5: Pareto optimal solutions not obtainable using the NBI method.

on the convex hull ($\mathbf{\Phi} = [\boldsymbol{f}^{1*}\ \boldsymbol{f}^{2*}]$) by solving the NBI subproblems for the different vectors $\boldsymbol{\beta}$. The method can find solutions in convex and nonconvex regions of the Pareto front. It can be seen that points $\{f, g, h\}$ are not Pareto optimal but are still found using the NBI method.

Besides sometimes non-Pareto optimal solutions are obtained, another limitation of the NBI method is that for dimensions more than two, the extreme Pareto optimal solutions are not obtainable in all the cases. This limitation is due to the restriction of $\mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1}$ parameters. This can be easily seen by considering a problem having a spherical Pareto front satisfying $f_1^2 + f_2^2 + f_3^2$ in the range $\forall i \in \{1, \ldots, m\}$: $0 \leq f_i \leq 1$. As seen from Figure 3.5, there are unexplored regions outside the simplex obtained by the convex hull of individual function minima.

3.2.5 Normal Constraint Method

Messac and Mattson [130] proposed the normal constraint (NC) method for generating a set of evenly spaced Pareto optimal solutions. To describe the NC method, the following notations are introduced first. The NC method uses the normalized function values to cope with disparate function scales. The normalized objective vectors ($\overline{\boldsymbol{f}} =$

$(\overline{f}_1, \ldots, \overline{f}_m)^{\mathrm{T}}$) are computed using the ideal ($\boldsymbol{f}^{\mathrm{ideal}} = (f_1^{\mathrm{ideal}}, \ldots, f_m^{\mathrm{ideal}})^{\mathrm{T}}$) and the nadir ($\boldsymbol{f}^{\mathrm{nadir}} = (f_1^{\mathrm{nadir}}, \ldots, f_m^{\mathrm{nadir}})^{\mathrm{T}}$) objective vectors. The following equation is used to perform the mapping:

$$\overline{f}_i = \frac{f_i - f_i^{\mathrm{ideal}}}{f_i^{\mathrm{nadir}} - f_i^{\mathrm{ideal}}}, \quad \forall i \in \{1, \ldots, m\}.$$

The points $\boldsymbol{\mu}^{i*} = \boldsymbol{f}(\boldsymbol{x}^{i*})$ $\forall i \in \{1, \ldots, m\}$ are called the anchor points, where $\boldsymbol{x}^{i*}$ is the minimizer of the i-th objective. The normalized anchor points are denoted as $\overline{\boldsymbol{\mu}}^{i*}$ $\forall i \in \{1, \ldots, m\}$. The m-dimensional hyperplane passing through the anchor points is constructed. A set of evenly distributed point is generated on the hyperplane. Any point on the hyperplane can be defined as a function of the anchor points:

$$\boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{\beta}$$

where $\boldsymbol{z}$ is a point on the hyperplane corresponding to a given vector $\boldsymbol{\beta}$ which satisfies

$$\forall i \in \{1, \ldots, m\} : 0 \leq \beta_i \leq 1 \wedge \sum_{i=1}^{m} \beta_i = 1,$$

and $\boldsymbol{\Phi} = (\overline{\boldsymbol{\mu}}^{1*}, \ldots, \overline{\boldsymbol{\mu}}^{m*}]$ is a $m \times m$ matrix.

The vectors to fixed anchor point $\overline{\boldsymbol{\mu}}^{l*}$ from other anchor points are computed:

$$\overline{\boldsymbol{v}}_i = \overline{\boldsymbol{\mu}}^{l*} - \overline{\boldsymbol{\mu}}^{i*}, \quad \forall i \in \{1, \ldots, m\} \wedge i \neq l.$$

Finally, the NC subproblem for a given point $\boldsymbol{z}$ on the constructed hyperplane is of the form:

$$\begin{aligned} \text{minimize} \quad & \overline{f}_l(\boldsymbol{x}) \\ \text{subject to} \quad & \overline{\boldsymbol{v}}_i^{\mathrm{T}}(\overline{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{z}) \leq 0, \quad \forall i \in \{1, \ldots, m\} \wedge i \neq l \\ & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{3.2.6}$$

The normal constraint method uses an inequality constraint reduction of the feasible space. Figure 3.6 shows the obtained solution using the NC method for two objectives. The hatched part is the feasible region corresponding to point B in the ideal plane. It can

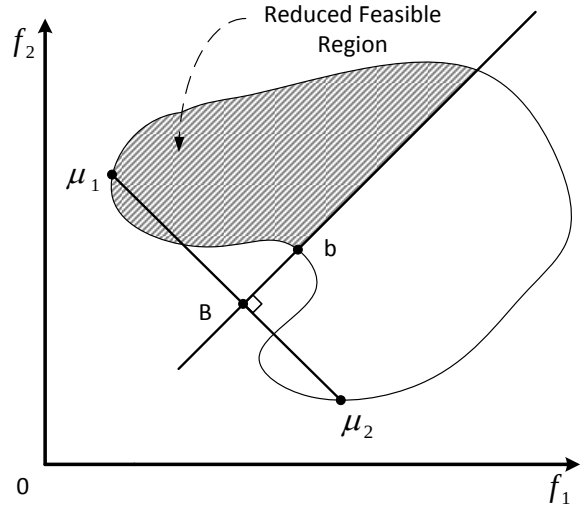


Figure 3.6: Representation of the NC method.

be seen that point b is not Pareto optimal but it is still found using the NC method. Since evenly distributed points on the ideal plane are used, the final points on the Pareto front are likely to be more evenly distributed than using the usual ϵ-constraint method.

3.2.6 Timmel's Method

Timmel [174] proposed a stochastic approach for finding multiple Pareto optimal solutions of a differentiable multiobjective optimization problem. It is a population based approach, where for a given parent solution, a child solution is created in the following manner:

$$\boldsymbol{x}^{\text{child}} = \boldsymbol{x}^{\text{parent}} - t_k \sum_{i=1}^{m} u_i \nabla f_i(\boldsymbol{x}^{\text{parent}}) \tag{3.2.7}$$

where $\boldsymbol{x}^{\text{parent}}$ is the parent decision vector, $\boldsymbol{x}^{\text{child}}$ is the generated child decision vector, $\nabla f_i(\boldsymbol{x}^{\text{parent}})$ is the gradient of the i-th objective function, t_k is the step size at the k-th iteration and u_i is a uniformly distributed random number between zero and one ($u_i \sim \mathbb{U}(0,1)$).

The above formulation ensures that not all objective functions can be worsened simultaneously. Thus, the child solution $\boldsymbol{x}^{\text{child}}$ is either nondominated when compared to the

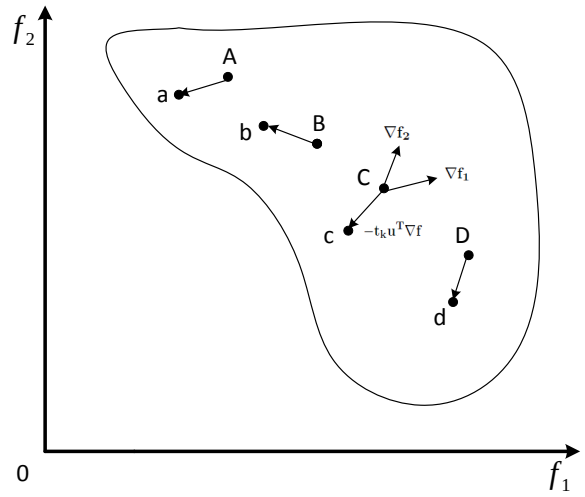


Figure 3.7: Representation of Timmel's method.

parent solution $\boldsymbol{x}^{\text{parent}}$, or it dominates the parent solution. The variation of the step size t over iterations must be made carefully to ensure convergence to the Pareto optimal front. In general, the step size update scheme must ensure the following aspects: (i) the step size should slowly decrease to zero as solutions closer to the Pareto optimal set are found; (ii) the decrease of the step size must not be slow enough so that the algorithm gets caught in suboptimal regions.

Figure 3.7 shows the creation of child solutions $\{a, b, c, d\}$ (denoted by the lowercase letters) from the corresponding parents (denoted by the capital letters). The vectors of the gradients are presented for the parent solution C. The child solution c is generated by adding the linear combination of the gradients to the parent solution C. For the other parent solutions children are generated in the same way. It can be seen that not all children dominate their parents. After the child population is created, it is combined with the parent population and only nondominated solutions are retained. Then, this set becomes the parent population, and this procedure is repeated for a specified number of iterations. The population size can vary with iterations, if fact, an increase in the population size is expected in most of the problems. It is interesting to note that this algorithm uses an elitist strategy, in which the best individuals from parents and offspring are retained.

3.3 Evolutionary Multiobjective Optimization Algorithms

The term evolutionary algorithm stands for a class of stochastic optimization methods that simulate the process of natural evolution [199]. The origins of EAs can be traced back to the late 1950s, however, the field remained relatively unknown to the broader scientific community for almost three decades. This was largely due to the lack of available powerful computer platforms at that time, but also due to some methodological shortcomings of those early approaches. Since the 1970s several evolutionary methodologies have been proposed, mainly genetic algorithms, evolutionary programming, and evolution strategies [4].

Evolutionary algorithms are particularly suitable for solving multiobjective optimization problems because they deal simultaneously with a set of possible solutions, called population, which allows to find several Pareto optimal solutions in a single run of the algorithm, instead of having to perform multiple runs with different parameter settings as in the case of the majority of classical methods for multiobjective optimization.

In the following, a number of state-of-the-art EMO algorithms are discussed. All algorithms are structured according to variation operators, except for MOEAs based on decomposition, which are put in a distinct category. A number of other approaches, such as indicator-based and scalarizing-based algorithms as well as algorithms designed specifically to handle many-objective optimization problems are discussed in the next section.

3.3.1 Genetic Algorithm-Based Approaches

The concept of a genetic algorithm (GA) was first introduced by Holland in the early 1970s at the University of Michigan. As the name suggests, genetic algorithm is a probabilistic search algorithm which borrows its working principal from natural selection and natural genetics. A comprehensive description of GAs can be found in [77, 85, 132, 134].

The first actual implementation of a multiobjective evolutionary algorithm was suggested by Schaffer [156]. The proposed vector evaluation genetic algorithm (VEGA) ba-

sically consisted of a simple genetic algorithm with a modified selection mechanism. The population at any generation is divided into N/m subpopulations, where m is the number of objectives and N is the population size. To each individual in each subpopulation is assigned a fitness based on the corresponding particular objective function only. In order to reduce the positional bias in the population, the population is shuffled before it is partitioned into subpopulations. After each solution is assigned a fitness, the proportionate selection operator [77] is applied until the complete subpopulation is filled. The described selection procedure emphasizes solutions which are good for individual objective functions. In order to find intermediate trade-off solutions, crossover is allowed between any two solutions in the entire population. In this way, the main idea is that a crossover between two good solutions, each corresponding to a different objective, may find offspring which are good compromised solutions between the two objectives.

The idea of calculating the fitness for individuals in the population based on the concept of Pareto optimality was for the first time suggested by Goldberg [77]. In turn, Fonseca and Fleming [70, 72] first proposed a multiobjective genetic algorithm (MOGA) which explicitly emphasizes nondominated solutions and simultaneously maintains diversity in the nondominated solutions. The proposed approach differs from a standard GA in the way fitness is assigned to each solution in the population. The rest of the algorithm (the stochastic universal selection, a single point crossover, and a bit-wise mutation) is the same as that in a classical GA. First, each solution is checked for its domination in the population. To a solution i, a rank equal to one plus the number of solutions n_i that dominate solution i is assigned: $r_i = 1 + n_i$. In this way, nondominated solutions are assigned a rank equal to 1, since no solution would dominate a nondominated solution in a population. Therefore, there must be at least one solution with rank equal to one and the maximum rank of any population member cannot be more than N (the population size). In order to maintain diversity among nondominated solutions, the authors have introduced niching among solutions of each rank. A shared fitness value is calculated by dividing the fitness of a solution by its niche count. In order to keep the average fitness of the solutions in a rank the same as that before sharing, these fitness values are scaled

so that their average shared fitness value is the same as the average assigned fitness value. After these calculations, the focus is shifted to the solutions of the next rank and an identical procedure is executed. This procedure is continued until all ranks are processed. Thereafter, the stochastic universal selection based on shared fitness values, the single point crossover, and the bit-wise mutation operators are applied to create a new population.

Goldberg's idea of using the nondominated sorting concept in GAs was more directly implemented by Srinivas and Deb [164]. The proposed approach, called nondominated sorting genetic algorithm (NSGA), uses a fitness assignment scheme which prefers nondominated solutions and employs a sharing strategy which preserves diversity among solutions of each nondominated front. The first step in NSGA is to sort the population P according to non-domination. The fitness assignment procedure begins from the first nondominated set and successively proceeds to dominated sets. Any solution i of the first nondominated set is assigned a fitness equal to $F_i = N$ (the population size). In this way, assigning more fitness to solutions belonging to a better nondominated set ensures a selection pressure towards the Pareto optimal front. In order to promote the diversity among solutions the sharing function method is used which degrades the assigned fitness based on the number of neighboring solutions. That is, for each solution i in the front P_j, the normalized Euclidean distance d_{ij} (in the decision space) from another solution j in the same front is calculated. Once these distances are calculated, they are used to compute a sharing function. The sharing function takes a value between zero and one, depending on the distance d_{ij} . Any solution j which has a distance greater than σ_{share} from the i-th solution contributes nothing to the sharing function value. After all $|P_j|$ sharing function values are calculated, they are added together to calculate the niche count nc_i, of the i-th solution. The niche count, in some sense, denotes the number of solutions in the neighborhood of the i-th solution, including the latter. If there exists no other solution within a radius of σ_{share} from a solution the niche count for that solution would be one. On the other hand, if $|P_j|$ solutions in the front are very close to each other compared to σ_{share}, the niche count of any solution in the group would be close to $|P_j|$. When the niche count is calculated, the fitness of the i-th solution is reduced as $F'_i = F_i/nc_i$. This process of degrading fitness of

a solution, which is crowded by many solutions, helps to emphasize the solutions residing in less crowded regions. This procedure completes the fitness assignment procedure of all solutions in the first front. Thereafter, a fitness value slightly smaller than the minimum shared fitness in the first front is assigned to solutions in the second nondominated front. This makes sure that no solution in the first front has a shared fitness worse than the assigned fitness of any solution in the second front. Once again, the sharing function method is applied among the solutions of the second front and the corresponding shared fitness values are computed. Next this procedure is applied to subsequent fronts until all solutions in the population are assigned a shared fitness. Finally, the assigned shared fitness is used to select the mating pool and genetic operators are applied to produce a new population.

Deb et al. [46] suggested an elitist nondominated sorting genetic algorithm (NSGA-II). In NSGA-II, each solution in the current population P is evaluated using Pareto ranking and a crowding measure. First the best rank is assigned to all the nondominated solutions in the current population. Solutions with the best rank are removed from the current population. Next, the second best rank is assigned to all the nondominated solutions in the remaining population. In this manner, ranks are assigned to all solutions in the current population. A fast nondominated sorting procedure was also proposed. Then, a binary tournament selection based on non-domination rank and crowding distance is performed to select a set of parent solutions. That is, when two solutions are selected, the one with the lower non-domination rank is preferred. Otherwise, if both solutions belong to the same rank, then the solution with the higher crowding distance is selected. Next, genetic operators such as recombination and mutation are applied to create an offspring population Q. Then, the two populations are merged together to form a combined population $R_t = P_t \cup Q_t$. Next, the combined population is sorted according to different nondominated front. If the size of the first nondominated front $\mathcal{F}_1$ is smaller then N, all members of the set $\mathcal{F}_1$ are chosen for the new population. The remaining members of the population P_{t+1} are chosen from subsequent nondominated fronts in the order of their ranking. Thus, solutions from the set $\mathcal{F}_2$ are chosen next, followed by solutions from the set $\mathcal{F}_3$, and so on. This procedure is continued until no more sets can be accommodated. Say that the set $\mathcal{F}_l$

is the last nondominated set beyond which no other set can be accommodated. In general, the count of solutions in all sets from $\mathcal{F}_1$ to $\mathcal{F}_l$ would be larger than the population size. To choose exactly N population members, the solutions of the last nondominated front $\mathcal{F}_l$ are sorted using the crowding distance in descending order and choose the best solutions needed to fill all population slots. Then the algorithm proceeds to the next iteration and the new population P_{t+1} is used for selection, crossover, and mutation to create a new population Q_{t+1}.

Zitzler and Thiele [195, 201] proposed a strength Pareto evolutionary algorithm (SPEA). SPEA uses a regular population and an archive to store nondominated solutions. The algorithm starts by randomly creating an initial population P_0 of size N and an empty archive A_0 with a maximum capacity of $\bar{N}$. In any generation t, the nondominated solutions of the population P_t are copied to the archive P_t, while any dominated individuals or duplicates (regarding the objective values) are removed from the archive during this update procedure. If the size of the updated archive exceeds a predefined limit, further archive members are deleted by a clustering technique. The fitness values are assigned to both archive and population members. Each individual i in the archive is assigned a strength value $S(i)$, which at the same time represents its fitness value $F(i)$. $S(i)$ is the number of population members j that are dominated by or equal to i, divided by the population size plus one. The fitness $F(j)$ of an individual j in the population is calculated by summing the strength values $S(i)$ of all archive members i that dominate or are equal to j, and adding one at the end. Next, the mating selection is performed where individuals from the union of population and archive are selected by means of binary tournaments. Here, minimum fitness corresponds to better performance, so each individual in the archive has a higher chance to be selected than any population member. After a pool of parents is selected, recombination and mutation are applied to generate an offspring population which replaces the old population. Thereafter, the external archive is updated.

To overcome some drawbacks encountered in SPEA, Zitzler et al. [200] suggested an improved version of strength Pareto evolutionary algorithm, called SPEA2. The algorithm starts by randomly generating an initial population P_0 of size N and an empty archive A_0 of

size $\bar{N}$. Next, the fitness values are calculated for all solutions in the population and archive. Each individual i in the archive $\bar{P}_t$ and the population P_t is assigned a strength value $S(i)$, representing the number of solutions it dominates. Then the raw fitness $R(i)$ of individual i is calculated by summing the strengths of its dominators. Next, the density $D(i)$ is calculated as $D(i) = 1/(\sigma_i^k + 2)$, where σ_i^k is the distance to the k-th nearest solution and k is determined as the square root of the sum of the population and archive size. Thereafter, the fitness of individual i is calculated by adding density $D(i)$ to the raw fitness value $R(i)$. Mating selection procedure performs binary tournament selection with replacement on the archive A_{t+1} in order to fill the mating pool. Then, recombination and mutation operators are applied to the mating pool, the resulting set of solutions forms a new population P_{t+1}. During environmental selection, all nondominated individuals from archive and population are copied to the archive of the next generation $\bar{P}_{t+1}$. If the nondominated front fits exactly into the archive the environmental selection step is completed. In the case, when the archive is larger than the number of nondominated solutions, the remaining slots are filled with the best $\bar{N} - |\bar{P}_{t+1}|$ dominated individuals from the previous archive and population. In the case, when the size of the current nondominated set exceeds $\bar{N}$, an archive truncation procedure is invoked which iteratively removes individuals from $\bar{P}_{t+1}$ until $\bar{P}_{t+1} = \bar{N}$. At each stage, the truncation procedure chooses the individual which has the minimum distance to another individual. If there are several individuals with minimum distance, the tie is broken by considering the second smallest distance and so forth.

3.3.2 Evolution Strategy-Based Approaches

Evolution strategies (ESs) are a class of stochastic search methods inspired by the theory of evolution by means of natural selection. Its roots date back to the mid 1960s when ES was developed by three students (Bienert, Rechenberg, Schwefel) at the Technical University in Berlin. ESs use mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions. A review of ESs for single-objective optimization can be found in [14, 15, 159].

Knowles and Corne [112, 113] suggested a Pareto-archived evolution strategy (PAES). The algorithm is a simple (1+1)-ES extended to multiobjective optimization. At any generation, PAES maintains the parent a and the offspring b along with an archive of the best solutions found so far. At first, an initial random solution a is generated and added to the archive. It is then mutated by using a normally distributed probability function with zero-mean and with a fixed mutation strength. The produced offspring b is evaluated. Thereafter, both parent and offspring are compared and the winner becomes the parent of the next generation. First, the parent a and the offspring b are compared for domination. If a dominates b, the offspring b is discarded and a new mutated solution is created from a for further processing. On the other hand, if b dominates a, the offspring is better than the parent. In this case, solution b is accepted as a parent of the next generation and is added to the archive. Otherwise, when both a and b are nondominated to each other, the offspring is compared with the current archive, which contains the set of nondominated solutions found so far. Three cases are possible here. In the first case, the offspring is dominated by a member of the archive. The offspring is then rejected and the parent a is mutated again to find a new offspring for further processing. In the second case, the offspring dominates a member of the archive. The dominated members of the archive are deleted and the offspring is accepted without any condition. The offspring then becomes the parent of the next generation. In the third case, the offspring is not dominated by any member of the archive and it also does not dominate any member of the archive. In such a case, the offspring is added to the archive only if there is a slot available in the latter. However, the acceptance of the offspring in the achieve does not qualify it to become the parent of the next generation. To decide who qualifies as a parent of the next generation, the density of solutions in their neighborhood is checked. The one residing in the least crowded area in the search space qualifies as the parent. If the archive is full, the above density-based comparison is performed between the parent and the offspring to decide who remains in the archive. The adaptive grid procedure based on recursively dividing up the d-dimensional objective space is used to estimate the density of solutions in the archive. First, each objective is divided into 2^d equal divisions, where d is a user-defined depth parameter. In

this way, the entire search space is divided into $(2^d)^m$ unique, equal-sized m-dimensional hypercubes. The archived solutions are placed in these hypercubes according to their locations in the objective space. Thereafter, the number of solutions in each hypercube is counted. If the offspring resides in a less crowded hypercube than the parent, the offspring becomes the parent of the next generation. Otherwise, the parent solution continues to be the parent of the next generation. If the archive is already full with nondominated solutions, the offspring cannot be included automatically. First, the hypercube with the highest number of solutions is identified. If the offspring does not belong to that hypercube, it is included in the archive and at random one of the solutions from the highest-count hypercube is eliminated. Whether the offspring or the parent qualifies to be the parent of the next generation is decided by the same parent-offspring density count.

Costa and Oliveira [34] proposed a multiobjective elitist evolution strategy (MEES). It is an extension of the traditional single-objective ES to multiobjective optimization. MEES maintains the main features of ESs such as real representation of the search parameters, self-adaptation of step sizes and recombination. Moreover, an adaptive sharing scheme was proposed together with a geometric selection to control and guide the search. At each generation, the algorithm maintains the main population and the secondary population (SP). The main population is used to generate $(\lambda + \theta)$ offspring solutions. Offspring are created by means of recombination and mutation common to single-objective ES, the step size for mutation is adapted with nonisotropic self-adaptation scheme. The parameter θ is introduced in order to control the elitist level. This parameter states the maximum number of nondominated solutions of the secondary population that are introduced in the main population. If the number of nondominated solutions in SP is greater or equal then θ, then θ nondominated solutions are randomly selected from SP to constitute the elite, otherwise all nondominated solutions are selected from SP. MEES uses the $(\mu+\lambda)$ selection scheme. Fitness assignment is performed in two steps. First, the population of $(\mu + \lambda)$ solutions is sorted according to non-domination and ranks are assigned to each solution. Thereafter, in order to maintain diversity a sharing scheme is applied to the fitness values of solutions. The fitness value of each solution is multiplied by a quantity, called niche

count, proportional to the number of solutions having a distance inferior to a parameter σ_{share}. This fitness assignment procedure is repeated for all nondominated fronts separately until all solutions are assigned with fitness values. Two selection schemes were proposed in MEES. In the first scheme, a deterministic selection is performed in the case where the number of solutions in the first nondominated front is not greater then μ. Otherwise, a tournament selection is performed when the number of solutions in the first front is greater then μ. The second scheme is more complicated, and the main idea is to select distinct quantities of solutions from each nondominated front for the next generation. The quantities of solutions for each front are chosen in such a way that the number of solutions selected from the first front will be greater than the number of solutions selected from the second front and so on. In MEES, the secondary population is used to store nondominated solutions found during the search. During the generations, new nondominated solutions are added to SP whereas that became dominated are eliminated. To keep SP of bounded size, a parameter d is introduced stating the minimum desirable distance in objective space between solutions in SP.

3.3.3 Differential Evolution-Based Approaches

Differential evolution (DE), proposed by Storn and Price in 1995 [167], is a simple and powerful population-based stochastic direct search method for real-parameter optimization. DE uses a simple mutation operator based on differences between pairs of solutions with the aim of finding a search direction based on the distribution of solutions in the current population. Since its advent, DE has attracted the attention of the researchers from diverse domains of knowledge, all over the world. This has resulted in a plenty of variants [141, 165, 166] of the basic DE algorithm and a steadily growing number of published articles. A comprehensive review of single-objective DE-based approaches can be found in [40, 137].

Abbass et al. [1] proposed a Pareto differential evolution (PDE) algorithm. The algorithm, which uses the $DE/current\text{-}to\text{-}rand/1/bin$ variant for reproduction, works as follows. Only nondominated solutions are retained in the population for recombination

(all dominated solutions are removed). Three parents are randomly selected (one as the main parent and also trial solution) and a child is generated using these solutions. The offspring is placed in the population only if it dominates the main parent. Otherwise, a new selection process takes place. This process continues until the population is completed. If the number of nondominated solutions exceeds a certain threshold, a distance metric is adopted to remove parents, which are too close from each other. In this approach, the scaling parameter F is generated from a Gaussian distribution $N(0,1)$ and the boundary constraints are preserved either by reversing the sign if the variable is less than zero or by repetitively subtracting one if it is greater then zero, until the variable is within the allowable boundaries. The algorithm also incorporates a mutation operator, which is applied after the crossover with a certain probability, in order to add a small variation to each variable.

Iorio and Li [96] proposed a nondominated sorting differential evolution (NSDE). This approach is a simple modification of the NSGA-II. The only difference between this approach and NSGA-II is in the method for generating new individuals. In NSDE, simulated binary crossover (SBX) and polynomial mutation operators are replaced by the DE operators. New candidates are generated using the $DE/current\text{-}to\text{-}rand/1/bin$ variant, which is known to be rotationally invariant. A number of experiments are conducted on a unimodal rotated problem. Comparing the results produced by NSDE and NSGA-II, it was shown that DE can provide rotationally invariant behavior on a multiobjective optimization problem.

Santana-Quintero and Coello Coello [149] proposed ϵ-MyDE. This approach keeps two populations: the main population to select the parents and the secondary population, in which the concept of ϵ-dominance is adopted to retain and to uniformly distribute nondominated solutions found so far. The concept of ϵ-dominance does not allow two solutions with a difference less than ϵ_i in the i-th objective to be nondominated with respect to each other, thereby allowing a good spread of solutions. The algorithm incorporates a constraint-handling mechanism that allows infeasible solutions to intervene during recombination. The $DE/rand/1/bin$ variant is used for reproduction. After a user-defined

number of generations is reached, three solutions used in the mutation operator are selected from the secondary population in such a way that they are close among them in the objective space. If none of solutions satisfies this condition, other solution is randomly chosen from the secondary population. To improve exploration capabilities, a uniform mutation operator is used.

Kukkonen and Lampinen developed a generalized differential evolution (GDE). The proposal extends $DE/rand/1/bin$ to solve multiobjective optimization problems. The first version [119] of this approach modifies the original DE selection operation by introducing Pareto dominance as a selection criterion between the old population member and the trial vector. Also, Pareto dominance in the constraint space is considered to handle constrained problems.

To promote a better distribution of nondominated solutions, Kukkonen and Lampinen [115] suggested a second version of their approach, called GDE2. In this version, a crowding distance measure is used to select the best solution when the old population vector and the trial vector are feasible and nondominated with respect to each other, in such a way that the vector located in the less crowded region is selected to the next generation. The authors acknowledge that GDE2 is sensitive to its initial parameters and that the modified selection mechanism slows down the convergence.

To deal with the shortcomings of GDE2 regarding the slow convergence, Kukkonen and Lampinen [116] proposed an improved version of their approach, called GDE3. This version uses the nondominated sorting (as in NSGA-II), and allows the current population to grow in order to improve the distribution of solutions and to decrease the sensitivity of the approach to its initial parameters. In GDE3, when the old population member and the trial vector are feasible and nondominated with respect to each other, both of them are maintained. Hence, the population size grows. To maintain a fixed population size for the next generation, the nondominated sorting is performed after each generation to prune the population size.

A good review of the state-of-the-art multiobjective DE algorithms can be found in [131].

3.3.4 Particle Swarm Optimization-Based Approaches

Particle swarm optimization (PSO) was originally proposed by Kennedy and Eberhart in 1995 [107]. The PSO algorithm is based on swarm intelligence techniques. The concept of swarm intelligence was inspired by the social behavior of groups of animals such as a flock of birds, a nest of ants, or a school of fish. More about PSO algorithms can be found in [10, 11, 144].

Coello Coello et al. [30] proposed a particle swarm-based approach for multiobjective optimization (MOPSO). MOPSO uses an external repository (ER), in which every particle deposits its flight experiences after each flight cycle. After initialization of the particles and their velocities, the particles that represent nondominated vectors are stored in ER. Hypercubes of the search space explored so far are generated and the particles are located using these hypercubes as a coordinate system. Each particle's coordinates are defined according to the values of its objective functions. The memory of each particle is also stored in the repository. Then, for the predefined number of flight cycle particle swarm optimization is performed using its conventional concepts. The speed of each particle is calculated taken into account its best position and the nondominated particle taken from the repository. To choose the particle from the repository, each hypercube receives a fitness value based on the number of particles it contains. Then, roulette-wheel selection is applied using these fitness values to select the hypercube from which the corresponding particle will be taken. Once the hypercube has been selected, a particle within such hypercube is selected at random. The best position of each particle is also stored in the repository. When the current position of the particle is better than the position contained in its memory, the particle's position is updated. To decide what position from memory should be retained, the particle is compared based on Pareto dominance. If the current position is dominated by the position in memory, then the position in memory is kept. Otherwise, the current position replaces the one in memory. On the other hand, if neither of them is dominated by the other, one of them is selected randomly. After evaluating new positions of the particles, the external repository is updated. The update consists of inserting all

the currently nondominated locations into the repository. Any dominated locations from the repository are eliminated in the process. Whenever ER has reached its maximum allowable capacity, the adaptive grid procedure is invoked, which gives priority to the particles located in the less populated areas of objective space over those lying in highly populated regions. MOPSO also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved. At the beginning of the search, the mutation operator covers the full range of each design variable and then the range covered over time is narrowed using a nonlinear function.

Reyes Sierra and Coello Coello [145] proposed an improving version of MOPSO, called OMOPSO. This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions. The density estimator factor based on crowding distance is used to filter out the list of leaders whenever the maximum limit imposed on such list is exceeded. Only the leaders with the best density estimator values are retained. The concept of ϵ-dominance is used to select the particles that will remain in the archive of final solutions. Additionally, the authors propose a scheme in which they subdivide the swarm into three different subsets. A different mutation operator is applied to each subset. However, for all other purposes, a single swarm is considered.

Durillo et al. [61] studied the performance of six MOPSOs representative of the state-of-the-art and found that all of them face significant difficulties in solving some multi-frontal problems satisfactorily. The authors concluded that the velocity of the particles in these algorithms can become too high, resulting in erratic movements towards the upper and lower limits of the positions of the particles. In order to overcome these difficulties the same authors proposed speed-constrained multiobjective PSO [135], called SMPSO. They used OMOPSO as a starting point and incorporated a velocity constriction procedure. At each generation, after the velocity for each particle is calculated, the resulting velocity is multiplied by a constriction factor, and the resulting values are constrained by bounds calculated for each component of the velocity.

A comprehensive survey of multiobjective PSO proposals can be found in [146].

3.3.5 Scatter Search-Based Approaches

Scatter search (SS) was first introduced by Glover in 1977 as a heuristic for integer programming [76]. Scatter search derives its foundations from earlier strategies for combining decision rules and constraints, with the goal of enabling a solution procedure based on the combined elements to yield better solutions than one based only on the original elements. A comprehensive description about scatter search heuristic can be found in [118].

A multiobjective scatter search algorithm, called M-scatter search, was presented in [178]. As traditional scatter search algorithms, it works simultaneously with two sets of possible solutions: the *diversity* and the *reference* sets. The authors use a nondominated sorting procedure that ranks every solution of the reference set. The solutions from the n-th front receive a higher score than the ones from the $(n+1)$-th front. This score is, then, depreciated by a mechanism that penalizes each solution from a certain front based on the number of points from this front that are closer than a defined niched radius. The adopted procedure is similar to the one used in [164]. M-scatter search also uses an *offline* set that stores nondominated solutions found during the computation. To maintain nondominated solutions uniformly distributed along the Pareto front, the NSGA niching method is applied in the updating procedure of the *offline* set.

Nebro et al. [136] proposed a hybrid metaheuristic for multiobjective optimization, called AbYSS. The proposal adapts the scatter search template for single objective optimization to the multiobjective domain. Initially, the diversification generation method is invoked to generate s initial solutions. The method is based on dividing the range of each variable in a number of subranges of equal size. Then, each solution is created in two steps. Firstly, a subrange is randomly chosen, with the probability of selecting a subrange being inversely proportional to its frequency count, the number of times the subrange has been already selected. Secondly, a value is randomly generated within the selected range. Thereafter, each solution is passed to the improvement method. The idea behind this method is to use a local search algorithm to improve the new solutions. The improvement method takes an individual as parameter, which is repeatedly mutated with the aim of obtaining

a better individual. The term better is defined by checking whether two individuals are feasible or not. If one of them is feasible and the other one is not, or both are infeasible but one of them has a smaller overall constraint violation, the test returns the winner. Otherwise, a dominance test is used to decide whether one of the individuals dominates the other one. If the original individual wins, the mutated one is discarded. If the mutated individual wins, it replaces the original one. Finally, if they are both nondominated, the original individual is moved into the external archive and the mutated individual becomes the newly original one. The result obtained after the improvement method is the initial set P. After the initial phase, the scatter search main loop starts.

The main loop begins by building the reference set from the initial set with the reference set update method. The reference set is composed of two subsets, $RefSet_1$ and $RefSet_2$. The first subset contains the best quality solutions in P, while the second consists of those individuals from P whose minimum Euclidean distance to $RefSet1$ is the highest. Then, solutions in the reference set are systematically grouped in subsets of two or more individuals by means of the subset generation method. In AbYSS, the subset generation method produces, on the one hand, pairwise combinations of individuals from $RefSet_1$ and, on the other hand, pairwise combinations of individuals from $RefSet_2$. In the next step, the simulated binary crossover operator (SBX) is used in solution combination method to produce new individuals from the combined solutions. The improvement method is applied to each newly generated solution. Afterwards, the produced solutions are tested for inclusion into the reference set. A new solution can become a member of the reference set if it is better than another one in $RefSet_1$ or it has a better distance value to the reference set than the individual with the worst distance value in $RefSet_2$. To decide whether the individual is better than those in $RefSet_1$, a dominance test is used. When the new individual is not dominated by the $RefSet_1$, it is inserted into this set only if it is not full. This means that the new individual has to dominate at least one individual in $RefSet_1$. If this condition does not hold, the individual is checked for the insertion into the external archive. Then, there is a re-start phase, which consists of three steps. First, the individuals in $RefSet1$ are inserted into P. Second, the best n individuals from

the external archive, according to the crowding distance, are also moved to P. Third, the diversification generation and improvement methods are used to produce new solutions for filling up the set P. Then, if the stopping condition of the algorithm is not met, the algorithm proceeds to the next iteration.

When a new solution is found in the improvement or the solution combination methods, it is compared with the content of the archive, on a one-per-one basis. If this new solution is dominated by an individual from the archive, then such solution is discarded. Otherwise, the solution is stored. If there are solutions in the archive that are dominated by the new element, then such solutions are removed. If the archive reaches its maximum allowable capacity after adding the new solution, the crowding distance is used to decide which individual has to be removed.

3.3.6 Simulated Annealing-Based Approaches

Simulated annealing (SA) is a probabilistic metaheuristic first introduced by Kirkpatrick et al. in 1983 [109]. SA utilizes the principles of statistical mechanics regarding the behavior of a large number of atoms at low temperature. The notion of slow cooling of the material is implemented in the SA algorithm as a slow decrease in the probability of accepting worse solutions as it explores the solution space. SA was successfully applied to solve single-objective optimization problems in diverse areas [8, 26, 169].

The first multiobjective version of SA has been proposed by Serafini [160]. The algorithm is almost the same as the algorithm of single-objective SA. The method uses a modification of the acceptance criteria of solutions in the original algorithm. Various alternative criteria have been investigated in order to increase the probability of accepting nondominated solutions. A special rule given by the combination of several criteria has been proposed in order to concentrate the search almost exclusively on the nondominated solutions. Furthermore, the majority of early multiobjective SA proposals are based on combining different objectives into a single one by using weighted sum approach [82, 170], and then SA is used as a single-objective optimizer.

Smith et al. [163] suggested a multiobjective SA algorithm (MOSA), which uses a dominance-based energy function. In their approach, if the true Pareto front is available, then the energy of a particular solution x is calculated as the total number of solutions that dominates x. However, as the true Pareto front is not available all the time, a proposal has been made to estimate the energy-based on the current estimate of the Pareto front F', which is the set of mutually nondominated solutions found so far in the process. Then, the energy of the current solution x is the total number of solutions in the estimated front which dominates x. If $||F'_{x'}||$ is the energy of the new solution x' and $||F'_x||$ is the energy of the current solution x, then energy difference between the current and the proposed solution is calculated as $\delta E(x', x) = (||F'_{x'}|| - ||F'_x||)/||F'||$. Here, division by $||F'||$ is used to ensure that δE is always less than unity and provides some robustness against fluctuations in the number of solutions in F'. If the size of F' is less than some threshold, then attainment surface sampling method is adopted to increase the number of solutions in the final Pareto front. Authors have perturbed a decision variable with a random number generated from the Laplacian distribution. Two different sets of scaling factors, traversal scaling which generates moves to a nondominated proposal within a front, and location scaling which locates a front closer to the original front, are kept. These scaling factors are updated with the iterations.

Bandyopadhyay et al. [9] proposed an archived multiobjective simulated annealing algorithm (AMOSA). In AMOSA, the archive is used to store nondominated solutions found so far. Two limits are kept on the size of the archive: a hard or strict limit denoted by HL, and a soft limit denoted by SL. During the optimization process, nondominated solutions are stored in the archive until the size of the archive increases to SL. If the size of the archive exceeds the SL threshold, clustering is applied to reduce the size of the archive to HL. The main idea of allowing the archive size to increase up to SL is not only to reduce excessive calls to clustering, but also to enable the formation of more spread out clusters and hence better diversity. The algorithm begins with the initialization of a number of solutions. Each of these solutions is refined by using a simple hill-climbing technique, accepting a new solution only if it dominates the previous one. This is continued for a

number of iterations. Thereafter, the obtained nondominated solutions are stored in the archive. Then, one of the points, called *current-pt*, is randomly selected from the archive. Next, *current-pt* is perturbed to generate a new solution called *new-pt*. The domination status of *new-pt* is checked with respect to *current-pt* and solutions in the archive. Based on the domination status between *current-pt* and *new-pt*, three different cases may arise. In the first case, when the *current-pt* and k points from the archive dominate the *new-pt*, the *new-pt* is selected to be the *current-pt* with a probability inversely proportional to the average amount of domination of the *new-pt* by $(k+1)$ points. In the second case, when the *current-pt* and *new-pt* are nondominated with respect to each other: (i) the *new-pt* is selected as the *current-pt* with a probability inversely proportional to the average amount of domination of the *new-pt* by k points, if some k points from the archive dominate *new-pt*; (ii) the *new-pt* is selected as the *current-pt* and added to the archive, if there are no points in the archive that dominate the *new-pt*; (iii) the *new-pt* is selected as the *current-pt* and added to the archive, if the *new-pt* dominates some points in the archive, while the dominated points are removed from the archive. In the third case, when the *new-pt* dominates the *current-pt*, based on the domination status of the *new-pt* and the members of the archive, the following three situations are considered: (i) if k points in the archive dominate the *new-pt*, then the minimum of the difference of domination amounts between the *new-pt* and the points is computed and the point from the archive which corresponds to the minimum difference is selected as the *current-pt* with a probability depending on the computed minimum difference, otherwise the *new-pt* is selected as the *current-pt*; (ii) if there are no points in the archive which dominate the *new-pt*, the *new-pt* is accepted as the *current-pt* and stored in the archive, if the *current-pt* is in the archive, then it is removed; (iii) if the *new-pt* also dominates k points in the archive, the *new-pt* is selected as the *current-pt* and added to the archive, while all the dominated points of the archive are removed. The described process is repeated for a predefined number of iterations for each temperature. Temperature is reduced using the cooling rate of α till the minimum temperature $T_{\min}$ is attained. Thereafter, the optimization process stops and the archive returns a set of nondominated solutions.

Furthermore, a good review of several multiobjective SA algorithms and their comparative performance analysis can be found in [169].

3.3.7 Covariance Matrix Adaptation Evolution Strategy-Based Approaches

Covariance matrix adaptation evolution strategy (CMA-ES) is a popular descendant of the evolution strategy algorithm developed by Hansen and Ostermeier [81]. CMA-ES learns the dependencies between the variables by means of a covariance matrix build from successful steps that the algorithm has taken. New candidate solutions are sampled according to a multivariate normal distribution. The on-line adaptation of the covariance matrix makes CMA-ES invariant with respect to orthogonal transformations of the coordinate system.

Igel et al. [94] proposed the covariance matrix adaptation evolution strategy for multiobjective optimization (MO-CMA-ES). It is an extension of the powerful single-objective evolution strategy with covariance matrix adaptation [81] to the multiobjective case. The proposed approach is $\lambda_{\text{MO}} \times (1 + 1)$ elitist evolution strategy. At each generation, each solution in the current population generates one offspring. Thereafter, the sets of parents and offspring are combined and λ best individuals are retained for the next generation. Here, the selection procedure is similar to one adopted in NSGA-II. However, contributing hypervolume is used as the second sorting criterion. In the next step, the step size of parent and its offspring is updated depending on whether the mutation was successful. The proposed scheme considers an offspring to be successful if its Pareto rank is better than those of a parent. In the case, when the both get the same Pareto rank, the decision is made based on the contributing hypervolume. The covariance matrix of the offspring is updated taking into account the mutation that has led to its genotype. Both the step size and the covariance matrix update are the same as in the single-objective $(1 + \lambda)$-CMA-ES. Experimental results on the set of benchmark problems showed that the algorithm inherited the rotationally invariant properties. MO-CMA-ES outperformed NSGA-II and NSDE on the most of tested problems.

In [95, 171], an efficient mechanism to update the covariance matrix for MO-CMA-ES was investigated. The new update mechanism does not need to factorize the covariance matrix, in turn, it operates directly on Cholesky factor used to obtain a multivariate normal distribution with a given covariance. It was shown that the proposed method is easy to implement, considerably faster in large dimensions and provides a significant improvement for high dimensional optimization. Additionally, in [95] the authors proposed two steady-state selection schemes for MO-CMA-ES. The first variant is $(\lambda_{\prec} + 1)$ where the parent is selected among the nondominated solutions in the population. The second one is $(\lambda + 1)$ where the parent is selected uniformly at random from all solutions in the population. It was observed that the steady-state versions perform better than generational MO-CMA-ES on the unimodal problems. However, the generational version is superior on the multimodal problems.

Voß et al. [181] studied and compared the performance of steady-state MO-CMA-ES with different scalarization algorithms, in which the elitist CMA-ES is used as a single-objective optimizer. They considered the weighted sum and Chebyshev methods to transform a multiobjective optimization problem into single-objective one. Then, a number of obtained single-objective problems were solved by CMA-ES and results compared with MO-CMA-ES using the same number of function evaluations. The experimental results showed that MO-CMA-ES outperformed the elitist CMA-ES combined with scalarization on the most of benchmark functions, however, it was also concluded that when the scalarization produces perfect quadratic fitness functions CMA-ES could exploit the resulting structure and perform on par with MO-CMA-ES.

Loshchilov et al. [128] studied different schemes of the parent selection in steady-state MO-CMA-ES. Two selection procedures were considered. One is based on the tournament selection, while another one inspired from the multi-armed bandit framework. There were also defined four types of rewards received by the parents and offspring: (i) a reward based on the survival of the offspring, (ii) a smoother reward defined by taking into account the rank of the newly inserted offspring, (iii) a reward based on the hypervolume contribution of the offspring, and (iv) a relaxed reward based on the hypervolume contribution of the

offspring involving a rank-based penalization. Experiments on several biobjective problems have shown a significant speed-up of MO-CMA-ES on unimodal problems. However, the proposed approach results in a poor convergence on multimodal problems, or problems where some parts of the Pareto front are much easier to reach than others.

Voß et al. [182] proposed a new covariance matrix update scheme for MO-CMA-ES based on the recombination of the learning strategy parameters. The basic idea is to combine covariance matrices of neighboring individuals in order to speed up the learning of the covariance matrix by sharing the information about the topology of the search space gathered by neighboring individuals. The proposed approach modifies the rank-one-update of the covariance matrix by including the weighted sum of matrices which aggregate the information from all successful offspring at generation. The weight is calculated anew in each generation and it is different for each individual in the offspring population. The weight reflects the relevance of individual j for the covariance matrix update of i-s individual. The importance of the individual depends on its distance. The closer j is situated to i, the higher the weight it is assigned. Empirical evaluation on biobjective benchmark functions showed that the new update scheme significantly improved the performance of MO-CMA-ES.

Voß et al. [183] proposed MO-CMA-ES with improved step size adaptation. The step size adaptation scheme differs from those used in the previous works [94, 95, 128, 181, 182] in the concept utilized to consider generated offspring as successful. In the original MO-CMA-ES, a mutation is regarded as successful if the offspring ranks better than its parent. In contrast, the authors propose to regard a mutation as successful if the offspring is selected into the next parental population. Empirical results using a set of benchmark functions showed that the new step size adaptation improved considerably the performance of the MO-CMA-ES. Moreover, for the first time, experiments with MO-CMA-ES comprised three-objective problems. There was also observed that the new update scheme in general leads to larger step size, thereby improving the convergence speed and at the same time reducing the risk of convergence into local optima.

3.3.8 Estimation of Distribution Algorithm-Based Approaches

Estimation of distribution algorithms (EDAs) are population-based probabilistic search techniques that search solution spaces by learning and sampling probabilistic models [140]. EDAs explicitly extract statistical information from the selected solutions and build a probability distribution model of promising solutions, based on the extracted information. Subsequently, new solutions are sampled from the model thus built.

Zang et al. [191] proposed a regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA). RM-MEDA starts by randomly generating an initial population. Then, until the stopping condition is not met, the algorithm performs the following steps: (i) a probability model of the distribution of the solutions in the population is built, (ii) a new set of solutions is generated using the built model, (iii) a new population is selected from the combined set of the generated solutions and the previous population. It is assumed that the Pareto set of a continuous multiobjective optimization problem is a piecewise continuous $(m-1)$-D manifold, where m is the number of objectives. The $(m-1)$-dimensional local principal component analysis is used to partition the population into K disjoint clusters. The centroid of each cluster is assumed to be a $(m-1)$-D hyperrectangle. Then, the obtained clusters are used to build a model for each $(m-1)$-D hyperrectangle. Thereafter, in order to uniformly distribute the solutions on the Pareto set, the probability of selecting each model is calculated, which is proportional to its volume. A new solution is generated by summing a point drawn from the built model and a noise vector. A nondominated sorting-based selection and the crowding distance are used for choosing solutions for the next generation from the combined set generated trail solutions and the population at the previous iteration.

Zhou et al. [193] proposed a probabilistic model-based multiobjective evolutionary algorithm (MMEA). MMEA generalizes the idea used in RM-MEDA and adopts the same framework, however it differs in the modeling phase. The modeling phase in MMEA works as follows. First, a utopian PF is build. For this purpose, a $(m-1)$-D simplex is used as the utopian PF. Next, the used number of subpopulations K is determined by uniformly

randomly choosing from the set $\{1, 2 \ldots, K_{\max}\}$. After that, the K uniformly spread reference points are selecting on the utopian PF. For each reference point, a number of points from P closest to it are selected for forming clusters. In the obtained clusters, different subpopulations may overlap, which could improve the search performance between different reference points. Principal component analysis is performed on each subpopulation P^k. The subpopulation is modeled as a hyper-cuboid in the decision space. On the contrary to the RM-MEDA which sets the dimensionality of the PS manifold to be $(m-1)$, MMEA needs to estimate it. This difference is due to the fact that these two algorithms are for different MOPs. Moreover, two different classes of MOPs were defined: (i) first class, when PS and PF are of the same dimensionality; (ii) second class, PF is an $(m-1)$-D continuous manifold and PS is a continuous manifold of a higher dimensionality. Thus, RM-MEDA is addressed for dealing with the first class of MOP while MMEA is developed for approximating PS and PF of MOP of the second class.

3.3.9 Multiobjective Evolutionary Algorithm Based on Decomposition

So far, in this section, there have been mainly discussed EMO algorithms which are the extensions of single-objective analogues to multiobjective case. In the following, a multiobjective evolutionary algorithm based on decomposition (MOEA/D) is considered. MOEA/D is especially designed for solving multiobjective problems, and was first suggested by Zhang and Li [189]. MOEA/D represents an efficient framework for multiobjective optimization which combines decomposition techniques in mathematics and optimization methods in evolutionary computation. MOEA/D decomposes a multiobjective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously in a single run instead of solving a multiobjective problem directly. Moreover, neighborhood relations among the single objective subproblems are defined based on the distances among their weight vectors. For each weight vector, a certain number of closest weight vectors are associated which constitute the neighborhood to this vector. Each subproblem

is optimized by using information from its several neighboring subproblems. The major motivation behind this idea is that any information about the solutions for the neighboring subproblems should be helpful for optimizing a given subproblem. It comes from the assumption that subproblems in one neighborhood have similar fitness landscape and their respective optimal solutions may probably be close to each other.

In the original version of the algorithm [189], first the set of evenly distributed weight vectors $\{\lambda^1, \ldots, \lambda^N\}$ is generated. For each weight vector λ^i, T closest weight vectors are identified by calculating the Euclidean distance among them. These vectors constitute the neighborhood $B(i)$ of the i-th subproblem. Then, an initial individual for each weight vector is randomly generated. The population size is the same as the number of weight vectors. Next, for each subproblem i a couple of parents are randomly selected among its neighborhood $B(i)$ and a child solution is generated by crossover and mutation. Then, a newly generated solution is compared with all solutions in its neighborhood $B(i)$. Individuals are compared with each other by the scalarizing function with the weight vector λ^i. If the offspring is better than any solution in its neighborhood, it replaces this solution. This process is repeated until the stopping criterion is met. Moreover, an external archive is maintained in order to store all nondominated solutions found during the search.

In the second version of MOEA/D [123], a genetic operator is replaced by a differential evolution operator which along with the polynomial mutation are utilized to produce a child solution. Furthermore, two extra features were introduced comparing to its predecessor. The first extra measure for maintaining the population diversity in MOEA/D-DE, which is not used in MOEA/D-SBX, allows three parent solutions to be selected from the whole population with a low probability. In such a way, a very wide range of child solutions is expected to be generated due to the dissimilarity among these parent solutions, thereby enhancing the exploration ability of the search. The second extra measure in MOEA/D-DE for maintaining its population diversity controls the maximum number of solutions replaced by a child solution. In MOEA/D-SBX, it could be as large as neighborhood size. If a child solution is of high quality, it may replace most of current solutions to its neighboring subproblems. As a result, diversity among the population could be reduced

significantly. To overcome this shortcoming, the maximal number of solutions replaced by a child solution in MOEA/D-DE is bounded. Therefore, there is little chance that a solution has many copies in the population. MOEA/D-DE was tested on a set of challenging multiobjective optimization problems with complicated Pareto sets and exhibited a good performance.

Chen et al. [27] proposed an enhancing version of MOEA/D. Here, DE operator was replaced by a guided mutation operator for reproduction which takes the subproblem neighbors as the guided target. Moreover, a new update mechanism with a priority order was introduced. The main difference with the original update mechanism is that in the earlier versions of MOEA/D solutions are randomly picked from the neighborhood of the whole population and the update checking is done. In the proposed version, all subproblems are allocated in a priority queue. Each time the update process is done, the element at the head of the queue is checked first and then the others in the order specified by the queue. When the best solution of any subproblem is successfully updated, the subproblem is moved back to the tail of the queue. Another difference of this mechanism lies in the selection of parent solutions. For each evaluation, the element at the queue tail, which is the newly updated one is taken as a parent.

In the versions of MOEA/D proposed in [27, 123, 189], all subproblems are treated equally, each of them receives about the same amount of computational effort. However, in the work proposed in [190], the authors suggest to assign different amounts of computational effort to different subproblems as these subproblems may have different computational difficulties. Thus, the suggested MOEA/D with dynamical resource allocation (DRA) computes a utility for each subproblem. In this way, computational efforts are distributed to the subproblems based on their utilities. At each iteration, a fifth part of the whole population is selected. First, the extreme subproblems are selected, then the remaining ones are selected using 10-tournament selection based on their utility. The utilities are updated every 50 generations. The variation operators used in this work are the same as in [123]. This version of MOEA/D with a strategy for dynamical resource allocation won the IEEE Congress on Evolutionary Computation 2009 algorithm contest.

Chiang and Lai [28] proposed an improving version of MOEA/D using an adaptive mating selection mechanism. Along with the traditional MOEA/D framework where parent solutions are selected from the neighborhood in the objective space, here after a specified number of iterations the individuals allowed to mate are selected according to the Euclidean distance in the decision space. To alleviate the computational burden related with the calculation of the distances between individuals in the population, the proposed procedure is invoked every ϵ iterations. Moreover, the subproblems are classified into solved and unsolved. A subproblem is said to be solved if its solution is not improved for α consecutive generations. If a subproblem is solved, it is disabled and its individual does not take part in the reproduction process. But with time each disabled problem can be enabled if its solution is improved by the offspring generated by other subproblems. Besides, the β individuals with the largest crowding distance are always enabled. The experimental results confirm the benefits of the proposed adaptive mating selection mechanism and controlled subproblem selection.

Ishibuchi et al. [100] studied the use of different types of scalarizing functions simultaneously in MOEA/D. For this purpose, the weighted Chebyshev and the weighted sum were used to assign a fitness function value. There was observed that each scalarizing function has its own advantages and disadvantages. For instance, the weighted sum has higher convergence ability to drive the population toward the Pareto front whereas it does not have high diversity maintenance ability to widen the population along the Pareto front. On the other hand, the weighted Chebyshev has higher diversity maintenance ability whereas its convergence ability seems to be inferior to the weighted sum. There were proposed two implementation schemes in order to simultaneously use multiple scalarizing functions in a single MOEA/D algorithm. One is to use multiple grids of weight vectors where each grid is used by a single scalarizing function. The other is to alternately assign a different scalarizing function to each weight vector in a single grid. The effectiveness of these implementation schemes was examined through computational experiments on multiobjective 0/1 knapsack problems with two, four and six objectives. Experimental results showed that the simultaneous use of the weighted sum and the weighted Chebyshev outperforms

their individual use in MOEA/D. Especially in the case of the six-objective 0/1 knapsack problem, much better results were obtained from the proposed approach than the use of a single scalarizing function.

Zhou et al. [194] proposed a multiobjective algorithm based on a probability model under the MOEA/D framework. A multivariate Gaussian model is used to build a probabilistic model and sample new trial solutions from the model. At each step of the algorithm, a set of parent solutions is used from the neighborhood with probability p_n otherwise from the entire population with probability $1 - p_n$ to build a Gaussian model. A trial solution is then sampled from this model. When a model is built from the neighboring solutions it is said the algorithm does exploitation otherwise exploration. A multivariate Gaussian model is built by calculating the covariance matrix, as the mean vector a parent x^i which is the best solution for i-s subproblem is considered. Then, a new child solution is sampled from the built model and mutated by the polynomial mutation. Thereafter, the original MOEA/D update mechanism is invoked. The algorithm showed the ability to solve all tested problems however it was concluded that operators based on probabilistic models are computationally costly comparing to generic offspring production operators.

Mashwani and Salhi [129] studied the use of different crossover operators in MOEA/D with dynamical resource allocation. They used the simplex crossover (SPX) and center of mass crossover operator (CMX). Initially, two crossovers have equal probability of creating offspring, after each generation, each crossover operator receives a reward according to the number of successful offspring produced and probability is updated. A crossover operation is considered to be successful if its new generated solution can replace at least one solution. Whenever a crossover operator is successful it gets a reward of 1, otherwise it gets a reward of 0. Then the probability for each crossover to be applied is recalculated taken into account the reward received. Consequently, the more successful crossover operator is applied to more subproblems than the less successful one. The authors compared three versions of the algorithm, two with one crossover SPX and CMX and another incorporated two crossover SPX+CMX on the CEC'09 benchmark problems [192]. The results showed that the version with combined SPX and CMX crossovers clearly outperforms two other

versions with one crossover at a time. As a result, it was concluded that where a crossover is not so good, the other one is used instead. Therefore, the search process uses the most suitable crossover and the search effectiveness of the process is increased.

3.4 Evolutionary Many-Objective Optimization Algorithms

Multiobjective problems with four or more objectives are often referred to as many-objective problems. Many-objective optimization is one of the main research activities in EMO, mainly due to the fact that research in this area has revealed that the existing state-of-the-art EMO algorithms scale poorly with the number of objectives [47, 91, 108, 142, 153].

In order to examine the relation between the percentage of nondominated solutions in a population and the number of objectives, 300 vectors are randomly generated in the m-dimensional unit hypercube $[0, 1]^m$ for $m = 2, 3, \ldots, 20$. Among the generated 300 vectors for each dimension, the percentage of nondominated vectors is calculated. The average percentage over 30 runs for each m is shown in Figure 3.8.

From Figure 3.8, one can see that almost all vectors are nondominated when m is larger than 10. As a result, a strong selection pressure toward the Pareto front cannot be generated by the Pareto dominance-based fitness assignment mechanism. Moreover, distance metrics used as a second sorting criterion emphasizes less crowded solutions, which are far away from the Pareto front. The above difficulties are often referred as the curse of dimensionality.

In the following, a number of different approaches designed specifically to deal with many-objective optimization problems are discussed. The algorithms are structured as follows:

- Selection pressure enhancement,
- Different fitness evaluation schemes,

- Preference incorporation,

- Dimensionality reduction.

However, it should be kept it mind that the above classification is not absolute and there may exist methods which can be put to more than one category or other methods. A review of evolutionary many-objective optimization algorithms can be found in [102].

3.4.1 Selection Pressure Enhancement

A straightforward idea for the scalability improvement of EMO algorithms to many-objective problems is to increase the selection pressure toward the Pareto front. One approach based on this idea is to modify Pareto dominance in order to decrease the number of nondominated solutions in each population. Another approaches suggest to rank solutions in the population according to some introduced criterion or use advanced distance assignment metrics.

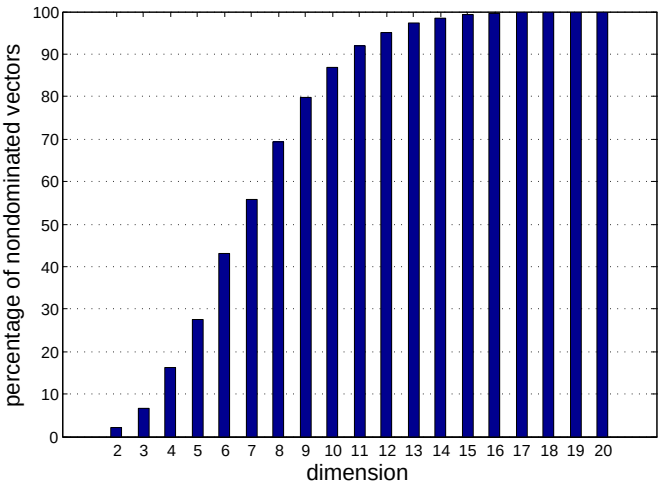


Figure 3.8: Average percentage of nondominated vectors among 300 randomly generated vectors in the m-dimensional unit hypercube.

Modification of Pareto Dominance

Sato et al. [152] demonstrated that the use of a modified dominance, instead of the standard Pareto dominance, clearly improved the performance of NSGA-II for many-objective problems. In order to increase the selection pressure toward the Pareto front by the Pareto sorting in NSGA-II, they modified Pareto dominance by widening the angle of the dominance region. In this manner, the selection pressure toward the Pareto front can be strengthened because the number of nondominated solutions in each population is decreased by the use of the modified dominance. The angle which extends the dominated region should be adjusted to the number of objectives. Roughly speaking, the increase in the number of objectives requires a wider angle of the dominated region.

Modification of Rank Definition

Drechsler et al. [60] proposed the use of a relation called *favour* to differentiate between nondominated solutions for the handling of many-objective problems. They defined the relation *favour* based on the number of objectives for which one solution is better than the other. More specifically, a solution x is viewed as being better than another solution y if x is better than y on more objectives than on which y is better than x. The relation *favour* can be defined as follows:

$$|\{i : f_i(x) < f_i(y), 1 \leq i \leq m\}| < |j : f_j(y) < f_j(x), 1 \leq j \leq m\}|.$$

The relation *favour* was modified in Sülflow et al. [168] by taking into account not only the number of objectives for which one solution is better than the other but also the difference in objective values between the two solutions.

On the other hand, Kukkonen and Lampinen [117] studied two different aggregation functions based on the ranks of solutions in terms of each separate objective. One fitness function uses the sum of ranks for each objective while the other uses the minimum rank among objectives. To maintain the diversity of the solutions, a linearly increasing diversity maintenance scheme is proposed. At the first generation, the population of the next

generation is selected from the combined parent and child population just based on the ranking-dominance relation. In the half of generations, half of the population of the next generation is selected based on the ranking-dominance relation from the combined parent and child population. The rest half of the population is filled with the best members according to diversity from the remaining individuals of the combined parent and child population. At the final generation, individuals from the combined parent and child population are selected just according to diversity. Experimental results indicated that in some cases the selection based on ranking-dominance is able to advance the search towards the Pareto front better than the selection based on Pareto dominance. However, in some cases it is also possible that the search does not proceed into the direction of the Pareto front because the ranking-dominance relation permits deterioration of individual objectives.

Corne and Knowles [32] compared various ranking methods with each other. They reported that the best results were obtained from a simple average ranking method than from more complicated ranking schemes. In the average ranking method, first a rank for each objective is assigned to each solution based on the ranking of its objective value for the corresponding objective among nondominated solutions in the current population. Thus each solution has M ranks, each of which is based on one of the M objectives. Then the average rank is calculated for each solution as its rank.

Substitute Distance Metrics

Köppen and Yoshida [114] discussed various substitute distance assignment schemes which are used to replace the crowding distance assignment in NSGA-II. These distances are based on measurement procedures for the highest degree, to which a solution is nearly Pareto dominated by any other solution: like the number of smaller objectives, the magnitude of all smaller or larger objectives, or a multi-criterion derived from the former ones. The following four measurements were considered.

Given two solutions A and B, the subvector dominance procedure $svd(A, B)$ counts the number of objectives of B that are smaller than the corresponding objectives in A. For

each solution $I[i]$ in a set I of solutions, the largest such value among all other solutions is assigned as distance value to $I[i]$. The smaller this value, the smaller is the number of lower objectives that appear among all other members of the set I. Such a solution is more close to being not Pareto dominated by any other solution.

On the other hand, the epsilon dominance procedure $mepsd(A, B)$ considers all objectives of B that are larger than the corresponding objectives of A. It computes the smallest value ϵ, which, if subtracted from all objectives of B, makes B Pareto dominating A. For each solution $I[i]$ in a set I of solutions, the smallest such value among all other solutions is assigned as distance value to $I[i]$. The larger this distance for a solution, the higher the effort that would be needed to make the other solutions Pareto dominating the former.

The fuzzy Pareto dominance procedure fuses all the magnitudes of larger objectives into a single value, instead of seeking the maximum difference. It uses the notion of bounded division of two reals x and y from $[0, 1]$:

$$\frac{x}{y} = \begin{cases} 1, & \text{if } y \leq x \\ x/y, & \text{if } x < y. \end{cases}$$

All bounded quotients of corresponding objectives in A and B are multiplied. For a smaller objective in B, this gives a factor of 1. For each solution $I[i]$ in a set I of solutions, the largest product value from all other solutions is assigned as distance value to $I[i]$. The smaller this value, the lower the degree by which a solution is dominated by any other solution in I.

The subobjective dominance count (SOD-CNT) procedure performs a multi-criterion ranking, where the number of larger objectives as well as its magnitude are considered. Taking a solution A of a nondominated solution set I, a set S_A of all pairs of two single-criterion distance measures ($svd(A, B)$ and $mepsd(A, B)$ were used) to all other solutions B of the set is derived. This set S_A has a Pareto set, which is composed of all solutions that perform well against A. Each solution in I gets the number of occurrences in all the possible Pareto sets assigned. The higher this number, the more often the corresponding solution performs well against some other solution in I.

For a number of many-objective test problems, all proposed substitute distance assignments resulted into a strongly improved performance of NSGA-II. Based on the analysis of the obtained results, the authors outlined two promising strategies for the application of NSGA-II to many-objective optimization problems: first is to replace the crowding distance completely by the epsilon dominance distance, second is to use the sub-objective dominance count distance for the first generations of the algorithm, as long as most of the individuals get rank 1 assigned, and switch to the crowding distance, once the SOD-CNT values tend to be equalized over the whole population.

3.4.2 Different Fitness Assignment Schemes

Another idea for the scalability improvement of EMO algorithms is the use of different fitness evaluation mechanisms instead of the Pareto dominance. One approach is to use a number of different scalarizing functions for fitness evaluation. Another approach based on this idea is the use of indicator-based evolutionary algorithms where indicator functions such as hypervolume are used to evaluate each solution.

Scalarizing-Based Fitness Assignment

Converting a multiobjective problem into a single-objective one by using different scalarizing functions is the basic idea behind many traditional mathematical programming methods for approximating the Pareto front. The main advantage of the use of scalarizing functions for many-objective problems is their efficiency. For example, weighted sums of multiple objectives can be easily calculated even when the number of objectives is large. On the other hand, the computation time for hypervolume calculation exponentially increases with the number of objectives.

In [91], there was shown that the use of many single objective optimizations based on aggregation functions is more effective than the use of a Pareto-ranking based optimizers on many-objective problems. However, the more efficient way is to use the ranking strategy based on the aggregation methods in the the single run of a multiobjective optimizer then

performing multiple runs of a single objective optimizer.

Hughes [90] proposed non-Pareto evolutionary multiobjective algorithm, called multiple single objective Pareto sampling (MSOPS). At every generation, the algorithm uses a set of target vectors to evaluate the performance of every individual in the population based on the chosen aggregation method. Thus, each member of the population has a set of scores that indicate how well the population member performs on each scalarizing function. The scores are held in a score matrix, which is ranked, with the best performing population member on the corresponding target vector being given a rank of 1, and the worst a rank of P. The rank values are stored in a rank matrix whose rows are sorted, with the ranks for each population member placed in ascending order. Then the rank matrix is used to rank the population, with the most fit being the solution that achieved the most scores which were ranked 1.

In [92], there was proposed the improved version of MSOPS, called MSOPS-II. In this work, two extensions are used compared to the previous work. The first provides automatic target vector generation, removing the requirement for initial a-priori designer intervention; and secondly redefines the fitness assignment method to simplify analysis and allow more comprehensive constraint handling. The significant enhancements allow the new MSOPS-II ranking process to be used as part of a general-purpose multi/many objective optimization algorithm, requiring minimal initial configuration.

In [93], Hughes proposed an algorithm for many-objective optimization. The algorithm is based on the same strategy as MSOPS [90] and MSOPS-II [92], which use a set of target vectors to direct and control the search process. Each target vector is coupled with at least one aggregation function that turns each objective vector into a scalar value for use in the optimization process. The algorithm operates by treating each target vector and associated aggregation function as a single-objective optimization process and tracks the parameter vector that best optimizes the aggregated objective values in the target vector direction. At each iteration, one target vector is chosen at random and its corresponding parameter vector is utilized to the line search. All points evaluated as part of line search are stored and used to update the population.

Ishibuchi et al. [97] proposed a hybrid algorithm, which employees an idea of probabilistically using a scalarizing fitness function in evolutionary multiobjective optimization. The authors used the NSGA-II algorithmic framework introducing two probabilities to specify how often the scalarizing fitness function is used for parent selection and generation update in the algorithms. When a pair of parent solutions are to be selected from the current population, the weighted sum fitness function and the NSGA-II fitness evaluation mechanism are used with the probabilities P_{PS} and $(1 - P_{\mathrm{PS}})$, respectively. As in the parent selection phase, the weighted sum fitness function is probabilistically used in the generation update phase. When one solution is to be chosen from the enlarged population and added to the next population, the weighted sum fitness function and the NSGA-II fitness evaluation mechanism are used with the probabilities P_{GU} and $(1 - P_{\mathrm{GU}})$, respectively. The computational experiments on multiobjective 0/1 knapsack problems showed that the proposed approach improves the performance of EMO algorithms.

Moreover, the scalarizing fitness functions were also used in other studies [98, 99, 103, 105]. Its use proved to be an efficient and effective way for fitness assignment in EMO algorithms. Furthermore, the scalarizing fitness assignment mechanism is used in MOEA/D, which was discussed in more detail in the previous section. In some recent studies, MOEA/D has been also applied to many-objective problems producing a good performance with the increasing number of objectives [44, 154].

Indicator-Based Fitness Assignment

A number of performance indicators have been proposed to quantitatively measure and compare the quality of outcomes produced by EMO algorithms. Assuming that a performance indicator provides a good ordering among sets that represent Pareto approximations, the fitness function used to select individuals may be defined in such a way that the chosen indicator is optimized. In fact, in recent years a trend can be observed to directly use specific measures such as the hypervolume indicator and the epsilon indicator to guide the search. The hypervolume is a popular performance evaluation measure but it was

rarely used until recently. This fact may be mostly explained by the complexity of the hypervolume calculation in terms of programming and computation time. However, theoretical studies [68, 203] showed that it is the only indicator known to be strictly monotonic with respect to Pareto dominance and thereby guaranteeing that the Pareto optimal front achieves the maximum hypervolume possible, while any worse set will be assigned a worse indicator value.

Several variants of indicator-based evolutionary algorithms have been proposed in the literature. In those variants, the hypervolume was almost always used as an indicator. However, the difficulty in the application of hypervolume-based algorithms to many-objective problem is a large computation cost for the hypervolume calculation. Knowles and Corne [110] were the first to propose the integration of the hypervolume indicator into the optimization process. In particular, they described an adaptive archiving algorithm to maintain a separate, bounded archive of nondominated solutions based on the hypervolume indicator.

Huband et al. [89] presented MOEA, which includes a modified SPEA2 environmental selection procedure, where a hypervolume-related measure replaces the original density estimation technique. They involved the product of the one-dimensional lengths to the next worse objective function value in the front for each objective as selection criterion in case of equal fitness values. In the two-objective case, this exactly corresponds to the S metric contribution. In case of more than two objectives, this procedure provides only a lower bound of the hypervolume value. They presented an evolution strategy with probabilistic mutation (ESP). It extends traditional evolution strategies in two principal ways: ESP applies mutation probabilistically in a GA-like fashion, and it uses a new hypervolume-based, parameterless, scaling independent measure for resolving ties during the selection process.

A general framework of indicator-based evolutionary algorithm (IBEA) was proposed by Zitzler and S. Künzli [198]. This approach can use an arbitrary indicator to compare a pair of candidate solutions, in their work authors consider the epsilon and hypervolume indicators. In the IBEA, any additional diversity preservation mechanism such as fitness

sharing is no longer required. In comparison to other EMO algorithms, the IBEA only compares pairs of individuals instead of entire approximation sets. A scheme similar to summing up the indicator values for each population member with respect to the rest of population is used to assign the fitness for each individual in the population. Binary tournament selection is performed in order to fill the temporary mating pool. Then, variation operators are applied to generate a set of offspring. Environmental selection is performed by iteratively removing from the combined population of parents and offspring a solution with the smallest fitness value and recalculating the fitness of the remaining individuals. Wagner et al. [184] reported good results by IBEA for many-objective problems. Since IBEA does not use Pareto dominance, its search ability is not severely deteriorated by the increase in the number of objectives. Ishibuchi et al. [101] proposed an iterative version of IBEA to decrease the computation cost by searching for only a small number of representative solutions. For the same purpose, Brockhoff and Zitzler [19, 20] examined dimensionality reduction in indicator-based evolutionary algorithms.

Emmerich et al. [64] and Beume et al. [13] suggested an EMO algorithm adapted specifically to the hypervolume indicator, called S metric selection evolutionary multiobjective algorithm (SMS-EMOA). It was the first EMO algorithm aiming explicitly at the maximization of the dominated hypervolume within the optimization process. Moreover, the proposed algorithm is the first steady-state evolutionary algorithm for multiobjective optimization. SMS-EMOA features a selection operator based on the hypervolume measure combined with the concept of nondominated sorting. The algorithm's population evolves to a well-distributed set of solutions, thereby focussing on interesting regions of the Pareto front. The algorithm was compared with state-of-the-art EMO algorithms on two and three-objective benchmark suites as well as on aeronautical real-world applications providing superior performance in terms of the hypervolume indicator. Furthermore, a fast algorithm for the calculation of the contributing hypervolume for three-dimensional case is suggested in [13].

Bader et al. [6] proposed a method to calculate the contributing hypervolume of solutions in the population using Monte Carlo simulation. The proposed procedure works

in two steps. First, a number of points are sampled within a hyperrectangle, next the number of hits is calculated and the hypervolume contribution is evaluated as the hit ratio multiplied with the volume of the sampling box. Moreover the algorithm uses an adaptive scheme to reduce the total number of samples as well as a statistical test to determine the probability that the solution with the smallest contribution has obtained the smallest contribution estimate [5].

In [7], Bader and Zitzler suggested a fast hypervolume-based EMO algorithm called HypE. It is a successor of the algorithm proposed in [6]. It also uses a fast method based on Monte Carlo simulations to estimate the hypervolume value of an approximation set, however, the entire objective space is utilized to draw samples. Additionally, both its environmental and mating selection step rely on a new fitness assignment scheme based not only on the hypervolume contribution, but also on parts of the objective space dominated by more than one solution. The main idea is not that the actual indicator values are important, but rather that the rankings of solutions induced by the hypervolume indicator.

Besides the algorithms based on optimizing popular quality indicators, such as the hypervolume and epsilon indicator, some recent studies suggest approaches which incorporate other performance metrics. Thus, Bringmann et al. [18] proposed an EMO algorithm that works with a formal notion of approximation and improves the approximation quality during its iterative process. In every generation, the algorithm aims at minimizing the additive approximation of the population with respect to the archive of nondominated solutions found so far. For each solution in the combined population of parents and offspring, the approximation is computed which is the approximation of the set of all solution except the solution for which the approximation is computed. The individuals with the smallest such value are iteratively removed. The algorithm produced competitive results with state-of-the-art algorithms on the set of problems having up to 20 dimensions.

Rodríguez Villalobos and Coello Coello [148] proposed an EMO algorithm based on the performance indicator called Δ_p. The Δ_p indicator consists of GD and IGD metrics and is viewed as an averaged Hausdorff distance [158]. The algorithm uses the reference set to approximate the Pareto front of MOP. The selection procedure is solely based on

the Δ_p indicator. After generating offspring population, the individuals with the lowest contribution to Δ_p are selected from the combined population to form the population of the next generation.

3.4.3 Use of Preference Information

Besides the difficulty related with the dominance based selection mechanism discussed at the beginning of this section, there is another difficulty which rises dealing with many-objective problems. As the number of objectives increases, the objective space becomes larger. In turn, this requires the exponential increase in the number of nondominated solutions that are necessary for the approximation of the Pareto front. However, it may become unacceptably difficult and computationally expensive to process a population with a larger number of solutions by EMO algorithms which are usually designed to search for a set of nondominated solutions that approximates the entire Pareto front. Therefore, it is a good idea to focus on a specific region of the Pareto front using the decision makers preference. In the following, we briefly discuss some studies where the preference information is incorporated in EMO algorithms in order to concentrate on a small region of the Pareto front handling high-dimensional objective space.

Fleming et al. [69] described a preference articulation method to many-objective problems where the focused region gradually becomes smaller during the evolutionary process. The parallel coordinates representation is used after a number of generations to represent the population to the decision maker. After analyzing the plots, some of the objectives are converted to be constraints. The values are constrained to the worst solution for that objective shown on the plot. Thereafter, the optimizer is released to cycle through more generations. After that, the process of isolating the best solution and converting its objective into a constraint is repeated. In this way, the search is progressively concentrated on the desirable objectives at the same time optimizing them until a very small set of solutions is obtained.

Deb and Sundar [49] incorporated reference point-based preference information into

NSGA-II. In their approach, the normalized distance to a reference point is taken into account to evaluate each solution after Pareto sorting. This means that the normalized distance is used as a secondary criterion instead of the crowding distance in NSGA-II. A number of nondominated solutions are obtained around the reference point. Multiple reference points can be handled to explore the decision maker's preferred regions of the Pareto front.

Thiele et al. [173] used reference point-based preference information in IBEA. First a rough approximation of the Pareto front is obtained. Such a rough approximation is used by the decision maker to specify a reference point. Then IBEA searches for a number of nondominated solutions around the reference point. The focus of the multiobjective search by IBEA can be controlled using a parameter value and a reference point. Usually, the focused region gradually becomes smaller through the interaction with the decision maker. The use of preference information is based on a similar idea to weighted integration in [196]. In the abovementioned methods, preference information is used to focus on a specific region in the high-dimensional objective space with many objectives while EMO algorithms are used to search for a number of nondominated solutions in such a focused region.

Deb and Jain [44] suggested a reference-point based many-objective NSGA-II, called MO-NSGA-II. MO-NSGA-II emphasizes population members which are nondominated yet close to a set of well-distributed reference points. The basic framework of MO-NSGA-II remains similar to the original NSGA-II, but significant changes were made in the selection mechanism. At every generation, MO-NSGA-II builds a hyper-plane on which a predefined set of evenly distributed reference points is generated. Then, the population members from the last nondominated front being retained to the next generation are projected to the constructed hyperplane and associated with a reference point that is closest to the projected solution. The diversity among obtained solutions is ensured by emphasizing population members which form less-crowded clusters around each reference point. During the parent selection procedure, when two solutions from the same nondominated front are compared the one which is associated with less-represented reference point is preferable. MO-NSGA-II showed a good performance on problems having up to 10 objectives.

3.4.4 Dimensionality Reduction

Another approach for dealing with many-objective problems is dimensionality reduction. These approaches assume the existence of redundant objectives in a given many-objective problem. Operating on the objective vectors of the nondominated solutions obtained from an EMO algorithm, these approaches aim to identify a smallest set of conflicting objectives which: (i) generates the same Pareto front as the original problem, or (ii) alternatively, preserves the dominance relations of the original problem. Such objectives are termed as essential and the remaining ones as redundant. Dimensionality reduction, always when it is possible, can remedy virtually all difficulties related with many-objective problems, namely it will contribute to higher search efficiency, lower computational cost and ease in visualization and decision-making.

Dimensionality reduction approaches can be classified into linear and nonlinear approaches. Furthermore, it may be referred to online or offline reduction method depending on whether it is performed during an EMO algorithm run to simplify the search or post EMO algorithm run to assist in the decision making, respectively.

Saxena and Deb [47] proposed dimensionality reduction method based on principal component analysis (PCA). In their proposal, referred as PCA-NSGA-II, a representative set of solutions for dimensionality analysis is obtained by running NSGA-II for a large number of generations. Thereafter, the correlation matrix R, with respect to the objectives, is computed using the objective values of the final population. The eigenvalues and corresponding eigenvectors are then analyzed in order to reduce the objectives. The procedure starts with the original set of objectives, and the objectives are eliminated iteratively, based on their contribution to the principal components and the degree of their conflict with the other objectives. Once the set of objectives cannot be reduced further, the procedure is stopped and the reduced set of objectives is declared. However, the proposed linear PCA technique has a drawback of misinterpreting the data when it lies on submanifolds.

To mitigate the difficulties related to the use of linear PCA for dimensionality reduction, Saxena and Deb [153] suggested nonlinear dimensionality reduction-based techniques.

While the reduction procedure remained largely the same as in [47], the key difference lay in the way the correlation was calculated. Instead of the linear PCA, two new proposals were made: correntropy PCA (C-PCA) and maximum variance unfolding (MVU). The corresponding algorithms were named C-PCA-NSGA-II and MVU-PCA-NSGA-II. Using these nonlinear techniques, more accurate results were obtained for up to 50-objective DTLZ5-(2,M) problems. Even so, the algorithms were found to be ineffective on a number of other problems, such as DTLZ5-(5,10) and DTLZ5-(5,20), since the population for dimensionality analysis was not converged near enough to the Pareto front for a meaningful analysis.

The dimensionality reduction approach proposed in [21] is based on preserving the dominance relations in the given nondominated solutions. For a given objective set $\mathcal{F}$, if the dominance relations among the objective vectors remains unchanged when an objective $f \in \mathcal{F}$ is removed, then f is considered to be nonconflicting with the other objectives in $\mathcal{F}$. Based on this criterion, an exact and a greedy algorithm is proposed to address δ-MOSS, finding the minimum objective subset corresponding to a given error δ and k-EMOSS, finding an objective subset of size k with minimum possible error problems. However, due to the underlying assumptions, this approach is limited to linear objective reduction and equally distributed solutions in the objective space.

López Jaimes et al. [127] have developed a dimensionality reduction scheme based on an unsupervised feature selection technique. In their approach, the objective set is first divided into homogeneous neighborhoods based on a correlation matrix of a nondominated set obtained using an evolutionary algorithm. The conflict between the objectives takes role of a distance, meaning the more conflict between the objectives, the more distant they are in the objective conflict space. Thereafter, the most compact neighborhood is chosen, and all the objectives in it except the center one are dropped, as they are the least conflicting.

Singh et al. [162] proposed Pareto corner search evolutionary algorithm (PCSEA). The algorithm searches for only the corners of the Pareto optimal front instead of aiming for the complete PF. The solutions so obtained are assumed to appropriately capture the

dependency of PF on different objectives. To find the corners solutions, PCSEA uses corner-sort ranking procedure. The solutions are sorted based on the increasing individual objective values and increasing all-but-one objective, converted to a single composite value using L_2 norm values. Then, dimensionality reduction procedure is performed. It is based on the premise that omitting a redundant and an essential objective will have negligible and substantial effect, respectively, on the number of nondominated solutions in the population. Starting from the first objective, each objective is dropped in turn, and the change in the number of nondominated solutions is observed. If the ratio between resulting nondominated solutions and the reference set exceeds a user-defined threshold, then the objective is deemed redundant and removed. However, there are a number of limitations related to the method due to the use of corner solutions only to identify dimensionality. The resulting set may not be able to capture the entire information regarding the conflict in the Pareto front, there also may be the situation where certain objectives do not contribute to the extremities of the Pareto front, but contribute elsewhere.

3.5 Summary

The goal of multiobjective optimization to approximate the Pareto set is dual in nature, it requires obtaining a set of solutions as close as possible to the true Pareto optimal set and, at the same time, as diverse as possible. Algorithms for solving multiobjective optimization problems must be able to satisfy this two goals. This chapter reviews a number of approaches designed to perform this task.

The majority of classical methods treat multiobjective optimization in the same way as single-objective optimization. They avoid the complexity induced by the presence of multiple objectives by converting a multiobjective optimization problem into a single-objective optimization problem with a real-valued objective function. This approach is referred to as scalarization, and the function that depends on user-defined parameters is termed the scalarizing function. In summary, the weighted sum method consists in minimizing a weighted sum of multiple objectives; ϵ-constraint method suggests optimizing one ob-

jective function imposing all other objectives as constraints; weighted metric methods are based on minimizing an L_p metric constructed from all objectives. The normal boundary intersection method suggests maximizing the distance along the normal vector towards the origin from a point on the constructed hyper plane satisfying additional equality constraint. Finally, the normal constraint method provides the reduction of the feasible region by imposing additional constraints and optimizing one objective.

All these methods result in a single-objective optimization problem, which must be solved using a single-objective optimization algorithm. In most cases, the optimal solution to the single-objective optimization problem is expected to be a Pareto optimal solution. Such a solution is specific to the parameters used in the conversion method. To find different Pareto optimal solutions, the parameters must be changed and the resulting new single-objective optimization problem has to be solved again. On the other hand, Timmel's method is a population-based stochastic approach, hence it does not require multiple runs with different parameter settings. It is expected to find the approximation to the Pareto optimal set in a single run. Its stochastic nature to generate new solutions and the use of a population of solutions instead of a single one represent philosophy similar to that used in evolutionary algorithms. However, it does not employ any specific strategy to provide the diversity among the obtained solutions. In turn, other classical methods rely on the uniformly distributed set of weight vector to achieve the diversity of the obtained solutions or on the value of ϵ in the case of the ϵ-constrained method.

Despite the simplicity of use and good theoretical properties, there are some limitations related to the classical methods. In particular, some algorithms do not guarantee finding solutions in the entire Pareto optimal region after performing a number of single-objective optimizations. Thus, the weighted sum method and the weighted metric method with small p have limitations in finding solutions in the nonconvex region of the Pareto optimal front. Furthermore, some methods require additional knowledge about the problem that must be provided in advance. Thus, the weighted metric method uses a reference point. The normal boundary intersection method and the normal constraint method use the anchor points to convert a multiobjective problem into a single-objective optimization problem.

The main limitation of Timmel's method is the use of the gradients of objectives to generate new solutions. This requires all objectives in the multiobjective optimization problem to be differentiable. However, Timmel's method can find Pareto optimal solutions in both convex and nonconvex regions of the Pareto front.

Contrarily to the majority of classical methods, evolutionary algorithms are population-based optimization techniques. This feature makes them particularly suitable for solving multiobjective optimization problems. The main aspects to be taken into consideration when implementing MOEAs are: (i) fitness assignment, (ii) diversity preservation, and (iii) elitism. Although the discussed algorithms are organized on the basis of their variation operators, one can trace how each of these aspects was included into MOEAs since the first pioneering studies appeared in the mid 1980s. Regarding the fitness assignment, one can distinguish the following most popular techniques: dominance-based, scalarizing-based, and indicator-based fitness assignment. Furthermore, diversity of solutions within the current Pareto set approximation is maintained by incorporating density information into the selection process of EMO algorithms. Some widely used diversity preserving methods are: (i) fitness sharing, (ii) hypergrid, (iii) clustering, (iv) nearest neighbor, (v) crowding distance. Recently, the use of quality indicators as the diversity estimator has become very popular. Elitism is addressed to the issue of maintaining good solutions found during the optimization process. The elite preserving operator is usually included in MOEAs to make them better convergent to the Pareto front. One way to deal with this problem is to combine the parent and offspring populations and to apply selection procedure to select a new population. Alternatively, the external archive can be maintained where the promising solutions are stored during the search.

The early MOEAs are mostly GA-based non-elitist algorithms. The first multiobjective evolutionary algorithm is VEGA. During its selection process individuals are selected into the mating pool based on the particular objective function values. However, it does not use any diversity technique. Later approaches proposed in the early 1990s (MOGA, NSGA, etc.) followed the Goldberg's suggestion of incorporating the concept of the Pareto dominance in the fitness assignment mechanism during the mating selection. The most

popular technique, at the time, to maintain the diversity among solutions is based on a sharing function. The early MOEAs showed the ability to approximate the Pareto set on a number of test functions and real-world applications, they provided substantial foundations for developing further generations of MOEAs.

In the early 2000s, the selection using the Pareto dominance-based fitness assignment became the most popular one. Additionally, the elitism is somehow implemented in almost all algorithms of that time. Until now, the most popular algorithm framework is the one used in NSGA-II. NSGA-II carries out a nondominated sorting of a combined population. The use of the combined population instead of simply replacing the old population by offspring makes the algorithm an elitist type. The introduced crowding distance is used to decide which solutions from the last accepted nondominated front are included into the new population. This operator preserves diversity of solutions in the population. Subsequently, a number of single-objective EAs (ES, DE, CMA-ES, etc.) were extended to deal with multiobjective optimization problems by simply replacing the GA operator in the NSGA-II framework. These proposals take advantage of their offspring generation operators adopting the same strategy to guide the search in the objective space. On the other hand, other algorithms are extended with some modifications. For instance, in MEES the elitism is implemented in such a way that a specified portion of nondominated solutions is introduced in the main population from the archive, while the main population contains mainly offspring produced by the previous population.

However, it is not possible for some single-objective EAs to be extended to deal with multiple objectives by only modifying their selection operators. For example, PSO requires to define the notion of the local and global best particles to guide the search of a particle. In general, this is accomplished using the external archive where the local and global best solutions are kept. Decision on the selection of the local and global best particles is usually made using the Pareto dominance relation. Other algorithms use a local search strategy to approximate the Pareto set. Thus, PAES uses point to point generation strategy, where one offspring is created and compared with the parent using the dominance relation. In the case when the parent and offspring are both nondominated with respect to each other,

the offspring is compared with the current archive. Once again, decision on acceptance of offspring is made considering the dominance relation and the density of solutions. Moreover, AMOSA uses SA updating rule to choose the parent of the next generation when an offspring is dominated by the parent. Furthermore, the presence of the external archive in these algorithms to store nondominated solutions found during the search ensures the elitism.

A different class of algorithms from the ones discussed so far is MOEA/D. Contrarily to other considered algorithms, which were developed on the basis of their single-objective counterparts, MOEA/D was designed specifically to handle multiobjective optimization problems. In MOEA/D, each solution is associated with one scalarizing function. MOEA/D optimizes multiple subproblems in a single run exploring the information about fitness landscape of the neighboring subproblems. Different variation operators (GA, DE, EDA, etc.) can be used to generate offspring individuals. In general, MOEA/D represents a flexible and powerful algorithmic framework, where such aspects as fitness assignment, diversity preserving and elitism are implemented through the use of a set of uniformly distributed weight vectors. Moreover, it can be easily hybridized using local search algorithms.

Although state-of-the-art Pareto dominance-based EMO algorithms usually work well on two- and three-objective problems, a number of studies in the early 2000s revealed that their search ability is severely deteriorated when the number of objectives increases. This weakness of the Pareto dominance-based fitness assignment prompted further research in the field of many-objective optimization. To improve the scalability of EMO algorithms, some studies suggest to increase the selection pressure toward the Pareto front modifying the Pareto dominance relation or assigning different ranks to nondominated solutions. On the other hand, the decision maker's preferences can be used during the search in order to concentrate on preferred regions of the Pareto front. Another idea for the scalability improvement is to use different fitness assignment mechanisms instead of the Pareto dominance. One approach to accomplish this is to incorporate the quality indicators (such as the hypervolume and epsilon indicator) in the fitness assignment mechanism. Another

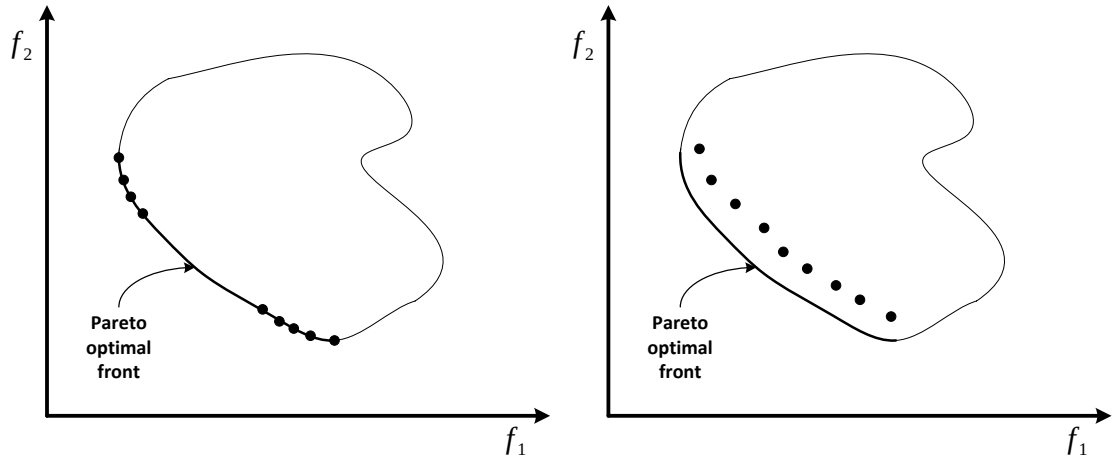
approach is to use a number of different scalarizing functions for assigning the fitness to individuals in the population. Another perspective to handling many-objective problems is to decrease the number of objectives. This approach is based on the notion that not all objectives are necessary to define the Pareto front. For this purpose, linear and nonlinear correlation-based dimensionality reduction techniques based on principal component analysis and maximum variance unfolding can be used. On the other hand, dimensionality reduction can be based on the Pareto dominance.

Chapter 4

Performance Assessment of Multiobjective Optimization Algorithms

4.1 Introduction

This chapter addresses the issue of performance assessment of different MO algorithms. Generally, in optimization the notion of performance includes both the quality of the outcome as well as the computational resources needed to generate this outcome. Concerning the latter aspect, the number of function evaluations or the overall computing time is commonly used. In this case, there is no difference between single and multiobjective optimization. However, there is a significant difference concerning the aspect related with the produced outcome. In single-objective optimization, the quality of the outcome can be defined by means of the objective function value: the smaller the value, the better performance (assuming minimization). While in multiobjective optimization, the outcome of the algorithm is usually a set of solutions. In the following, the outcome of a multiobjective optimizer is considered as a set of incomparable solutions, which is denoted as an approximation set. In terms of the objective space, this can be formalized as follows [204].



(a) Approximation set obtained by Algorithm 1 (convergence is good, distribution is poor)

(b) Approximation set obtained by Algorithm 2 (convergence is poor, distribution is good)

Figure 4.1: Performance produced by two algorithms on a same problem.

Definition 4.1.1 (approximation set)**.** *Let $A \subseteq \mathbb{R}^m$ be a set of objective vectors. A is called an approximation set if any element of A does not weakly dominate any other objective vector in A. The set of all approximation sets is denoted as Ω.*

The motivation behind this definition is that all solutions dominated by any other solution returned by the optimization algorithm are of no interest, and therefore can be discarded. Nevertheless, the above definition does not comprise any notion of quality, while it is obvious that the final goal of an optimization process is not to obtain any approximation set, but a good approximation set.

As it is mentioned in this thesis, there are two distinct goals in multiobjective optimization. Therefore, there are two primary requirements to any approximation set: (i) it should be as close as possible to the true Pareto optimal front, and (ii) it should be as diverse as possible. However, it is not straightforward how to compare, under these distinct and somewhat conflicting requirements, approximation sets produced by different algorithms.

The difficulty in comparing two approximation sets can be easily understood observing Figure 4.1. The figure illustrates two approximation sets obtained by two different

algorithms on the same problem. It can be seen that the approximation set produced by the first algorithm (Figure 4.1(a)) converges well to the Pareto optimal front, but clearly there is lack of diversity among solutions. This algorithm has failed to provide information about the intermediate Pareto optimal region. On the other hand, the second algorithm (Figure 4.1(b)) has obtained a well distributed set of solutions, but they are not close to the true Pareto optimal front. Although the latter approximation set can provide a rough idea of different trade-off solutions, the exact Pareto optimal solutions are not discovered. So, with such sets of obtained solutions, it is difficult to conclude which approximation set is better in an absolute sense.

Due to the stochastic nature of evolutionary algorithms, it is common practice to perform several runs on a chosen set of test problems in order to assess their performance. The produced outcomes are quantitatively assessed using quality indicators. Quality indicators can be categorized as follows: (i) first type of indicators measures the convergence to the Pareto optimal front, (ii) second type of indicators measures the diversity among the obtained solutions, and (iii) third type of indicators can be used to measure both goals of multiobjective optimization. The formal definition of quality indicators is provided later in this chapter. Assigning real values to each approximation set allows to perform common statistical tests and to obtain quantitative and statistically sound inferences about the algorithm's performance.

In the following, this chapter reviews some existing test suites for multiobjective optimization, a number of unary quality indicators and a few statistical comparison methods that can be used for performance assessment of multiobjective optimizers.

4.2 Benchmark Problems

Benchmark problems are an important aspect of optimization. They are often used for both assessing the qualities of optimization algorithms and designing new algorithms. Artificially constructed test problems offer many advantages over real-world problems for the purpose of general performance testing. Test problems can be designed to be easy to describe,

easy to understand, easy to implement, fast, and their optima are often known in advance. This is also true for multiobjective optimization. First multiobjective test problems that have been designed in the early stages of EMO research are relatively simple. Moreover, in most problems neither the difficulty caused by such problems for multiobjective algorithms cannot be controlled nor the dimensionality can be changed. As a result, they are rarely used nowadays. So, during the last decade a number of publications has appeared in the corresponding literature where the issue of designing challenging test problems has been addressed. This section reviews some recent benchmark problems that are widely used for comparing the performance of EMO algorithms.

Deb [41] first suggested a general methodology for constructing two-objective problems from single-objective optimization problems. The proposed approach allows known difficult features of single-objective problems (such as multi-modality, isolation, or deception) to be directly transferred to the corresponding multiobjective problem. Based on this construction process Zitzler et al. [197] framed six problems, since then they have been widely used and are known as ZDT test suites.

Furthermore, in order to study the performance of EMO algorithms on problems with more than two objectives Deb et al. [50] proposed DTLZ test suite. The suggested test problems, unlike all other earlier multiobjective test problems, are scalable with respect to the number of parameters as well as the number of objectives. This important characteristic has facilitated a number of investigations in the field of many-objective optimization. In the original technical report [50], seven unconstrained (DTLZ1-DTLZ7) and two constrained (DTLZ8 and DTLZ9) problems were suggested, while later in the conference paper [51] some of the problems were dropped.

Despite the extensive use of scalable DTLZ test suite in many EMO studies, it has been very often criticized for being too simple and due to its limitations. Specifically, none of its problems is deceptive, none of its problems is non-separable, and the number of position parameters is always fixed relative to the number of objectives. Moreover, DTLZ5 and DTLZ6 do not behave as expected, their Pareto fronts are unclear beyond three-objectives [87]. So, Huband et al. [87, 88] proposed a toolkit for creating problems with an

arbitrary number of objectives, where desirable features can easily be incorporated. The proposed WFG toolkit defines a problem in terms of an underlying vector of parameters $\boldsymbol{x}$. The vector $\boldsymbol{x}$ is associated with a simple underlying problem that defines the fitness space. The vector $\boldsymbol{x}$ is derived, via a series of transition vectors, from a vector of working parameters $\boldsymbol{z}$. Each transition vector adds complexity to the underlying problem. The WFG toolkit allows a user to control, via a series of composable transformations, which features will be present in the test problem. To create a problem, the user selects several shape functions to determine the geometry of the fitness space, and employs a number transformation functions that facilitate the creation of transition vectors. A number of shape and transition functions were suggested. Additionally, the authors proposed nine test problems (WFG1-WFG9) that possess some of the pertinent problem characteristics, some of of them are nonseparable, multimodal, deceptive problems and with different geometry of the Pareto front. Moreover, all problems are scalable objective- and parameter-wise, where the number of position- and distance-related parameters can be controlled by the user.

Deb et al. [48] addressed the issue of developing multiobjective test problems which introduce controllable linkages among variables using ZDT and DTLZ test problems. The authors studied three types of variable linkage: (i) linkages among variables affecting either the convergence or diversity individually, (ii) linkages among all variables causing a simultaneous effect in both convergence and maintenance of diversity among solutions, and (iii) linkages which are non-linear, causing linear operators to face difficulty in preserving optimality of solutions.

In [177], 19 multiobjective test problems are described, including two-objective, three-objective, and five-objective problems. The major part of the proposed test suite is constructed from the known benchmark functions. Thus, some of ZDT and DTLZ test functions were shifted or rotated thereby adding additional complexity and overcoming problems related with the original functions. The resulting test suite presents a variety of difficulties to EMO algorithms and was used in the CEC 2007 Special Session and Competition.

Li and Zhang [123] introduced a general class of continuous multiobjective optimization test instances with arbitrarily prescribed Pareto set shapes. Like DTLZ and WFG test suites, the proposed test suite uses component functions for defining its Pareto front and introducing multimodality. Its major advantage over others is that the Pareto set can be easily prescribed. The authors proposed nine challenged problems with complicated Pareto set shapes, which take challenges to many MOEAs. Moreover, in the CEC 2009 Special Session and Competition, unconstrained and constrained test functions are designed with complicated Pareto set shapes [143]. Furthermore, Saxena et al. [155] extended the framework of constructing test instances with controlling Pareto sets to the case of four and more objectives.

Moreover, a number of multiobjective test problems were suggested for performance comparison of MOEAs in different studies [65, 94, 138, 191, 193]. In turn, the paper [67] addresses the issue of developing dynamic multiobjective test problems by suggesting a baseline algorithm. A suite of five test problems offering different patterns of changing the Pareto front with time and different difficulties in tracking the dynamic Pareto optimal front are presented. Additionaly, a review of so-called classical multiobjective test problems can be found in [29, 88].

4.3 Quality Indicators

Various quality indicators for measuring the quality of approximation sets have been proposed to compare the performance of different multiobjective optimization algorithms. Some aim at measuring the distance of an approximation set to the Pareto optimal front. Other indicators try to capture the diversity of an approximation set. However, some of the quality indicators are able to measure the convergence to the true Pareto optimal front as well as diversity of solutions in an approximation set. Generally speaking, it can be stated that quality indicators map approximation sets to the set of real numbers. The following definition formalizes this notion as stated in [204].

Definition 4.3.1 (quality indicator)**.** *A unary quality indicator is a function* $I : \Omega \mapsto \mathbb{R}$ *that assigns each approximation set a real number.*

The underlying idea behind the use of the quality indicators is to quantify quality differences between approximation sets by applying common metrics to the resulting real numbers. The indicators can be either unary or binary quality indicators; however, in principle, a quality indicator can take an arbitrary number of arguments. In the following, all indicators are categorized into three categories, and only the unary quality indicators are presented.

4.3.1 Indicators Evaluating Convergence

The indicators presented in this section explicitly compute a measure of the closeness of an approximation set A from the Pareto optimal front $\mathcal{PF}$. They provide a good estimate of convergence if a large set for $\mathcal{PF}$ is chosen. On the one hand, in some problems, the Pareto front may be known and a set of finite Pareto optimal solutions can be computed. On the other hand, when the Pareto front is not known, a reference set can be used instead.

Error Ratio Indicator

The error ratio indicator was proposed by Veldhuizen [179] to indicate the percentage of solutions from an approximation set A that are not members of the Pareto front $\mathcal{PF}$. This indicator is defined as:

$$I_{ER} = \frac{\sum_{i=1}^{|A|} e_i}{|A|} \tag{4.3.1}$$

where $e_i = 0$ if vector i is a member of $\mathcal{PF}$, and $e_i = 1$ otherwise. It is clear that $I_{ER} = 0$ indicates an ideal behavior, since it would mean that all vectors generated by a multiobjective optimizer belong to $\mathcal{PF}$. It should be noted that this metric requires knowing the number of elements of the Pareto optimal front, which is often impossible to determine.

Maximum Pareto Front Error Indicator

The maximum Pareto front error indicator was suggested by Veldhuizen [179] and determines a maximum error band which when considered with respect to an approximation set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$ encompasses every vector in the Pareto front $\mathcal{PF} = \{\boldsymbol{a}'_1, \ldots, \boldsymbol{a}'_{|\mathcal{PF}|}\}$. Stating otherwise, this is the largest minimum distance between each vector in A and the corresponding closest vector in $\mathcal{PF}$. This indicator is defined as:

$$I_{ME} = \max_k \min_n \left(\sum_{i=1}^{m} \left| a_i^n - a_i'^k \right|^p \right)^{\frac{1}{p}} \tag{4.3.2}$$

where $n = 1, \ldots, |A|$, $k = 1, \ldots, |\mathcal{PF}|$, m is the number of objectives, and $p = 2$. A result of 0 indicates $A \subseteq \mathcal{PF}$, any other value indicates at least one vector in A is not in $\mathcal{PF}$.

Generational Distance Indicator

The generational distance (GD) indicator was introduced by Veldhuizen and Lamont [180] as a way of estimating how far are the elements in an approximation set A from those in the Pareto front $\mathcal{PF}$ and is defined as:

$$I_{GD} = \frac{\left(\sum_{i=1}^{|A|} d_i^p \right)^{\frac{1}{p}}}{|A|} \tag{4.3.3}$$

where d_i is the Euclidean distance from i-th point in an approximation set A to its nearest member of the Pareto optimal front $\mathcal{PF}$.

Set Coverage Indicator

The set coverage indicator was proposed by Zitzler [**?**] for comparing the performance of two algorithms by calculating the proportion of solutions in an approximation set produced by one algorithm which are weakly dominated by solutions in an approximation set produced by another algorithm. However, if a reference set R is used, this indicator will determine the proportion of solutions in an approximation set A, which are weakly dominated by

members of R. Thus, this indicator is defined as:

$$I_{SC} = \frac{|\{\boldsymbol{a} \in A \,|\, \exists \boldsymbol{a}' \in R : \boldsymbol{a}' \preceq \boldsymbol{a}\}|}{|A|}. \tag{4.3.4}$$

The value $I_{SC} = 1$ means that all members of A are weakly dominated by R, so smaller values of I_{SC} are preferable.

Epsilon Indicator

The unary additive ϵ-indicator was suggested by Zitzler et al. [204]. It is based on the concept of additive ϵ-dominance [120] and defined with respect to a reference set R as:

$$I_\epsilon = \inf_{\epsilon \in \mathbb{R}} \{\forall \mathbf{a}' \in R \; \exists \mathbf{a} \in A : \; \mathbf{a} \preceq_\epsilon \mathbf{a}'\}. \tag{4.3.5}$$

The ϵ-indicator gives the minimum factor ϵ such that any objective vector in R is ϵ-dominated by at least one objective vector in A. Smaller values of I_ϵ are preferable.

4.3.2 Indicators Evaluating Diversity

In the following, a few quality indicators are presented that are used to measure how the second goal of multiobjective optimization is achieved. These indicators quantitatively assess the diversity among obtained nondominated solutions.

Overall Nondominated Vector Generation Indicator

This indicator introduced by Veldhuizen [179] measures the total number of nondominated vectors found during MOEA execution and is defined as:

$$I_{ONVG} = |A|. \tag{4.3.6}$$

Although counting the number of nondominated solutions gives some feeling for how effective a multiobjective optimizer is in generating desired solutions, it does not reflect on how far from $\mathcal{PF}$ the vectors in A are neither how well the solutions in A are distributed. There are a few special cases where this indicator can be used to assess the quality of an approximation set, for example, if the entire search space contains only nondominated points.

Spacing Indicator

The spacing indicator was suggested by Schott [157], and it is used to calculate a relative distance measure between consecutive solutions in an approximation set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$. This indicator is defined as:

$$I_{SS} = \sqrt{\frac{\sum_{n=1}^{|A|}(d_n - \overline{d})^2}{|A|}} \tag{4.3.7}$$

where

$$d_n = \min_{k \in A \wedge k \neq n} \sum_{i=1}^{m} |a_i^n - a_i^k|$$

and $\overline{d}$ is the mean value of the above distance measure

$$\overline{d} = \sum_{n=1}^{|A|} \frac{d_n}{|A|}.$$

The distance measure is the minimum value of the sum of the absolute difference in objective function values between the n-th solution and any other solution in the approximation set. This distance measure is different from the minimum Euclidean distance between two solutions.

Spread Indicator

The spread indicator proposed by Deb et al. [46] measures the extend of spread achieved among the obtained solutions. This indicator is used to calculate the nonuniformity in the distribution of the solutions in an approximation set A. The spread indicator is defined as:

$$I_S = \frac{\sum_{i=1}^{m} d_i^e + \sum_{n=1}^{|A|-1} |d_n - \overline{d}|}{\sum_{i=1}^{m} d_i^e + (|A| - 1)\overline{d}} \tag{4.3.8}$$

where d_n can be any distance measure between neighboring solutions and $\overline{d}$ is the mean value of these distance measures. The Euclidean distance, the sum of the absolute differences in objective values or the crowding distance can be used to calculate d_n. The

parameter d_i^e is the distance between the extreme solutions of $\mathcal{PF}$ and A corresponding to i-th objective function.

In the case, where there is a large variance in d_n, the indicator may be greater than one. However, a good distribution would make all distances d_n equal to $\overline{d}$ and would make $d_i^e = 0\ \forall i \in \{1, \ldots, m\}$. Thus, for the most widely and uniformly distributed set of nondominated solutions, the numerator of I_S would be zero, making the indicator to take a value zero. For any other distribution, the value of the metric would be greater than zero.

In order to use the spread indicator for more then two objectives, Nebro et al. [136] modified the spread indicator defined in (4.3.8) by computing the distance from a given point to its nearest neighbor. On the other hand, Custódio et al. [36] proposed a slightly modified spread indicator, which is defined as follows:

$$I_S = \max_{1 \le i \le m} \left(\frac{d_{0,i} + d_{|A|,i} + \sum_{n=1}^{|A|-1} |d_{n,i} - \overline{d}_i|}{d_{0,i} + d_{|A|,i} + (|A| - 1)\overline{d}_i} \right) \tag{4.3.9}$$

where $\overline{d}_i$, for $i = 1, \ldots, m$, is the average of the distances $d_{n,i},\ n = 1, \ldots, |A| - 1$. The authors report very similar results of their indicator compared to the original spread indicator defined in [46] for $m = 2$.

Maximum Spread Indicator

The maximum spread indicator defined by Zitzler [**?**] measures the length of the diagonal of a hyperbox formed by the extreme function values observed in an approximation set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$. This indicator is defined as:

$$I_{MS} = \sqrt{\sum_{i=1}^{m} \left(\max_{1 \le n \le |A|} a_i^n - \min_{1 \le n \le |A|} a_i^n \right)^2}. \tag{4.3.10}$$

A normalized version of the maximum spread indicator can be defined as follows:

$$I_{\overline{MS}} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(\frac{\max_{1 \le n \le |A|} a_i^n - \min_{1 \le n \le |A|} a_i^n}{a_i^{\max} - a_i^{\min}} \right)^2}. \tag{4.3.11}$$

where $a_i^{\max}$ and $a_i^{\min}$ are the maximum and minimum values of the i-th objective in the approximation set A. In this way, if the maximum spread indicator is one, a widely spread set of solutions is obtained.

Chi-Square-Like Deviation Indicator

The chi-square-like deviation indicator proposed by Srinivas and Deb [164] is used to estimate the distributing ability of multiobjective optimization algorithms. A neighborhood parameter ϵ is used to count the number of solutions, n_k, within each chosen Pareto optimal solution $k \in \mathcal{PF}$. This indicator measures the deviation between this counted set of numbers with an ideal set in the chi-square sense and is defined as:

$$I_{CD} = \sqrt{\sum_{k=1}^{|\mathcal{PF}|+1} \left(\frac{n_k - \overline{n}_k}{\sigma_k} \right)^2}. \tag{4.3.12}$$

Since a uniform distribution is considered as an ideal distribution of solutions in an approximation set, the number of solutions allocated in the niche of each chosen Pareto optimal solution should be $\overline{n}_k = |A|/|\mathcal{PF}|$. The parameter $\sigma_k^2 = \overline{n}_k(1 - \overline{n}_k/|A|)$ is suggested for $k = 1, \ldots, |\mathcal{PF}|$. However, the index $k = |\mathcal{PF}| + 1$ represents all solutions which do not reside in the ϵ-neighborhood of any of the chosen Pareto optimal solutions. For this index, the ideal number of solutions and its variance are calculated as follows:

$$\overline{n}_{|\mathcal{PF}|+1} = 0, \quad \sigma^2_{|\mathcal{PF}|+1} = \sum_{k=1}^{|\mathcal{PF}|} \sigma_k^2 = |A| \left(1 - \frac{1}{|\mathcal{PF}|} \right).$$

Number of Distinct Choices Indicator

The number of distinct choices indicator proposed by Wu and Azarm [188] counts the number of distinct solutions in a given approximation set A. Each objective is divided into $1/\mu$ equal divisions, where μ is a user specified number. In this way, the objective space is divided into $(1/\mu)^m$ unique, equal-sized m-dimensional hypercubes. Each of the hypercubes $h_i,\ i = 1, \ldots, (1/\mu)^m$ is referred as the indifference region. Within this region any two solutions $\boldsymbol{a}_k, \boldsymbol{a}_l \in A$ are considered similar to one another. The quantity $NT(h_i)$

will be equal to 1 as long as there is at least one solution in A falling into the indifference region defined by the hypercube h_i. In general, $NT(h_i)$ can be stated as:

$$NT(h_i) = \begin{cases} 1 & \exists \boldsymbol{a}_k \in A \quad \boldsymbol{a}_k \in h_i \\ 0 & \forall \boldsymbol{a}_k \in A \quad \boldsymbol{a}_k \notin h_i. \end{cases}$$

So, this indicator, which is the number of distinct choices for a specified value of μ, is defined as:

$$I_{NDC} = \sum_{i=1}^{(1/\mu)^m} NT(h_i). \tag{4.3.13}$$

Thus, an approximation set A with a higher value of the quantity I_{NDC} is preferable.

Cluster Indicator

The number of distinct choices indicator indicates the number of distinct solutions in an approximation set A, however, it can not properly interpret how clustered these solutions are. Thus, the cluster indicator [188] can be used to estimate how clustered solutions are. This indicator is defined as:

$$I_{CL} = \frac{|A|}{I_{NDC}}. \tag{4.3.14}$$

The value of $I_{CL} = 1$ would indicate that every solution in $|A|$ is distinct, in all other cases $I_{CL} > 1$. Thus, the higher the value of the cluster indicator is, the more clustered an approximation set A is, and hence the less preferred.

Diversity Indicator

The diversity indicator suggested by Li et al. [124] evaluates the spread of solutions in an approximation set A considering not only the standard deviation of distances between solutions in A but also the maximal spread. This indicator is defined as:

$$I_{DV} = \frac{\sum_{i=1}^{m}(f_i^{(max)} - f_i^{(min)})}{1 + \sqrt{\frac{1}{|A|}\sum_{k=1}^{|A|}(d_k - \overline{d})^2}} \tag{4.3.15}$$

where d_k is the Euclidean distance between the k-th solution and its closest neighbor, and $\overline{d}$ is the mean value of all d_k, $f_i^{(max)}$ and $f_i^{(min)}$ represents the maximum and minimum objective function value of i-th objective. The larger value of I_{DV} means a better diversity of solutions in A.

4.3.3 Indicators Evaluating Convergence and Diversity

In the following, some quality indicators where both tasks are evaluated in a combined sense are presented. Such indicators can provide a qualitative measure of convergence as well as diversity. Nevertheless, they are often used along with one of the above indicators to get a better overall evaluation.

Distance from Reference Set Indicator

Czyzak and Jaszkiewicz [37] suggested the distance from reference set indicator, which measures the mean distance over the points in a reference set $R = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{|R|}\}$ of the nearest point in an approximation set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$. This indicator is defined as:

$$I_D = \frac{\sum\limits_{\boldsymbol{r}\in R} \min\limits_{\boldsymbol{a}\in A}\{\ d(\boldsymbol{r}, \boldsymbol{a})\}}{|R|} \tag{4.3.16}$$

where $d(\boldsymbol{r}, \boldsymbol{a}) = \max\limits_{1\leq i\leq m}\{\lambda_i(r_i - a_i)\}$ and $\lambda_i = 1/s_i$, with s_i being the range of objective i in set R. Smaller values of I_D are preferable.

R Indicators

Hansen and Jaszkiewicz [79] proposed R indicators to assess and compare approximation sets on the basis of a set of utility functions. A parameterized utility function u_λ with a corresponding set of parameters $\Lambda = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{|\Lambda|}\}$ is used to represent the decision maker's preferences. In [79], there were proposed several ways to transform such a family of utility functions into a quality indicator. In particular, I_{R2} and I_{R3} indicators can be defined as follows:

$$I_{R2} = \frac{\sum\limits_{\boldsymbol{\lambda}\in\Lambda} u^*(\boldsymbol{\lambda}, A) - u^*(\boldsymbol{\lambda}, R)}{|\Lambda|} \tag{4.3.17}$$

$$I_{R3} = \frac{\sum_{\boldsymbol{\lambda}\in\Lambda} [u^*(\boldsymbol{\lambda}, R) - u^*(\boldsymbol{\lambda}, A)]/u^*(\boldsymbol{\lambda}, R)}{|\Lambda|} \tag{4.3.18}$$

where A is an approximation set, R is a reference set, $u^*(\boldsymbol{\lambda}, A) = \max_{\boldsymbol{a}\in A} u_\lambda(\boldsymbol{a})$ is the maximum value reached by the utility function u_λ with parameter $\boldsymbol{\lambda}$ on A, and $u^*(\boldsymbol{\lambda}, R) = \max_{\boldsymbol{r}\in R} u_\lambda(\boldsymbol{r})$ is the maximum value reached by the utility function u_λ with parameter $\boldsymbol{\lambda}$ on R.

There are various possibilities with respect to the choice of the parameterized utility function u_λ. For instance, u_λ can represent the weighted Chebyshev function:

$$u_\lambda(\boldsymbol{z}) = -\max_{1\le i\le m} \lambda_i |z_i^* - z_i|$$

where $\boldsymbol{z}$ is a vector in A or R, $\lambda = (\lambda_1, \ldots, \lambda_m)^{\mathrm{T}} \in \Lambda$ is a particular weight vector, and $\boldsymbol{z}^* = (z_1^*, \ldots, z_m^*)^{\mathrm{T}}$ is a reference point. Smaller values of I_{R2} and I_{R3} are preferable.

Hypervolume Indicator

The hypervolume indicator or S-metric was introduced by Zitzler and Thiele [202]. Since then, the hypervolume indicator, which measures the volume of the dominated portion of the objective space relative to a reference set, has become a popular quality indicator used to compare the performance of multiobjective algorithms. It can be defined as the Lebesgue measure Λ of the union of hypercuboids in the objective space:

$$I_H = \Lambda\left(\bigcup_{\boldsymbol{a}\in A\wedge\boldsymbol{r}\in R} \{f_1(\boldsymbol{a}'), \ldots, f_m(\boldsymbol{a}') : \boldsymbol{a} \prec \boldsymbol{a}' \prec \boldsymbol{r}\}\right) \tag{4.3.19}$$

where $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$ is an approximation set, and $R = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_{|R|}\}$ is an appropriately chosen reference set. The higher value of I_H, the more preferable an approximation set is.

In some cases, it is useful to consider the hypervolume difference to a reference set R. This indicator is defined as:

$$I_H^- = I_H(R) - I_H(A) \tag{4.3.20}$$

where $I_H(A)$ is as defined in (4.3.19), and $I_H(R)$ is a volume of objective space bounded by R. Here, smaller values correspond to the higher quality in contrast to the original hypervolume I_H.

Inverted Generational Distance Indicator

Inverted generational distance (IGD) indicator proposed by Bosman and Thierens [17] computes the average distance from each point in the Pareto front $\mathcal{PF}$ to the closest solution in an approximation set A. This indicator can be referred as the distance from the Pareto front to an approximation set, and is defined as:

$$I_{IGD} = \frac{\sum_{\boldsymbol{a}' \in \mathcal{PF}} d(\boldsymbol{a}', A)}{|\mathcal{PF}|} \tag{4.3.21}$$

where $d(\boldsymbol{a}', A)$ is the minimum Euclidean distance between $\boldsymbol{a}'$ and the points in A. A smaller value of I_{IGD} corresponds to the higher quality of an approximation set A.

Averaged Hausdorff Distance Indicator

Schütze et al. [158] proposed the quality indicator I_{Δ_p}, which measures the averaged Hausdorff distance of an approximation set $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|A|}\}$ to the Pareto front $\mathcal{PF} = \{\boldsymbol{a}'_1, \ldots, \boldsymbol{a}'_{|\mathcal{PF}|}\}$. This indicator is based on slightly modified I_{GD} and I_{IGD}, and is defined as:

$$I_{\Delta_p} = \max(I_{GD_p}, I_{IGD_p}) \tag{4.3.22}$$

where

$$I_{GD_p} = \left(\frac{\sum_{i=1}^{|A|} d_i^p}{|A|} \right)^{\frac{1}{p}}, \tag{4.3.23}$$

d_i is the Euclidean distance from $\boldsymbol{a}_i$ to its nearest member of $\mathcal{PF}$, and

$$I_{IGD_p} = \left(\frac{\sum_{i=i}^{|\mathcal{PF}|} \tilde{d}_i^p}{|\mathcal{PF}|} \right)^{\frac{1}{p}} \tag{4.3.24}$$

$\tilde{d}_i$ is the Euclidean distance from $\boldsymbol{a}'_i$, to its nearest member of A. In [158], the authors argue that a slight modification of both indicators I_{GD} and I_{IGD} (i.e., by using the power mean of the considered distances) leads to more "fair" indicators. As larger archive sizes (for I_{GD_p}),

respectively, finer discretizations of the Pareto front (for I_{IGD_p}) do not automatically lead to "better" approximations as in their original definitions.

Thus, the resulting I_{Δ_p}, which defines an averaged Hausdorff distance (d_H) for $p < \infty$ and coincides with (d_H) for $p = \infty$, offers better metric properties than its components. Smaller values of I_{Δ_p} are preferable.

4.4 Statistical Comparison

In the case of stochastic optimizers, the relation between performance of different algorithms is not fixed, but described by a corresponding density function. Accordingly, every statement about the performance of any stochastic algorithm is probabilistic in nature. Thus, it is crucial to apply proper statistical procedures in order to get statistically sound conclusions about the performance of different algorithms. In multiobjective optimization, two basic approaches for the statistical comparison can be distinguished. One is the attainment function approach, which models the outcome of a multiobjective optimizer as a probability density function in the objective space. Another approach is indicator-based, which quantitatively measures the quality of each outcome and performs the statistical analysis on the corresponding distribution of performance values.

4.4.1 Attainment Surface

Fonseca and Fleming [71] introduced a statistical comparison method called an empirical attainment surface. Figure 4.2 shows two approximation sets returned by two algorithms A and B. The lines joining the points (solid for A and dashed for B) indicate the attainment surfaces. An attainment surface divides objective space into two regions: one containing vectors which are dominated by the approximation set produced by the algorithm, and another one that contains vectors that dominate elements of the approximation set produced by the algorithm. As shown in Figure 4.2, a number of sampling lines ($L_1, \ldots, L_9$) can be drawn from the reference point, which intersect the attainment surfaces, across the

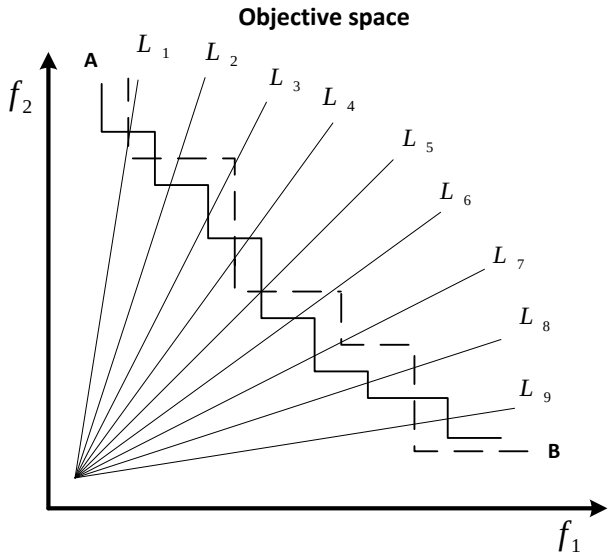


Figure 4.2: Attainment surfaces and lines of intersection.

full range of the Pareto front. For a given sampling line, the intersection of an algorithm closer to the reference point is the winner. Thus, algorithm A is winner for line L_8 and B is winner for line L_9.

If multiobjective optimizers run r times, each algorithm returns r attainment surfaces, one from each run. Having these r attainment surfaces, some from algorithm A and some from algorithm B, a single sampling line yields r points of intersection, one for each surface. These intersections form a univariate distribution, therefore, standard non-parametric statistical procedures can be performed to determine whether or not the intersections for one of the algorithms occur closer to the origin with some statistical significance. In this way, the results of such analysis will yield a percentage of the surface in which algorithm A outperforms algorithm B with statistical significance, and a percentage when algorithm B outperforms algorithm A.

In addition, the concept of attainment surface can be used to visualize the outcomes of multiple runs of an optimizer. For instance, one may be interested in plotting all the goals that have been attained in $k\%$ of the runs. Thus, for each line the points of intersection with an attainment surface produced by each run of the algorithm are calculated. These

points define a sample distribution, where the desired moments of the distribution can be estimated and the resulting surface plotted. For details, see [72].

4.4.2 Statistical Testing

As it was mentioned at the beginning of this chapter, quality indicators perform a transformation by mapping from a sample of approximation sets generated from multiple runs of an optimizer to a set of real numbers. The ultimate purpose of generating the samples and applying the transformations is to describe and make inferences about the underlying approximation set distributions of the optimizers, thereby enabling to compare their performance.

The standard statistical inference that is made considering the optimizer's underlying approximation set distributions is based on hypothesis testing. Hypothesis testing is the use of statistics to determine the probability that a given hypothesis is true. There are two types of statistical hypotheses: (i) the null hypothesis, denoted by H_0, usually can be stated as: samples A and B are drawn from the same distribution, (ii) the alternative hypothesis, denoted by H_1, is a statement that directly contradicts a null hypothesis. An inferential statistical test is used to compute the p-value, which is compared with the significance level α. The null hypothesis is rejected if the p-value is less than or equal to the significance level.

The definition of the alternative hypothesis usually takes one of two forms. On the one hand, it can be stated as: sample A comes from a better distribution than sample B. On the other hand, it can be defined as: sample A and sample B are from different distributions. The former inferential test is referred as a one-tailed test while the latter is called a two-tailed test [111]. A one-tailed test is more powerful than a two-tailed test, in the sense that for a given α, it rejects the null hypothesis more readily in cases where it is actually true.

Some inferential statistical tests make assumptions about the parameters of the population distribution. Thus, it is assumed that a distribution from which the data is drawn

is a known distribution such as the normal distribution. Such known distributions are completely defined by their parameters (the mean and standard deviation), and these tests are thus termed parametric statistical tests. Parametric tests are powerful because even quite small difference between the means of two normal distributions can be detected accurately, and should be used unless assumptions have been clearly not met. The examples of parametric statistical tests that can be used for the performance comparison are: ANOVA and t-test.

However, in order to to use the parametric tests, it is necessary to check the following conditions [74]:

- Independence: In statistics, two events are independent when the fact that one occurs does not modify the probability of the other one occurring.
- Normality: An observation is normal when its behavior follows a normal or Gauss distribution with a certain value of average μ and variance σ^2.
- Heteroscedasticity: This property indicates the existence of a violation of the hypothesis of equality of variances.

While it is obvious that the condition of independence of the events is satisfied in the case of independent runs of the algorithm, the other two conditions can be checked by carrying out the following statistical tests: (i) Kolmogorov-Smirnov, Shapiro-Wilk or D'Agostino-Pearson test to perform the normality analysis, (ii) Levene's test can be used for checking whether or not samples present homogeneity of variances (homoscedasticity).

However, when the above stated conditions are not satisfied nonparametric statistical tests should be used, which make fewer and less stringent assumptions than their parametric counterparts. Nonparametric methods are often more powerful than parametric methods if the assumptions for the parametric model cannot be met. Two main types of nonparametric tests can be distinguished: rank tests and permutation tests. Rank tests pool the values from several samples and convert them into ranks by sorting them, and then employ tables describing the limited number of ways in which ranks can be dis-

tributed to determine the probability that the samples come from the same population. In turn, permutation tests use the original values without converting them to ranks but estimate the likelihood that samples come from the same population explicitly by Monte Carlo simulation. Rank tests are the less powerful but are also less sensitive to outliers and computationally cheap. Permutation tests are more powerful because information is not thrown away, and they are also better when there are many tied values in the sample, however they can be expensive to compute for large samples. The following nonparametric statistical tests can be used: Kruskal-Wallis test, Friedman's test, Mann-Whitney test, Wilcoxon test, and Fisher's test.

More about the use of statistical tests can be found in [24, 57, 74, 111].

4.4.3 Performance Profiles

Another statistical tool that can be used for the performance comparison using quality indicators are the so-called performance profiles. Performance profiles were first introduced by Dolan e Moré [59] to compare the performance of deterministic algorithms over a set of distinct optimization problems, and they can be extended to the context of stochastic algorithms with some adaptations [33]. Performance profiles are depicted by a plot of a cumulative distribution function $\rho(\tau)$ representing a performance ratio for the different solvers. Performance profiles provide a good visualization and easiness of making inferences about the performance of the algorithms. A brief description of the performance profiles follows.

Let $\mathcal{P}$ and $\mathcal{S}$ be the set of problems and the set of solvers in comparison, respectively, and let $m_{p,s}$ be the performance metric required to solve problem $p \in \mathcal{P}$ by solver $s \in \mathcal{S}$. The comparison is based on performance ratios defined by

$$r_{p,s} = \frac{m_{p,s}}{\min\{m_{p,s} : s \in \mathcal{S}\}}$$

and the overall assessment of the performance of a particular solver s is given by

$$\rho_s(\tau) = \frac{1}{\text{total number of problems}}\{\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}\}.$$

For $\tau = 1$, $\rho_s(\tau)$ gives the probability that the solver s will win over the others in the set. Thus, for $\tau = 1$, the uppermost curve shows the algorithm with the highest percentage of problems with the best metric value. However, for large values of τ, the $\rho_s(\tau)$ measures the solver robustness. Overall, the higher the ρ_s values, the better the solver is. Also, for solver s that performs the best on a problem p, $r_{p,s} = 1$. If $r_{p,s} = 2$, it means that the m-fold improvement by solver s on problem p is twice the best value found by another solver on the same problem p.

In the original study [59] the authors used the computing time, $t_{p,s}$, required to solve a problem p by a solver s to evaluate the performance of the solvers. However, they suggested that other measures can be used instead. So, in the context of multiobjective optimization the quality indicators can be used as a performance metric. However, it should be noted that the concept of performance profiles requires minimization of a performance metric. Generally, performance profiles can be defined representing a statistic computed for a given quality indicator value obtained in several runs (e.g., minimum, median, etc.).

4.5 Summary

The performance comparison concerning multiobjective optimization algorithms is more difficult than in the case of single-objective optimization. As it has been stressed throughout the thesis, there are two goals in multiobjective optimization: (i) find a set of solutions which is as close as possible to the true Pareto optimal front, and (ii) find a set of solutions which is as diverse as possible. Thus, when comparing different multiobjective algorithms the ability of achieving these two goals simultaneously by the algorithms should be assessed. To perform a comparative study, it is commonly accepted practice to run MO algorithms on a set of artificially constructed tests problems and compare their outcomes. The outcome of a given run approximates the Pareto front for a given test problem, and is called approximation set. In turn, approximation sets can be compared in several ways.

One way is to use quality indicators to quantitatively assess how good the produced approximation set is. The use of quality indicators is an attractive approach because that

maps each approximation set to a real number, which allows to apply statistical methods to conclude which algorithm provides statistically better approximation sets. However, the statements need to be seen in the context of the preference represented by the considered indicator. Some indicators measure the convergence to the Pareto front (such as generational distance indicator I_{GD}, epsilon indicator I_ϵ, etc.), other measure the diversity among the obtained solutions (such as spacing indicator I_{SS}, spread indicator I_S, etc.). On the other hand, some indicators can be used to measure both the convergence and diversity (for instance, hypervolume indicator I_H and inverted generational distance indicator I_{GD}). Therefore, it is preferable to use several different quality indicators in order to provide quality assessments with respect to different preferences (say one indicator for evaluating convergence and another one for evaluating diversity). Some authors recommend to use the combination of Pareto compliant indicators.

Another approach to statistical evaluation is based on attainment functions, which give for each vector in the objective space the probability that it is attained. An empirical attainment function summarizes the outcomes of multiple runs of one algorithm by calculating for each vector the relative frequency that it has been attained. Comparing to the approach based on the quality indicators, there is only little loss of information caused by the transformation. Accordingly, the statistical comparison of two empirical attainment functions can reveal a lot of information concerning where the outcomes of two algorithms differ.

Additionally, whenever it is possible (the case of two or three objectives), it is always useful to plot the outcomes produced by the algorithms. As for the most of test problems the Pareto optimal front is know, a simple visual comparison can provide unbiased insights about algorithm's ability to converge to the Pareto front and to maintain the adequate diversity among solutions.

Part II

Algorithms

Chapter 5

Hybrid Genetic Pattern Search Augmented Lagrangian Algorithm

5.1 Introduction

The previous part of this thesis presents the necessary background and reviews existing approaches for solving multiobjective optimization problems. However, it is well-known that overall successful and efficient general solvers do not exist. Statements about the optimization algorithms' performance must be qualified with regard to the "no free lunch" theorems for optimization [187]. Its simplest interpretation is that a general-purpose universal optimization strategy is theoretically impossible, and the only way one strategy can outperform another is if it is specialized to the specific problem under consideration [84]. Therefore, the development of new optimization methods is essential for successful solving difficult problems emerging in the modern world.

The chapters presented in this part of the thesis introduce new optimization techniques based on evolutionary algorithms for solving multiobjective optimization problems. In particular, this chapter suggests an approach based on a hybridization of an augmented Lagrangian with a genetic algorithm and a pattern search method for solving constrained single-objective optimization problems. After combining with the NBI and NC methods,

the proposed approach is extended to deal with multiobjective optimization problems. Since an efficient algorithm is required for solving the NBI and NC constrained subproblems, the proposed framework relying on augmented Lagrangian penalty can be a viable alternative for solving such problems.

5.2 HGPSAL

In this section, a particular case of multiobjective optimization problem ($m = 1$) is under consideration:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, p \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, q \\ & \boldsymbol{x} \in \Omega \end{aligned} \tag{5.2.1}$$

where $\boldsymbol{x}$ is an n dimensional decision vector and $\Omega \subset \mathbb{R}^n$ ($\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$), $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ is the vector of inequality constraints and $\boldsymbol{h}(\boldsymbol{x}) = 0$ is the vector of equality constraints. In the following, the algorithm developed for solving a problem shown in (5.2.1), which is termed hybrid genetic pattern search augmented Lagrangian algorithm (HGPSAL), is described.

5.2.1 Augmented Lagrangian

An augmented Lagrangian technique solves a sequence of simple subproblems where the objective function penalizes all or some of the constraint violations. This objective function is an augmented Lagrangian that depends on a penalty parameter, as well as on the multiplier vectors, and works like a penalty function. Using the ideas in [12, 31, 122], the herein implemented augmented Lagrangian function is

$$\Phi(\boldsymbol{x}; \boldsymbol{\lambda}, \boldsymbol{\delta}, \mu) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{h}(\boldsymbol{x}) + \frac{1}{2\mu} \|\boldsymbol{h}(\boldsymbol{x})\|_2^2 + \frac{\mu}{2} \sum_{i=1}^{p} \left(\max \left\{ 0, \delta_i + \frac{g_i(\boldsymbol{x})}{\mu} \right\}^2 - \delta_i^2 \right)$$

where μ is a positive penalty parameter, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_q)^{\mathrm{T}}$ and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_p)^{\mathrm{T}}$ are the Lagrange multiplier vectors associated with the equality and inequality constraints,

respectively. Function Φ aims to penalize solutions that violate only the equality and inequality constraints. The simple bounds $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ are not integrated into the penalty terms. Hence, the corresponding subproblem is formulated as:

$$\begin{aligned} \text{minimize} \quad & \Phi(\boldsymbol{x}; \boldsymbol{\lambda}^{(j)}, \boldsymbol{\delta}^{(j)}, \mu^{(j)}) \\ \text{subject to} \quad & \boldsymbol{x} \in \Omega \end{aligned} \tag{5.2.2}$$

To simplify the notation, from now $\Phi^{(j)}(x)$ is used instead of $\Phi(\boldsymbol{x}; \boldsymbol{\lambda}^{(j)}, \boldsymbol{\delta}^{(j)}, \mu^{(j)})$, $\|.\|$ represents the Euclidean norm, and $\boldsymbol{v}_+ = \max\{0, \boldsymbol{v}\}$. The solution of (5.2.2) for each set of fixed $\boldsymbol{\lambda}^{(j)}$, $\boldsymbol{\delta}^{(j)}$ and $\mu^{(j)}$, gives an approximation to the solution of (5.2.1). This approximation is denoted by $\boldsymbol{x}^{(j+1)}$. Here the superscript (j) is the counter of the outer iterative process. To ensure global convergence, the penalty parameter must be driven to zero as the Lagrange multipliers estimates must have reasonable behavior. Thus, as $j \to \infty$ and $\mu^{(j)} \to 0$, the solutions of the subproblems (5.2.2) converge to the solution of (5.2.1). For details, see [12].

The Lagrange multipliers $\boldsymbol{\lambda}^{(j)}$ and $\boldsymbol{\delta}^{(j)}$ are estimated in the iterative process according to proper first-order updating formulae. The traditional augmented Lagrangian methods are locally convergent if the subproblems (5.2.2) are solved according to a certain tolerance, herein reported by $\varepsilon^{(j)}$, for sufficiently small values of the penalty parameter. The outline of the general augmented Lagrangian algorithm for solving problem (5.2.1) is presented in Algorithm 1.

The stopping criterion is based on an error function, $E(\boldsymbol{x}, \boldsymbol{\delta})$, that requires rather small feasibility and complementarity levels from the computed approximation, where

$$E(\boldsymbol{x}, \boldsymbol{\delta}) = \max\left\{ \frac{\|\boldsymbol{h}(\boldsymbol{x})\|_\infty}{1 + \|\boldsymbol{x}\|}, \ \frac{\max\left\{ \|\boldsymbol{g}(\boldsymbol{x})_+\|_\infty, \max\limits_i \delta_i |g_i(\boldsymbol{x})| \right\}}{1 + \|\boldsymbol{\delta}\|} \right\}.$$

If the algorithm finds an approximation $\boldsymbol{x}$ such that

$$E(\boldsymbol{x}, \boldsymbol{\delta}) \leq \eta^* \ \text{ and } \ \varepsilon \leq \varepsilon^*$$

for small and positive constants η^* and ε^*, then the algorithm stops; otherwise, the algorithm runs until a maximum of (outer) iterations, $j_{\max}$, is reached. The tolerance ε varies

Algorithm 1 HGPSAL

input: $\lambda_{\min} < \lambda_{\max}$, $\delta_{\max} > 0$, $0 < \gamma < 1$, $\mu_{\min} \ll 1$, $\eta^* \ll 1$, $\epsilon^* \ll 1$, $\lambda_i^{(0)} \in [\lambda_{\min}, \lambda_{\max}]$, $i = 1, \ldots, q$, $\delta_i^{(0)} \in [0, \delta_{\max}]$, $i = 1, \ldots, p$, $\mu^{(0)} > 0$, $\eta^{(0)} > 0$, $\pi < 1$;
Compute $\varepsilon^{(0)}$;
Randomly generate a point $\boldsymbol{x}^{(0)} \in \Omega$;
$j \leftarrow 0$;
repeat
    For a certain tolerance $\varepsilon^{(j)}$ find an approximate minimizer $\boldsymbol{x}^{(j+1)}$ to
    the subproblem (5.2.2) using Algorithm 2;
    $\delta_i^{(j+1)} \leftarrow \max\left\{0, \min\left\{\delta_i^{(j)} + \frac{g_i(\boldsymbol{x}^{(j+1)})}{\mu^{(j)}}, \delta_{\max}\right\}\right\},\ \forall i \in \{1, \ldots, p\}$;
    if $E(\boldsymbol{x}^{(j+1)}, \boldsymbol{\delta}^{(j+1)}) \leq \eta^{(j)}$ **then**
        $\lambda_i^{(j+1)} \leftarrow \lambda_i^{(j)} + \max\left\{\lambda_{\min}, \min\left\{\frac{h_i(\boldsymbol{x}^{(j+1)})}{\mu^{(j)}}, \lambda_{\max}\right\}\right\},\ \forall i \in \{1, \ldots, q\}$;
        $\mu^{(j+1)} \leftarrow \mu^{(j)}$;
    else
        $\lambda_i^{(j+1)} \leftarrow \lambda_i^{(j)}$;
        $\mu^{(j+1)} \leftarrow \max\left\{\mu_{\min}, \gamma\mu^{(j)}\right\}$;
    end if
    $\eta^{(j+1)} \leftarrow \pi\eta^{(j)}$;
    Compute $\varepsilon^{(j+1)}$;
    $j \leftarrow j + 1$;
until the stopping criterion is met
output: $\boldsymbol{x}_{\min}$, $f_{\min}$;

with the Lagrange multipliers and penalty parameter values according to

$$\varepsilon = \tau\left(1 + \|\boldsymbol{\lambda}\| + \|\boldsymbol{\delta}\| + (\mu)^{-1}\right)^{-1},\ \tau > 0.$$

Note that a decreasing sequence of μ values will yield a decreasing sequence of ε values forcing more and more accurate solutions to the subproblems (5.2.2). As shown in Algorithm 1, the updating of the Lagrange multipliers relies on safeguards to maintain the multiplier vectors bounded throughout the process. The sequence of penalty parameters should also be maintained far away from zero so that solving the subproblem (5.2.2) is an

Algorithm 2 GAPS

1: input $\boldsymbol{x}^{(j)} \in \Omega$;
2: $Y^{(j)} \leftarrow \texttt{GA}(\boldsymbol{x}^{(j)})$;
3: $X^{(j+1)} \leftarrow \texttt{HJ}(Y^{(j)})$;
4: $\boldsymbol{x}^{(j+1)} \leftarrow$ compute the best point of $X^{(j+1)}$;
5: output: $\boldsymbol{x}^{(j+1)} \in \Omega$;

easy task. The main differences between augmented Lagrangian algorithms are located on the framework used to find an approximate global solution, for a defined tolerance $\varepsilon^{(j)}$, to the subproblem (5.2.2). The herein proposed technique for solving (5.2.2) uses a population based algorithm, known as genetic algorithm, followed by a local search procedure. The outline of the hybrid genetic algorithm pattern search (GAPS) approach is shown in Algorithm 2.

Since the genetic algorithm is a population based method, $Y^{(j)}$ is a set of $\boldsymbol{y}^{(j)}$ points, with best fitness found by the algorithm. Details concerning each step of the algorithm are presented below.

5.2.2 Genetic Algorithm

In Algorithm 2, genetic algorithm is adopted with real representation of the search parameters to deal with continuous problems instead of the traditional binary representation. The outline of GA is given by Algorithm 3.

GAs start from a population of points P of size s. Each point of the population $\boldsymbol{z}_l$, for $l = 1, \ldots s$, is an n dimensional vector. A fitness function is defined as evaluation function in order to compare the points of the population and to apply a stochastic selection that guarantees that better points are more likely to be selected. The fitness function corresponds to the objective function of the subproblem (5.2.2), i.e., $\Phi^{(j)}(\boldsymbol{x})$. A tournament selection was considered, i.e., tournaments are played between two points and the best point (with lower fitness value) is chosen for the mating pool.

New points in the search space are generated by the application of genetic operators

Algorithm 3 GA

input: $\boldsymbol{x}^{(j)}$, s, e, p_c, p_m, η_c, η_m;
$\boldsymbol{z}_1 \leftarrow \boldsymbol{x}^{(j)}$ and randomly generate $\boldsymbol{z}_l \in \Omega$, for $l = 2, \ldots, s$ *(Initialization of P)*;
$k \leftarrow 0$;
repeat
    Compute $\Phi^{(j)}(\boldsymbol{z}_l)$, for $l = 1, \ldots, s$ *(Fitness Evaluation)*;
    Select by tournaments $s - e$ points from P *(Selection)*;
    Apply SBX crossover with probability p_c *(Crossover)*;
    Apply polynomial mutation with probability p_m *(Mutation)*;
    Replace the worst $s - e$ points of P *(Elitism)*;
    $Y^{(j)} \leftarrow P^{(k)}$;
    $k \leftarrow k + 1$;
until the stopping criterion is met
output: $Y^{(j)}$;

(crossover and mutation) to the selected points from the population. Elitism is implemented by maintaining a given number, e, of the best points in the population.

Crossover combines two points in order to generate new ones. Simulated binary crossover [43], which simulates the working principle of single-point crossover operator for binary strings, is implemented. Two points, $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$, are randomly selected from the mating pool and, with probability p_c, two new points, $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ are generated as:

$$w_i^1 = 0.5\left((1+\beta_i)z_i^1 + (1-\beta_i)z_i^2\right)$$
$$w_i^2 = 0.5\left((1-\beta_i)z_i^1 + (1+\beta_i)z_i^2\right)$$

for $i = 1, \ldots, n$. The values of β_i are obtained from the following distribution:

$$\beta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_c+1}} & \text{if} \quad r_i \leq 0.5 \\ \left(\frac{1}{2(1-r_i)}\right)^{\frac{1}{\eta_c+1}} & \text{if} \quad r_i > 0.5 \end{cases}$$

where $r_i \sim \mathbb{U}(0,1)$ and $\eta_c > 0$ is an external parameter of the distribution. This procedure is repeated until the number of generated points is equal to the number of points in the mating pool.

Since crossover cannot exclusively assure the exploration of new regions of the search space, polynomial mutation is applied with a probability p_m to the points produced by the crossover operator. This operator guarantees that the probability of creating a new point $\boldsymbol{t}^l$ closer to the previous one $\boldsymbol{w}^l$ $(l = 1, \ldots, s)$ is more than the probability of creating one away from it. It can be expressed by:

$$t_i^l = w_i^l + (u_i - l_i)\iota_i$$

for $i = 1, \ldots, n$. The values of ι_i are given by:

$$\iota_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} - 1 & \text{if} \quad r_i < 0.5 \\ 1 - (2(1-r_i))^{\frac{1}{\eta_m+1}} & \text{if} \quad r_i \geq 0.5 \end{cases}$$

where $r_i \sim \mathbb{U}(0,1)$ and $\eta_m > 0$ is an external parameter of the distribution.

This procedure ends when $|\Phi^{(j)}(\boldsymbol{z}_{\text{best}}^{(k)}) - \Phi^{(j)}(\boldsymbol{z}_{\text{best}}^{(k-k_\Delta)})| \leq \varepsilon^j$, where $\Phi^{(j)}(\boldsymbol{z}_{\text{best}}^{(k)})$ is the fitness value of the best point in the population, at iteration k, and k_Δ is a parameter that defines a periodicity for testing the criterion. However, if the stopping criterion is not met in $k_{\max}$ iterations, the procedure is terminated and the best point in the population is returned.

5.2.3 Hooke and Jeeves

A pattern search method is a derivative-free method that performs, at each iteration k, a series of exploratory moves around a current approximation, $\boldsymbol{z}^{(k)}$, in order to find a new approximation $\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \Delta^{(k)}\boldsymbol{s}^{(k)}$, with a lower fitness value. In the following, k is used for the iteration counter of this inner iterative process. For $k = 0$, the initial approximation to start the search is $\boldsymbol{z}^{(0)} = \boldsymbol{y}^{(j)}$ $(\boldsymbol{y}^{(j)} \in Y^{(j)})$, see Algorithm 2. The scalar $\Delta^{(k)}$ represents the step length and the vector $\boldsymbol{s}^{(k)}$ determines the direction of the step. The exploratory moves to produce $\Delta^{(k)}\boldsymbol{s}^{(k)}$, the updating of $\Delta^{(k)}$ and $\boldsymbol{s}^{(k)}$ defines a particular pattern search method. Their choices are crucial to the success of the algorithm. If $\Phi^{(j)}(\boldsymbol{z}^{k+1}) < \Phi^{(j)}(\boldsymbol{z}^{(k)})$ then the iteration is considered successful; otherwise it is unsuccessful. If an iteration is

successful, the step length is not modified, while in an unsuccessful iteration $\Delta^{(k)}$ is reduced. For details, see [121, 175].

In the proposed approach, $\Delta^{(k)}\boldsymbol{s}^{(k)}$ is computed by the Hooke and Jeeves (HJ) search method [86]. This algorithm differs from the traditional coordinate search since it performs two types of moves: the exploratory move and the pattern move. An exploratory move is a coordinate search - a search along the coordinate axes - around a selected approximation, using a step length of $\Delta^{(k)}$. A pattern move is a promising direction that is defined by $\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)}$ when the previous iteration was successful and $\boldsymbol{z}^{(k)}$ was accepted as the new approximation. A new trial approximation is then defined as $\boldsymbol{z}^{(k)} + (\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)})$ and an exploratory move is then carried out around this trial point. If this search is successful, the new approximation is accepted as $\boldsymbol{z}^{(k+1)}$. For details, see [86, 121]. This HJ iterative procedure terminates providing a new set of approximations $X^{(j+1)}$ to the problem (5.2.1), $\boldsymbol{x}^{(j+1)} \leftarrow \boldsymbol{z}^{(k+1)}$, when the stopping condition is met, $\Delta^{(k)} \leq \varepsilon^{(j)}$. However, if this condition cannot be satisfied in $k_{\max}$ iterations, then the procedure is stopped with the last available approximation.

The inner iterative process must return a feasible approximation. While using the genetic algorithm and the Hooke and Jeeves method, any computed approximation $\boldsymbol{x}$ that does not lie in the feasible set Ω is projected component by component as follows: $\forall i \in \{1, \ldots, n\} : x_i = \min\{\max\{x_i, l_i\}, u_i\}$.

5.2.4 Performance Assessment

HGPSAL is implemented in the MATLAB® programming language and tested on a set of 24 benchmark problems. This set of difficult constrained problems with very distinct properties is taken from [125]. The characteristics of these problems are summarized in Table 5.1 that indicates the type of objective function, the number of decision variables (n), the number of inequality constraints (p), the number of equality constraints (q), the number of active constraints at the optimum (n_{act}) and the optimal value known (f_{global}).

All parameters of the HGPSAL algorithm are kept constant for all problems. No effort

Prob.		Type of $f(\boldsymbol{x})$	n	p	q	n_{act}	f_{global}
g01	min	quadratic	13	9	0	6	-15.00000
g02	max	nonlinear	20	2	0	1	0.803619
g03	max	polynomial	10	0	1	1	1.000000
g04	min	quadratic	5	6	0	2	-30665.54
g05	min	cubic	4	2	3	3	5126.498
g06	min	cubic	2	2	0	2	-6961.814
g07	min	quadratic	10	8	0	6	24.30621
g08	max	nonlinear	2	2	0	0	0.095825
g09	min	polynomial	7	4	0	2	680.6301
g10	min	linear	8	6	0	6	7049.248
g11	min	quadratic	2	0	1	1	0.750000
g12	max	quadratic	3	1	0	0	1.000000
g13	min	nonlinear	5	0	3	3	0.053945
g14	min	nonlinear	10	0	3	3	-47.76488
g15	max	quadratic	3	0	2	2	-961.7150
g16	min	nonlinear	5	38	0	4	-1.905155
g17	min	nonlinear	6	0	4	4	8853.539
g18	max	quadratic	9	13	0	6	0.866025
g19	min	nonlinear	15	5	0	0	32.65559
g20	min	linear	24	6	14	16	0.204979
g21	min	linear	7	1	5	6	193.7245
g22	min	linear	22	1	19	19	236.4309
g23	min	linear	9	2	4	6	-400.0551
g24	max	linear	2	2	0	2	5.508013

Table 5.1: Test problems.

$\lambda_{\min}$	$\lambda_{\max}$	$\delta_{\max}$	μ^0	$\mu_{\min}$	γ	ε^*	η^*	$\lambda_i^0, \forall i$	$\delta_i^0, \forall i$	η^0	$j_{\max}$
-10^{12}	10^{12}	10^{12}	1	10^{-12}	0.5	10^{-12}	10^{-6}	1	1	1	300

Table 5.2: Augmented Lagrangian parameters.

$k_{\max}$	s	e	p_c	η_c	p_m	η_m	ϵ	Δ_g
200	20	2	0.9	20	$1/n$	20	10^{-8}	$2n$

Table 5.3: Genetic algorithm parameters.

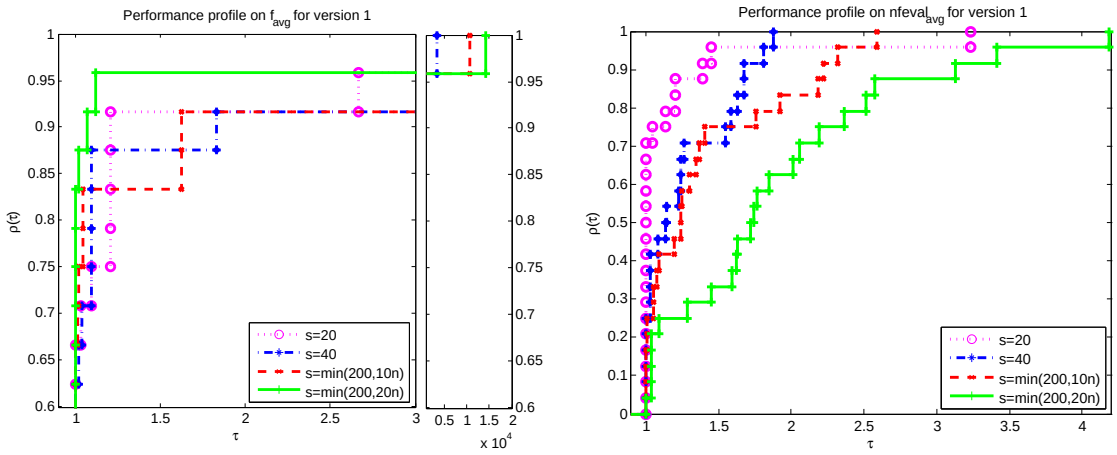
is made in finding the best parameter setting for each problem. Table 5.2 shows the parameters of the augmented Lagrangian used in all experiments. The genetic algorithm parameters are listed in Table 5.3. The maximum number of function evaluations for HGPSAL is set to $200,000$.

Hybridization Schemes

Four different hybridization schemes are proposed. They differ in the number of points of the population that are selected to be improved in the local search. The tested alternatives are: just one point, the best point of the population; the 10% best points; the 25% best points; and the 50% best points. For simplicity, the hybridization schemes are denoted as follows:

- version 1 that improves the best population point found by GA with HJ;
- version 2 that improves the best 10% population points found by GA with HJ;
- version 3 that improves the best 25% population points found by GA with HJ;
- version 4 that improves the best 50% population points found by GA with HJ.

Figure 5.1 allows to analyze the effect of the population size s on the algorithm performance. The profiles in Figure 5.1(a), based on the metric f_{avg}, the central tendency of the

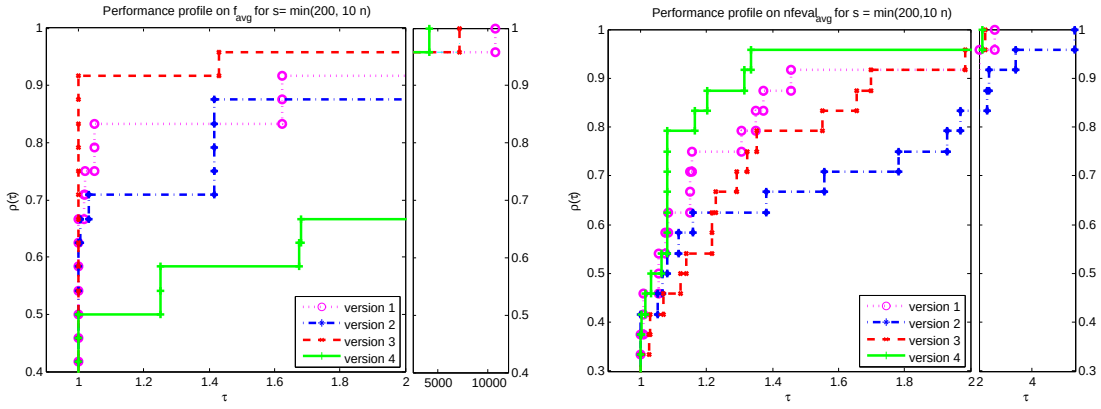


(a) f_{avg} (b) $nfeval_{\text{avg}}$

Figure 5.1: Performance profiles on f_{avg} and $nfeval_{\text{avg}}$ for version 1.

best solutions found over the 10 runs, show that $s = \min(200, 20n)$ outperforms the other executions with different (and smaller) population sizes in comparison. The execution with this population size gives the best average solution in about 84% of the tested problems. It is followed by the execution with $s = \min(200, 10n)$, which attains the best average solution in 75% of the problems. The execution with the smallest population size, $s = 20$, is able to reach the least average function value in about 62% of the problems. From Figure 5.1(a), it can be concluded that the larger the population size the better the accuracy of the average solution is. However, as expected, the size of the population affects the computational effort of the algorithm. Figure 5.1(b) displays the performance profiles of the average number of function evaluations $nfeval_{\text{avg}}$ computed over the 10 runs. A large population size s requires consequently a large number of function evaluations. The difference here in the percentage of solved problems with least number of function evaluations between the executions $s = 20$ (with 70% of better values) and $s = \min(200, 20n)$ (with 5% of better values) is around 65%. Thus, a compromise seems crucial. A population size of $s = \min(200, 10n)$ seems appropriate for the remaining numerical experiments.

Figure 5.2 allows to analyze the four proposed hybridization schemes. The figure shows performance profiles on the central tendency of the best solutions, f_{avg}, and the average



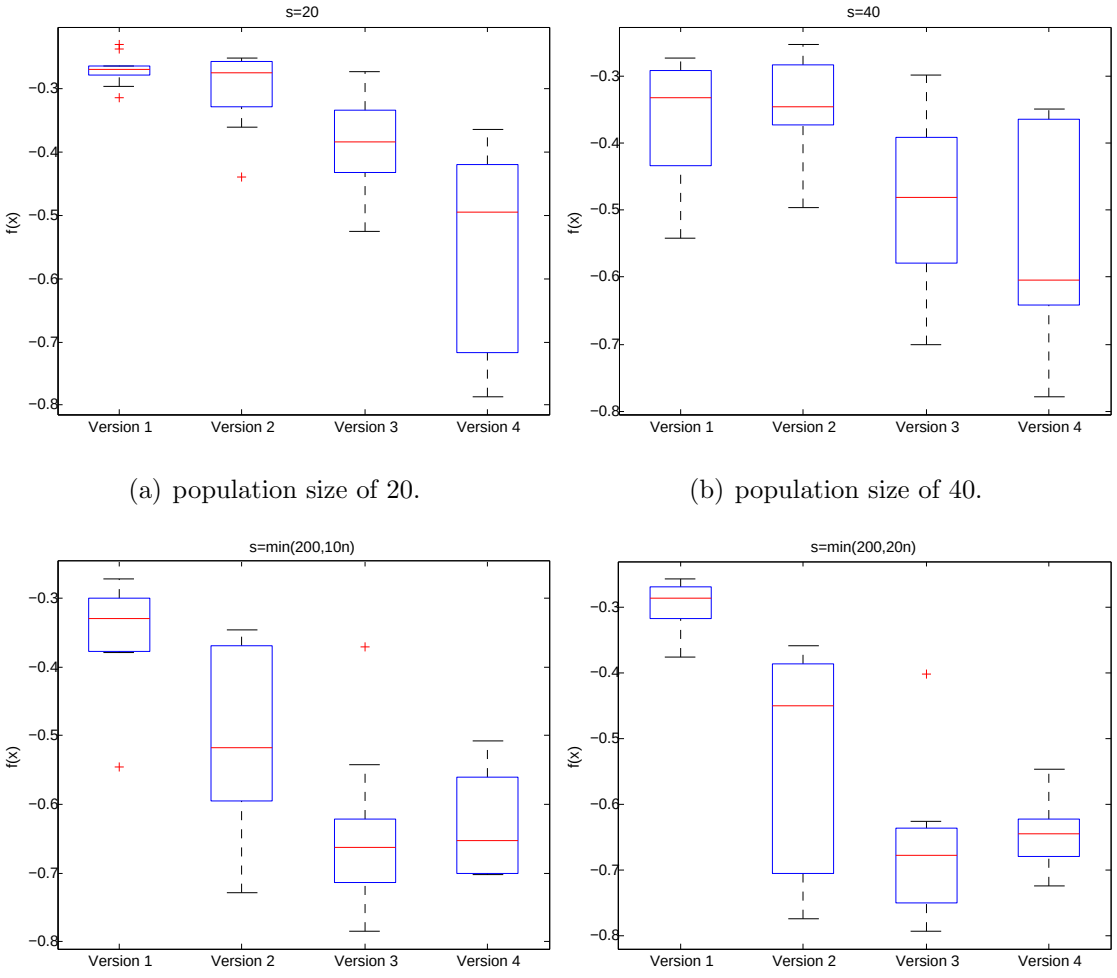
(a) f_{avg} (b) $nfeval_{\text{avg}}$

Figure 5.2: Performance profiles on f_{avg} and $nfeval_{\text{avg}}$ for $s = \min(200, 10n)$.

number of function evaluations, $nfeval_{\text{avg}}$, over 10 runs.

In Figure 5.2(a), it can observed that version 3 outperforms the other versions, attaining the best average solution in more than 90% of the problems. Recall that this version improves the best 25% best points of the population obtained by the GA algorithm with the HJ local search. The worst performance is obtained by version 4, meaning that it is self defeating to use an excessive number of points of the population to improve by local search. As to the $nfeval_{\text{avg}}$ (Figure 5.2(b)), version 3 obtains the lower number of average function evaluations in about 33% of the problems. Despite version 2 and version 4 are less expensive in this item (42% of efficiency), it seems a good compromise to choose version 3 as the one that gives the best results.

Figure 5.3 shows the boxplot representations of the comparative performance of the HGPSAL versions with different population sizes for the problem g02. It should be noted that this problem is difficult and, in general, solvers fail to achieve an accurate approximation to the global optimum. The figure indicates the distribution of the approximations obtained in 10 runs (the worst, upper quartile, median, lower quartile, and best approximations are depicted). In Figures 5.3(a) and 5.3(b), it can be observed that the best performance is obtained with the version 4, which presents the best median. However, for larger population sizes, version 3 outperforms version 4 (Figures 5.3(c) and 5.3(d)).



(a) population size of 20.

(b) population size of 40.

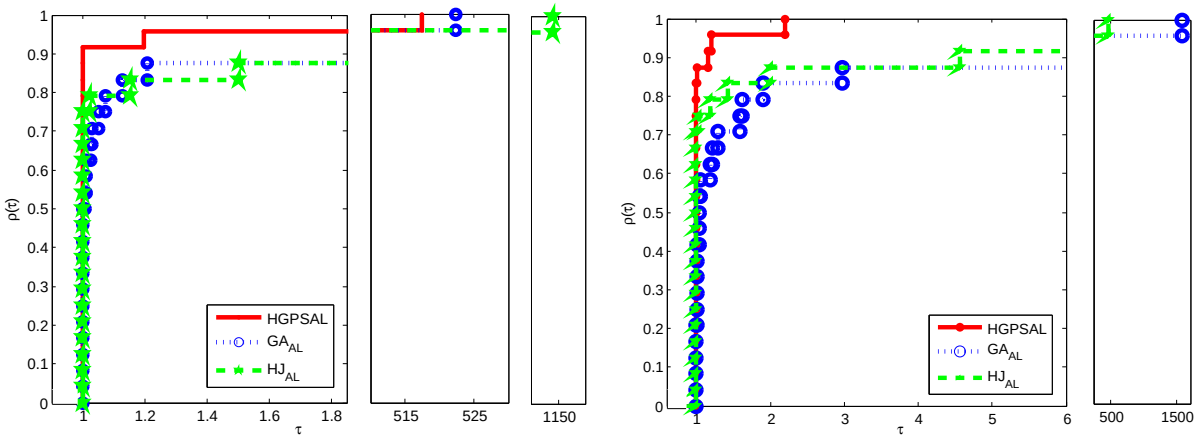
(c) population size of $\min(200, 10n)$.

(d) population size of $\min(200, 20n)$.

Figure 5.3: Boxplots for different population sizes (problem g02).

Performance Comparison

In order to compare the performance of HGPSAL with non-hybrid approaches using augmented Lagrangian technique, two different algorithms are considered to find a minimizer $x^{(j+1)}$ in Algorithm 1 (line 6). The first algorithm uses GA to find a solution for the subproblem (5.2.2), it is referred as GA_{AL}. The second algorithm uses HJ to find a solution for the subproblem (5.2.2), it is referred as HJ_{AL}. For each algorithm, 10 independent runs are performed setting the maximum number of function evaluations equal to $200,000$. A



(a) f_{best} (b) f_{median}

Figure 5.4: Performance profiles on f_{best} and f_{median}.

population size of $s = \min(200, 20n)$ is used for HGPSAL and GA_{AL}. For HJ_{AL}, the initial solution is randomly generated within the bounds. Furthermore, version 2 of hybridization is used for HGPSAL. The remaining parameter settings for Lagrangian and GA are presented in Tables 5.2 and 5.3, respectively.

Figure 5.4 presents performance profiles on the best and the median function values obtained over 10 runs. From Figure 5.4(a), it can be seen that HGPSAL outperforms other algorithms in terms of accuracy achieving the best function values in 58% of the problems. The second best is GA_{AL} providing the best values in 38% of the problems. In terms of robustness, the best algorithms is also HGPSAL followed by GA_{AL}.

Figure 5.4(b) shows performance profile on the median function values. In terms of accuracy, the best algorithm is HGPSAL, which provides the best median values in 67% of the problems, followed by GA_{AL}, which achieves the best median values in 25% of the problems. In terms of robustness, the best algorithms is HGPSAL followed by HJ_{AL}.

From the above discussion, it can be concluded that the proposed hybridization of GA and HJ with the augmented Lagrangian technique performs better than any of the considered algorithms separately. Thus, HGPSAL seems to be a valid optimizer for constrained single-objective optimization.

5.3 MO-HGPSAL

So far in this chapter the optimization of single-objective function has been discussed. The proposed HGPSAL shows ability to solve challenging constrained single-objective problems. However, the focus of the present thesis is on solving multiobjective optimization problems. Therefore, in the following the application of HGPSAL to multiobjective optimization is discussed.

Chapter 3 discusses some classical methods for multiobjective optimization. Almost all of them convert a given MOP into several SOPs. In particular, the normal boundary intersection method and the normal constraint method transform unconstrained problems into constrained ones. Such methods require to use efficient single-objective optimization algorithms able to handle all types of constraints. In turn, as it is shown early in this chapter, the proposed HGPSAL is an efficient algorithm for solving constrained optimization problems. Thus, it seems a promising idea to apply HGPSAL to solve multiobjective optimization problems using scalarizing methods, which require to solve a set of constrained single-objective optimization problems.

In the following, two approaches are considered to extend HGPSAL to solve MOPs. First approach uses the normal boundary intersection method to perform scalarization, it is referred as MO-HGPSAL$_{\text{NBI}}$. Second approach uses the normal constraint method to perform scalarization, it is referred as MO-HGPSAL$_{\text{NC}}$. The outline of a general approach to multiobjective optimization using HGPSAL (MO-HGPSAL) is shown in Algorithm 4.

Since both the NBI and NC methods require a knowledge of anchor (critical) points needed to construct a hyperplane $\boldsymbol{\Phi}\boldsymbol{\beta}$, the optimization process starts by initializing $\boldsymbol{\Phi} = (\boldsymbol{f}(\boldsymbol{x}^{1*}), \boldsymbol{f}(\boldsymbol{x}^{2*}), \ldots, \boldsymbol{f}(\boldsymbol{x}^{m*}))$ and generating a set of evenly distributed points $\boldsymbol{\beta} \in B$, where $\forall i \in \{1, \ldots, m\} : 0 \leq \beta_i \leq 1 \wedge \sum_{i=1}^{m} \beta_i = 1$. After that, $\forall i \in \{1, \ldots, |B|\} : \boldsymbol{\beta}_i \in B$, the corresponding subproblem is solved using HGPSAL (line 4). A solution found after solving each subproblem becomes an initial point for solving subsequent subproblem using HGPSAL (line 5). All found solutions are stored in the archive A (line 6), which is eventually returned as an approximation to the Pareto set for a given MOP.

Algorithm 4 MO-HGPSAL

1: initialize: $\boldsymbol{x}^{(0)}$, $\boldsymbol{\beta} \in B$, $\Phi = (\boldsymbol{f}(\boldsymbol{x}^{1*}), \ldots, \boldsymbol{f}(\boldsymbol{x}^{m*}))$;
2: $A \leftarrow \{\}$;
3: **for** $\boldsymbol{\beta} \in B$ **do**
4: $(\boldsymbol{x}^*, \boldsymbol{f}(\boldsymbol{x}^*)) \leftarrow \text{HGPSAL}(\boldsymbol{x}^{(0)})$ on f_{agr};
5: $\boldsymbol{x}^{(0)} \leftarrow \boldsymbol{x}^*$;
6: $A \leftarrow A \cup \{\boldsymbol{x}^*\}$;
7: **end for**
8: output: A;

5.3.1 Performance Assessment

In the following, the performance of MO-HGPSAL$_{\text{NBI}}$ and MO-HGPSAL$_{\text{NC}}$ is studied on the continuous ZDT test suite. All problems are adopted with 30 decision variables. For each algorithm, 10 independent runs are performed on each problem, running for 100,000 function evaluations. The number of evenly distributed points used to construct the hyperplane is 100 ($B = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{100}\}$). This means that 100 constrained single-objective optimizations must be solved using HGPSAL.

Parameter settings adopted for HGPSAL are as follows. The version 1 of hybridization is used. The maximum number of function evaluations for HGPSAL is set to 1,000. GA is run for 10 generations with a population of 5 individuals. In this case, GA performs like a micro-genetic algorithm. The maximum number of function evaluations used by Hooke and Jeeves method is 50. The remaining parameter settings for HGPSAL are the same as shown in Tables 5.2 and 5.3.

Table 5.4 presents the median values of IGD for the Pareto set approximations obtained by MO-HGPSAL$_{\text{NBI}}$ and MO-HGPSAL$_{\text{NC}}$ after 10 runs. The best values are marked bold. The last column in the table indicates the results of the Wilcoxon rank-sum test performed at the significance level of $\alpha = 0.05$, where "+" indicates that there is a significant difference between two algorithms while "−" indicates that there is no significant difference between two algorithms.

	MO-HGPSAL$_{\text{NBI}}$	MO-HGPSAL$_{\text{NC}}$	significance
ZDT1	0.0375	**0.0095**	+
ZDT2	**0.0138**	0.0249	+
ZDT3	**0.0388**	0.0522	+
ZDT4	0.0879	**0.0107**	+
ZDT6	0.2394	**0.2393**	−

Table 5.4: Median values of IGD.

The results presented in Table 5.4 show that MO-HGPSAL$_{\text{NBI}}$ performs better on the ZDT2,3 test problems, whereas MO-HGPSAL$_{\text{NC}}$ performs better on the ZDT1,4 test problems. Additionally, MO-HGPSAL$_{\text{NC}}$ provides the lower median value of IGD on ZDT6, however, no significant difference is observed between the two algorithms on this problem.

Figure 5.5 depicts performance profiles on the best and median values of IGD. Analyzing the presented plots, it can be concluded that both algorithms have quite similar performance in terms of accuracy based on the best and median values of IGD. However, in terms of robustness, MO-HGPSAL$_{\text{NC}}$ outperforms MO-HGPSAL$_{\text{NBI}}$ based on both best and median values of IGD. The better performance of MO-HGPSAL$_{\text{NC}}$ in terms of robustness can

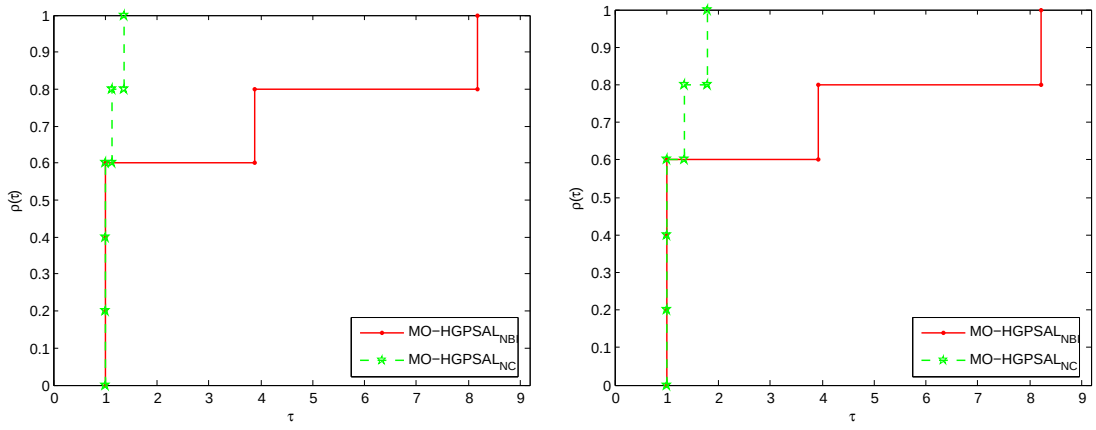
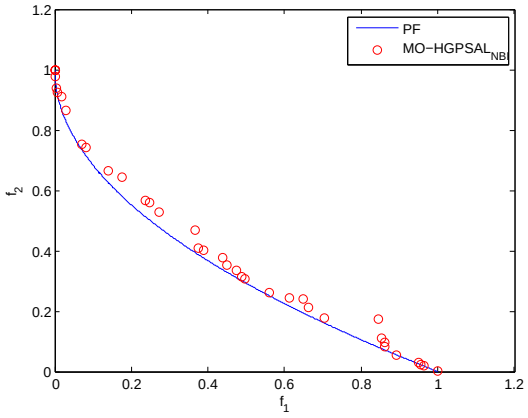
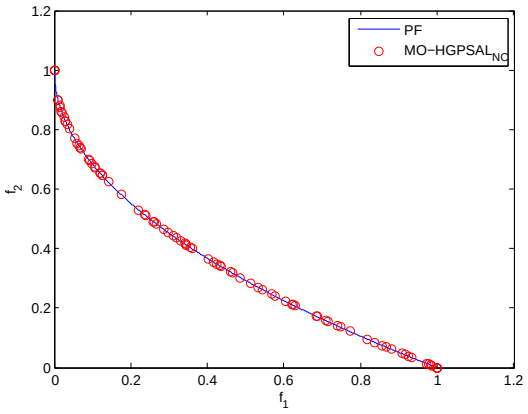


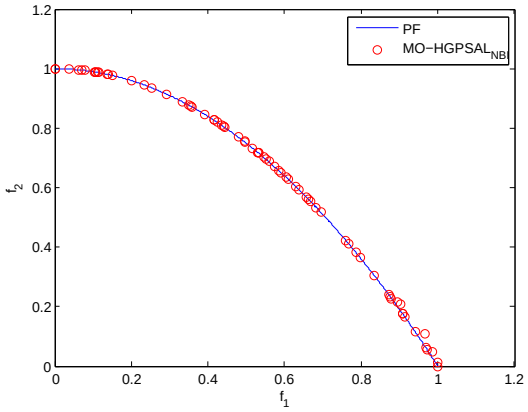
Figure 5.5: Performance profiles on the best (left hand side) and median (right hand side) values of IGD.



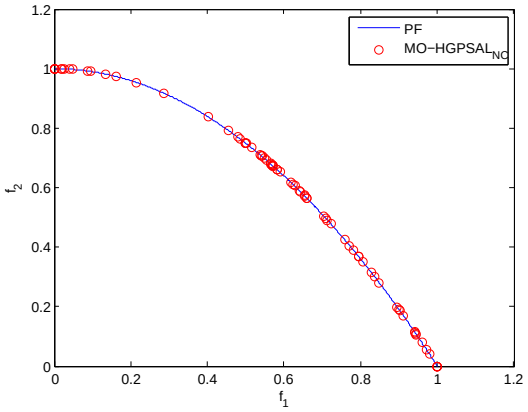
(a) MO-HGPSAL$_{\text{NBI}}$ on ZD1



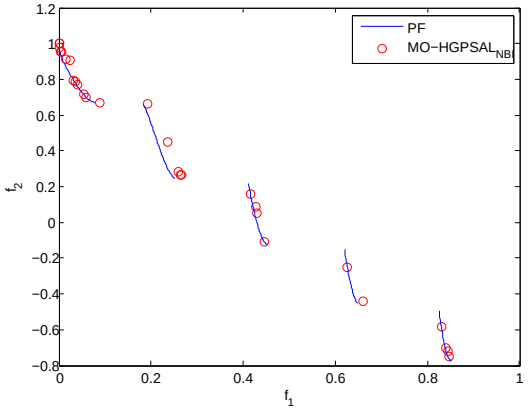
(b) MO-HGPSAL$_{\text{NC}}$ on ZDT1



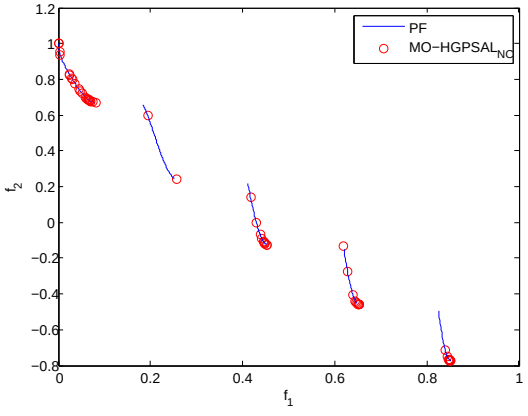
(c) MO-HGPSAL$_{\text{NBI}}$ on ZDT2



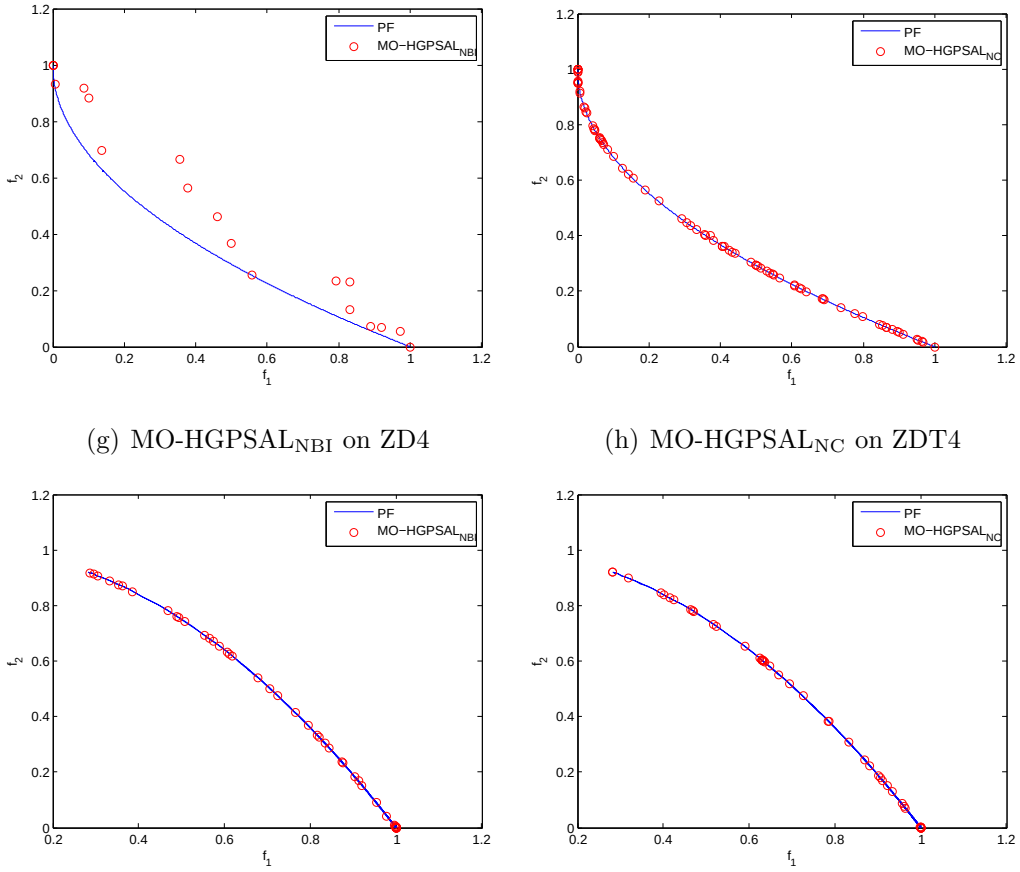
(d) MO-HGPSAL$_{\text{NC}}$ on ZDT2



(e) MO-HGPSAL$_{\text{NBI}}$ on ZDT3



(f) MO-HGPSAL$_{\text{NC}}$ on ZDT3



(g) MO-HGPSAL$_{\text{NBI}}$ on ZD4

(h) MO-HGPSAL$_{\text{NC}}$ on ZDT4

(i) MO-HGPSAL$_{\text{NBI}}$ on ZDT6

(j) MO-HGPSAL$_{\text{NC}}$ on ZDT6

Figure 5.6: Performance of MO-HGPSAL on the ZDT test suite.

be explained by the fact that it seems to be easier for HGPSAL to solve a constrained problem with inequality rather than equality constraints. Therefore, MO-HGPSAL$_{\text{NC}}$ can be considered a good choice to handle multiobjective optimization problems. However, the major limitations of this approach is a high computational cost and the need to provide the critical points of the Pareto front in advance. Since some studies have clearly shown that EMO algorithms are more efficient for solving MOPs than classical methods for multiobjective optimization [161, 181], the performance comparison of MO-HGPSAL with multiobjective evolutionary algorithms is not provided in this chapter. Additionally, the Pareto front approximations with the lowest values of IGD obtained by both algorithms over 10 runs are presented in Figure 5.6.

5.4 Summary

Many optimization problems in science and engineering involve a number of constraints, which the optimal solution must satisfy. Such problems are posed as constrained optimization problems. There are many studies on solving constrained optimization problems. Penalty function methods are among the most popular approaches, because of their simplicity and ease of implementation. In general, the penalty function approaches are applicable to any type of constraints (inequality or equality).

This chapter introduces a new hybrid algorithm within an augmented Lagrangian framework for constrained optimization. In each iteration of the augmented Lagrangian, a set of promising solutions is found by a genetic algorithm. Afterwards, a specified number of these solutions is improved using a pattern search method. It is shown that the hybrid approach exhibits a better performance than each of the methods separately. Moreover, different hybridization schemes are investigated.

The proposed single-objective algorithm is extended to solve MOPs. Two variants of the resulting multiobjective approach are developed. The first variant uses the NBI method for converting MOP into SOP. The second variant employees the NC method for scalarization. The performance of both variants is studied on the continuous ZDT test suite. Although two variants show ability to solve MOPs producing quite similar performance in terms of IGD, there are some disadvantages of MO-HGPSAL compared with EMO algorithms. Since both variants use a hyperplane passing through the critical points of the Pareto front, these points must be known before the search. One way to overcome this limitation is to apply a single-objective optimizer to find a minimum of each objective. However, this does not guarantee that the obtained points will be critical. On the other hand, MO-HGPSAL needs to perform a relatively large number of function evaluations. This may be unacceptable to solve real-world optimization problems with computationally expensive function evaluations. Therefore, the following chapters address MO using EAs designed to overcome these limitations.

Chapter 6

Descent Directions-Guided Multiobjective Algorithm

6.1 Introduction

This chapter introduces a descent directions-guided multiobjective algorithm (DDMOA). DDMOA is a hybrid multiobjective evolutionary algorithm, which borrows the idea of generating new promising solutions from Timmel's method incorporating it in a general framework of EMO algorithms. In order to overcome the limitations of original Timmel's method imposed by the assumptions of differentiability of the objective functions, descent directions for each objective are used instead of the gradients. Thus, new promising solutions are generated by adding to a parent solution a linear combination of descent directions for each objective function. In DDMOA, these descent directions are calculated using a pattern search method. However, it is clear that performing computation of descent directions for each population member and for each objective function is an inefficient way to conduct the search. Therefore, a strategy based on subpopulations is implemented to avoid the direct computation of descent directions for the entire population. This simple mechanism is proposed to avoid computational overhead caused by the local search procedure.

6.2 DDMOA

The main loop of DDMOA is given by Algorithm 5. It is a hybrid evolutionary algorithm with $(\mu + \lambda)$ selection scheme, where in each generation the adaptation of strategy parameters of population members is followed by the successive application of parent selection, mutation, and environmental selection.

In DDMOA, the population consists only of nondominated individuals. Furthermore, an individual a_i in the current population $P^{(g)}$ in generation g is a tuple of the form $[\boldsymbol{x}_i, \delta_i, \boldsymbol{S}_i, \sigma_i]$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the decision vector, $\delta_i > 0$ is the step size used for local search, $\boldsymbol{S}_i \in \mathbb{R}^{n \times m}$ is the search matrix, and $\sigma_i > 0$ is the step size used for reproduction.

DDMOA starts by randomly generating an initial population P^0 of size μ and initializing the set of strategy parameters of each population member. Then, the population is evaluated and all dominated individuals are removed from the population. The evolutionary process is started by computing descent directions for each objective for all individuals in the population in `updateSearchMatrix` procedure. These directions are stored in the search matrix $\boldsymbol{S}$ of the corresponding individual. Additionally, during this procedure all nondominated solutions with respect to the current population found during local search

Algorithm 5 DDMOA

1: $g \leftarrow 0$;
2: `initialize`: $P^{(g)}$;
3: **repeat**
4: $\quad g \leftarrow g + 1$;
5: $\quad (P^{(g)}, Q^{(g)}) \leftarrow$ `updateSearchMatrix`$(P^{(g)})$;
6: $\quad P^{(g)} \leftarrow$ `updateStepSize`$(P^{(g)})$;
7: $\quad R^{(g)} \leftarrow$ `parentSelection`$(P^{(g)})$;
8: $\quad Q^{(g)} \leftarrow$ `mutation`$(R^{(g)}, Q^{(g)})$;
9: $\quad P^{(g)} \leftarrow$ `environmentalSelection`$(P^{(g)} \cup Q^{(g)})$;
10: **until** the stopping criterion is met
11: output: $P^{(g)}$;

are added to the offspring population, which is empty at the beginning of each generation. Next, the step size σ of each individual a is updated in `updateStepSize` procedure. Then, in order to select a pool of parents $R^{(g)}$, `parentSelection` procedure is performed that uses binary tournament selection based on crowding distance. After the pool of parents is selected, a set of offspring is generated and added to the offspring population in `mutation` procedure. Then, a set of fittest individuals is selected from the composed multiset in `environmentalSelection` procedure. This procedure selects nondominated individuals and uses crowding distance to keep the population of bounded size. Finally, if the stopping criterion is met, the evolutionary process terminates, and the algorithm returns its final population, otherwise the algorithm proceeds to the next generation. The following two stopping conditions are used: (i) the maximum number of objective function evaluations is reached, and (ii) $\delta \leq \delta_{tol}$ for all individuals in the population.

In the following, the components of DDMOA are discussed in more detail.

6.2.1 Initialize Procedure

First, the initial population is generated and the set of strategy parameters is initialized $P^0 = \{[\boldsymbol{x}_i^{(0)}, \delta_i^{(0)}, \boldsymbol{S}_i^{(0)}, \sigma_i^{(0)}], \forall i \in \{1, \ldots, \mu\}\}$. The initial values of decision vectors $\forall i \in \{1, \ldots, \mu\} : \boldsymbol{x}_i^{(0)}$ can be generated in multiple ways. They can be either generated randomly such that all the variables are inside the search space or can be uniformly sampled. In turn, DDMOA creates the initial population using Latin hypercube (LH) sampling [126] since it gives a good overall random distribution of the population in the decision space. The set of strategy parameters of each generated individual is initialized taking default values.

After the initial populations is sampled, it is evaluated and only nondominated individuals are retained while all dominated solutions are removed from the population. So usually the number of individuals in P^0 is less than μ.

6.2.2 Update Search Matrix Procedure

The procedure `updateSearchMatrix` is used to update the search matrix of each individual in the current population. Each column of the search matrix stores a descent direction for the corresponding objective function. Thus, to update the search matrix of a given individual, m descent directions need to be computed. The procedure is designed in such a way that the direct computation of descent directions for all individual is avoided, thereby alleviating the computational burden.

Main Loop

The steps of `updateSearchMatrix` procedure are outlined in Algorithm 6.

Algorithm 6 updateSearchMatrix

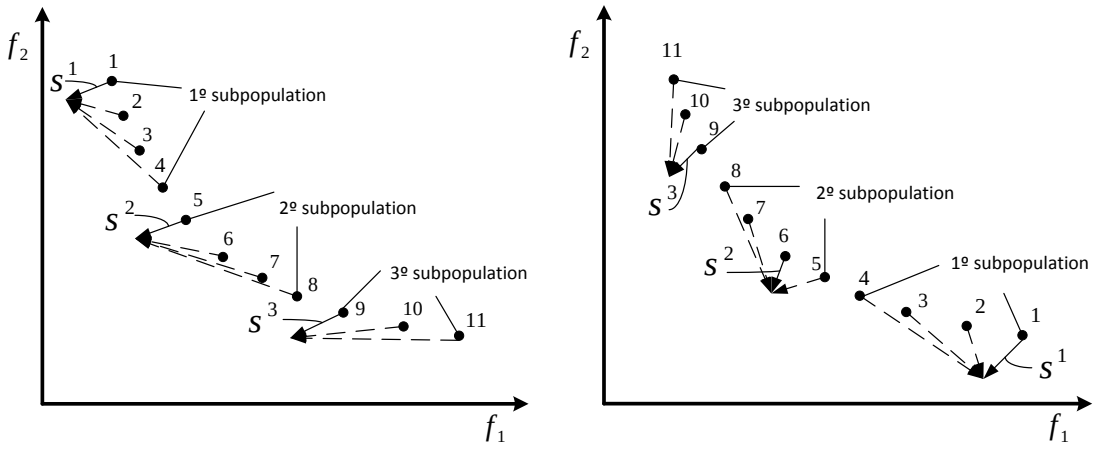
1: input: P;
2: $Q \leftarrow \{\}$;
3: **for** $m = 1 \dots M$ **do** ▷ M - is the number of objectives
4: $i \leftarrow 0$;
5: sort population P in ascending order according to f_m;
6: partition sorted P into α subpopulations:
7: $P = \{p_1, p_2, .., p_\alpha\}$;
8: **for** $k = 1 \dots \alpha$ **do**
9: identify the leader individual $a_{\text{leader}} \in p_k$;
10: $(a_{\text{leader}}, \boldsymbol{s}_{\text{leader}}, Q) \leftarrow \texttt{localSearch}(a_{\text{leader}}, m, Q)$;
11: **for** $j = 1 \dots |p_k|$ **do**
12: $i \leftarrow i + 1$;
13: $\boldsymbol{S}_i(:, m) \leftarrow \boldsymbol{x}_{\text{leader}} - \boldsymbol{x}_i + \boldsymbol{s}_{\text{leader}}$;
14: **end for**
15: **end for**
16: **end for**
17: output: P, Q;

The search matrices of all individuals in the current population are updated by columns. At the beginning, the first column of the search matrix of each individual, which stores a descent direction for the first objective, is updated. Then, the second column of the search matrix of each individual, which stores a descent direction for the second objective, is updated and so on.

For each objective function, the population is sorted in ascending order and partitioned into α equal parts. This way, α subpopulations are defined in order to promote different reference points for the computation of descent directions. It follows that in each subpopulation p_k a leader individual a_{leader} is selected. The leader of the subpopulation is a solution with the lower value of the corresponding objective function among other solutions in the subpopulation and $\delta > \delta_{\text{tol}}$. Thus, if $\delta \leq \delta_{\text{tol}}$ of the solution with the lower value of the corresponding objective then the second best solution is selected as the leader and so on. Thereafter, a descent direction for the m-th objective function is computed for the leader in `localSearch` procedure (line 10). The procedure `localSearch` returns the leader with updated set of the strategy parameters, descent direction for the leader, and offspring population Q, which at the moment contains individuals nondominated with respect to the current population found in `localSearch` procedure. Using descent direction $\boldsymbol{s}_{\text{leader}}$, the m-th column of the search matrix of each individual in the subpopulation is updated:

$$\boldsymbol{S}_i(:, m) = \boldsymbol{x}_{\text{leader}} - \boldsymbol{x}_i + \boldsymbol{s}_{\text{leader}}, \tag{6.2.1}$$

where $\boldsymbol{S}_i(:, m)$ is the m-th column of the search matrix of the i-th individual in the subpopulation, $\boldsymbol{x}_i$ is the decision vector of the i-th individual in the subpopulation, $\boldsymbol{x}_{\text{leader}}$ is subpopulation leader, and $\boldsymbol{s}_{\text{leader}}$ is descent direction for the leader. At the end, all m columns of the search matrix $\boldsymbol{S}$ of each individual in the population are updated. Partitioning the population into different subpopulations and the calculation of descent directions for solutions in the subpopulation based on a descent direction for the leader allow to significantly reduce the number of function evaluations. This way, the direct computation of descent directions for the entire population using `localSearch` procedure, which is computationally expensive, is avoided.



(a) descent directions for f_1 (b) descent directions for f_2

Figure 6.1: Representation of the working principle of `updateSearchMatrix` procedure in the objective space.

The working principle of `updateSearchMatrix` procedure is illustrated in Figure 6.1. The figure presents the population of 11 nondominated solutions at some generation of the algorithm. The population and the corresponding directions are shown in the objective space. Solid arrows denote descent directions found for subpopulation leaders while dashed arrows denote descent directions for the remaining individuals.

In Figure 6.1(a), the population is sorted according to f_1. Three subpopulations are defined ($\alpha = 3$). The first and second subpopulations consist of 4 individuals, whereas the third subpopulation consists of 3 individuals. When the exact division is not possible, the last subpopulation has always less individuals. In the first subpopulation, solution 1 is the leader. Descent direction with respect to f_1 for the leader is S^1. Dashed arrows from points $\{2, 3, 4\}$ show descent directions for the corresponding solutions in the first subpopulation. In the second subpopulation, the leader is solution 5. Descent direction with respect to f_1 for the leader is S^2. Dashed arrows from points $\{6, 7, 8\}$ show descent directions for the corresponding solutions in the second subpopulation. In the third subpopulation, the leader is solution 9. Descent direction with respect to f_1 for the leader is S^3. Dashed

arrows from points $\{10, 11\}$ show descent directions for the corresponding individuals in the third subpopulation.

In Figure 6.1(b), the population is sorted according to f_2. Three subpopulations are defined. In the first subpopulation, solution 1 is the leader. Descent direction with respect to f_2 for the leader is S^1. Dashed arrows from points $\{2, 3, 4\}$ show descent directions for the corresponding solutions in the first subpopulation. In the second subpopulation, the leader is solution 6. Although solution with the lower value of f_2 in the second subpopulation is solution 5, solution 6 is chosen as leader because $\delta_5 < \delta_{\text{tol}}$. This means that the vicinity of this solution has been already explored. Consequently, the next best solution is chosen. Thus, descent direction with respect to f_2 for the leader is S^2. Dashed arrows from points $\{5, 7, 8\}$ show descent directions for the corresponding solutions in the second subpopulation. In the third subpopulation, the leader is solution 9. Descent direction with respect to f_2 for the leader is S^3. Dashed arrows from points $\{10, 11\}$ show descent directions for the corresponding solutions in the third subpopulation.

Local Search Procedure

In line 10 of Algorithm 6, `localSearch` procedure is invoked to compute descent directions for subpopulation leaders. In general, any local search procedure can be used for this purpose. DDMOA uses a simple coordinate search method [175], which is adopted with slight modifications. The steps of `localSearch` procedure are outlined in Algorithm 7.

A randomization procedure is introduced whose idea is similar to that used in factorial design where the experiment is performed in a random order to guarantee that the environment in which treatments are applied is as uniform as possible, and the unknown biases are avoided. Thus, this procedure randomly generates the order of the variables to be tested (line 3), instead of the usual standard order, and the direction along which a given variable is tested (line 4). The order of the variables and the corresponding direction are determined randomly so that the algorithm can escape from a local optimum during the coordinate search. This becomes even more relevant to large-scale problems.

Algorithm 7 localSearch

input: $a = [\boldsymbol{x}, \delta, \boldsymbol{S}, \sigma]$, m, Q;
$\boldsymbol{s} \leftarrow \boldsymbol{0}$, $\rho \leftarrow 0$, $min \leftarrow f_m(\boldsymbol{x})$;
$order \leftarrow$ randomly permute $\{1, 2, \ldots, n\}$;
$direction \leftarrow$ randomly permute $\{1, -1\}$;
while $\delta > \delta_{\text{tol}}$ **do**
    for $i = order$ **do**
        for $d = direction$ **do**
            $\boldsymbol{s}_{\text{trial}} \leftarrow \boldsymbol{s} + d\,\delta\,\boldsymbol{e}_i$; ▷ $\boldsymbol{e}_i$ denotes the standard coordinate vector
            $\boldsymbol{x}_{\text{trial}} \leftarrow \boldsymbol{x} + \boldsymbol{s}_{\text{trial}}$;
            evaluate $\boldsymbol{f}(\boldsymbol{x}_{\text{trial}})$;
            if $\nexists\, \boldsymbol{y} \in P : \boldsymbol{y} \prec \boldsymbol{x}_{\text{trial}}$ **then**
                $Q \leftarrow Q \cup \{\boldsymbol{x}_{\text{trial}}\}$;
            end if
            if $f_m(\boldsymbol{x}_{\text{trial}}) < min$ **then**
                $\rho \leftarrow min - f_m(\boldsymbol{x}_{\text{trial}})$;
                $min \leftarrow f_m(\boldsymbol{x}_{\text{trial}})$;
                $\boldsymbol{s} \leftarrow \boldsymbol{s}_{\text{trial}}$;
                break
            end if
        end for
    end for
    if $\rho > 0$ **then**
        break
    else
        $\delta \leftarrow \delta/2$;
    end if
end while
output: $a = [\boldsymbol{x}, \delta, \boldsymbol{S}, \sigma]$, $\boldsymbol{s}$, Q;

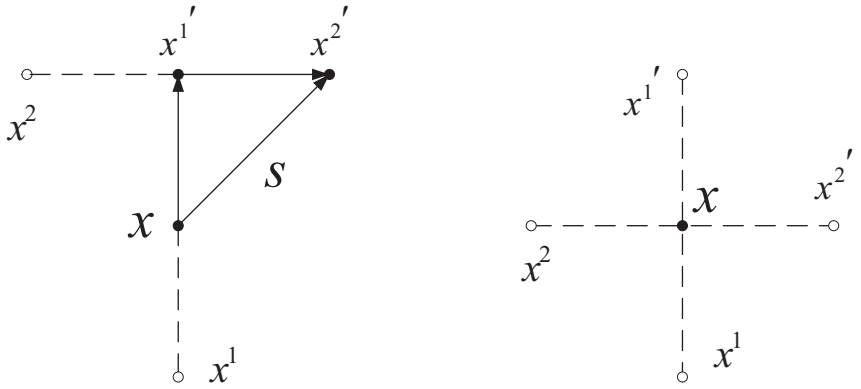


Figure 6.2: Representation of the working principle of `localSearch` procedure.

Figure 6.2 illustrates the working principle of `localSearch` procedure when $n = 2$. The x^i's denote trial points considered during the course of the iteration. Solid circles indicate successful intermediate steps while open circles indicate points at which the function was evaluated but that did not produce further decrease in the value of the objective function. The initial point is denoted by x, and the resulting descent direction is denoted by s. Suppose that randomization results in the following outcome: $order = \{2, 1\}$ and $direction = \{-1, 1\}$. The successful scenario is presented on the left hand side of Figure 6.2. Thus, the first step along the axis x_2 from x to x^1 did not result in a decrease in the objective function. The step in the opposite direction from x to $x^{1'}$ produced a decrease in the value of the objective function. Next, the steps along the axis x_1 are performed. Thus, the first step from $x^{1'}$ to x^2 did not lead to a further decrease in the objective function, while the step in the opposite direction from $x^{1'}$ to $x^{2'}$ gave a decrease in the value of the objective function. The resulting descent direction is $s = x^{2'} - x$. The scenario presented on the right hand side of the figure illustrates the worst case when none of the trial points produced a decrease in the objective function. In this case, the step size δ must be reduced and the above described steps must be performed. If the step size δ is less than tolerance δ_{tol}, the procedure is terminated. Each time a trial solution is added to the offspring population (line 12), the set of strategy parameters is initialized taking default values $(\delta^{(0)}, \boldsymbol{S}^{(0)}, \sigma^{(0)})$.

Algorithm 8 updateStepSize

1: input: P;
2: **for** $i = 1, \ldots, |P|$ **do**
3: $\quad \sigma_i = \max\left\{\frac{\sigma^{(0)}}{g}, \delta_{\text{tol}}\right\}$;
4: **end for**
5: output: P;

6.2.3 Update Step Size Procedure

As it is mentioned earlier, step size σ is associated with each individual in the population. The step size is used to control the mutation strength. In Algorithm 5, step size σ of each individual is updated in `updateStepSize` procedure.

However, there is no common rule to update step size σ, but it must be done carefully to ensure convergence to the Pareto optimal set. DDMOA uses a simple deterministic parameter control rule, where values of σ are determined as a function of time (generation). The outline of `updateStepSize` procedure is given by Algorithm 8.

Using this method step size σ is equal among individuals in the population and decreases during the generations from the initial value $\sigma^{(0)}$ at the first generation and never becomes less then δ_{tol}. Graphically, step size adaptation during the generations is shown in Figure 6.3.

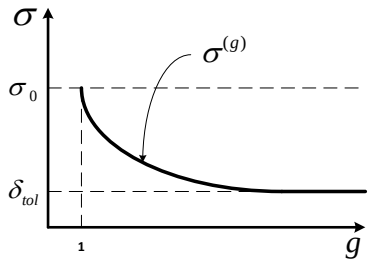


Figure 6.3: Step size adaptation.

6.2.4 Parent Selection Procedure

At every generation, a pool of parents R of size μ is selected in `parentSelection` procedure. The outline of this procedure is shown in Algorithm 9. The procedure `parentSelection` performs binary tournament selection based on crowding distance measured in the objective space. Therefore, individuals with the higher crowding distances have more probability of creating offspring. However, as DDMOA uses only nondominated solutions, the situation where the number of individuals in the population is very small or even equal to one may occur. So if the number of solutions in the population is less or equal to the number of objectives then parents are randomly selected from given individuals. Otherwise tournament selection is performed. This selection process that combines crowding distance with a stochastic selection operator promotes a good spread of solutions in the objective space

Algorithm 9 parentSelection

```
1: input: P;
2: R ← {};
3: while |R| < μ do
4:     if |P| ≤ M then
5:         randomly pick a ∈ P;
6:         R ← R ∪ {a};
7:     else
8:         randomly pick a ∈ P, b ∈ P ∧ a ≠ b;
9:         if I[a] < I[b] then                ▷ I[] denotes crowding distance measure
10:            R ← R ∪ {b};
11:        else
12:            R ← R ∪ {a};
13:        end if
14:    end if
15: end while
16: output: R;
```

Algorithm 10 mutation

1: input: $R = \{[\boldsymbol{x}_i, \delta_i, \boldsymbol{S}_i, \sigma_i] : 1 \leq i \leq |R|\}$, Q;
2: **for** $i = 1, \ldots, |R|$ **do**
3: $\quad \boldsymbol{x}'_i \leftarrow \boldsymbol{x}_i + \sigma_i \boldsymbol{S}_i \boldsymbol{\nu}$;
4: $\quad \boldsymbol{x}'_i \leftarrow \min\{\max\{\boldsymbol{x}'_i, \boldsymbol{l}\}, \boldsymbol{u}\}$;
5: $\quad$ evaluate $\boldsymbol{x}'_i$;
6: $\quad Q \leftarrow Q \cup \{\boldsymbol{x}'_i\}$;
7: **end for**
8: output: Q;

as well as the exploration of new promising areas of the decision space.

6.2.5 Mutation Procedure

After the pool of parents is selected, offspring are generated by mutating the corresponding parent in `mutation` procedure. The outline of this procedure is given by Algorithm 10. Search matrix $\boldsymbol{S}$ and step size σ of the parent are used for the mutation (line 3), where $\boldsymbol{\nu}$ is a column vector of random numbers ($\forall i \in \{1, \ldots, m\} : \nu_i \sim \mathbb{U}(0, 1)$). The mutation is similar to that used in Timmel's method. The only difference is that the gradients of the objective functions are replaced by descent directions, which are stored in search matrix $\boldsymbol{S}$. In order to guarantee that each new solution $\boldsymbol{x}' = [x'_1, \ldots, x'_n]^{\mathrm{T}}$ belongs to Ω, projection is applied to each component of the decision vector (line 4). After offspring $\boldsymbol{x}'$ is repaired, it is evaluated (line 5) and added to the offspring population (line 6), which is initialized in `localSearch` procedure by saving promising solutions found during the computation of descent directions.

6.2.6 Environmental Selection Procedure

The procedure `environmentalSelection` is used to select a set of fittest individuals from the multiset composed of parents and offspring. The outline of `environmentalSelection` procedure is shown in Algorithm 11. Two main steps of `environmentalSelection` proce-

Algorithm 11 environmentalSelection

```
1: input: P' = P ∪ Q;
2: P ← {};
3: for i = 1, ..., |P'| do
4:     for j = 1, ..., |P'| and j ≠ i do
5:         if P'(j) ≺ P'(i) then
6:             break
7:         end if
8:     end for
9:     if j = |P'| then
10:        P ← P ∪ P'(i);
11:    end if
12: end for
13: if |P| > μ then
14:    P ← truncation(P);
15: end if
16: output: P;
```

dure can be distinguished. First, all dominated individuals are removed from the combined population (lines 3-12). Then, if the size of the resulting population P is greater than user-specified population size μ, `truncation` procedure (lines 14) is used to reduce the number of individuals in the population, at the same time, preserving diversity among population members. DDMOA uses `truncation` procedure based on crowding distance, where individuals having higher values of crowding distance measure are selected for the next generation.

6.3 Performance Assessment

This section presents and discusses empirical results obtained by DDMOA. The performance produced by DDMOA is compared with that produced by representative state-of-

the-art multiobjective evolutionary algorithms. The experimental study is performed in two stages. In the first stage, the performance of original DDMOA is studied. In the second stage, more extensive study is performed, where DDMOA is tested with slight modifications.

6.3.1 Preliminary Experiments

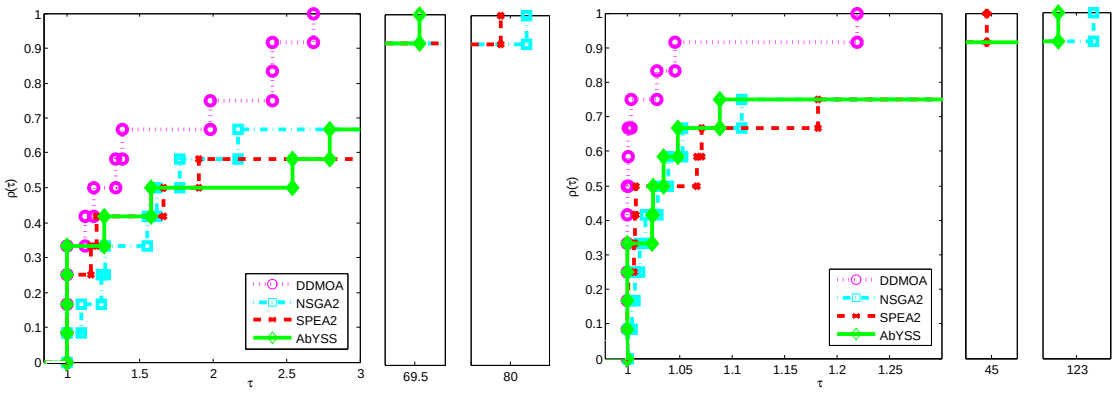
In this study, the results obtained by DDMOA are compared with that produced by NSGA–II [46], SPEA2 [200] and AbYSS [136]. The performance of all algorithms is studied on the ZDT [197] (except for ZDT5, which is a binary problem) and three-objective DTLZ [50] test suites.

Experimental Setup

DDMOA is implemented in the MATLAB® programming language, whereas the other algorithms are used within the jMetal framework [62]. Problems ZDT1-3 and 6 are used with 30 decision variables, and ZDT4 is tested using 10 decision variables. Problems DTLZ1-6 are adopted with 12 decision variables, and DTLZ7 is tested with 22 decision variables. For each algorithm, 30 independent runs are performed on each problem with population size of $\mu = 100$, running for $20,000$ function evaluations. Further, the size of archive in SPEA2 and AbYSS is set to 100, while the default settings are used for the other parameters. Parameter settings used in DDMOA are the following: the initial step size for local search $\delta^{(0)} = 1$, the initial step size for reproduction $\sigma^{(0)} = 5$, the tolerance for step size $\delta_{\text{tol}} = 10^{-3}$, the number of subpopulations $\alpha = 5$. Although DDMOA uses the step size adaptation rule presented in Algorithm 8, in this study the step size is updated as:

$$\sigma = \max\left\{\sigma^{(0)}(1 - g/E_{\text{g}}),\, 1\right\},$$

where $E_{\text{it}} = 10$ is the expected number of generations, and g is the current generation.



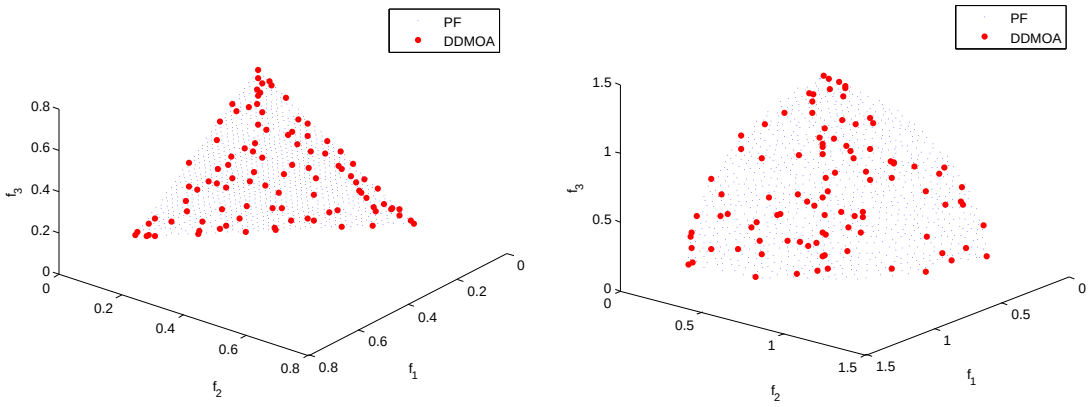
(a) median values of $I_{\epsilon+}$ (b) median values of I_H^-

Figure 6.4: Performance profiles on the median values of quality indicators.

Experimental Results

Figure 6.4 shows the median run comparisons in terms of the epsilon and hypervolume indicators. Figure 6.4(a) shows that the most accurate algorithm with respect to the epsilon indicator is DDMOA together with AbYSS, providing the best median values in 33% of the tested problems. In terms of robustness with respect to $I_{\epsilon+}$, the best algorithm is DDMOA. Figure 6.4(b) shows that the most accurate algorithm with respect to the hypervolume indicator is DDMOA, providing the best median values in 50% of the tested problems. In terms of robustness with respect to I_H^-, the best algorithm is again DDMOA.

Large difference in the values of τ for $\rho(1)$ observed in the above plots is explained by the fact that NSGA2, SPEA2, and AbYSS face difficulties in finding adequate approximation sets within the given number of function evaluations (20,000) on the DTLZ1 and DTLZ3 problems. This results in higher values of $I_{\epsilon+}$ and I_H^-. On the contrary, DDMOA obtains good approximation sets on these problems. This result is achieved due to the reproduction operator used in DDMOA. While the other three considered EMO algorithms use stochastic reproduction operators that may often require a significant computational effort to be spent until the optimal region is reached, DDMOA uses the combination of a pattern search method to learn the fitness landscape and a stochastic technique to generate offspring.



(a) DTLZ1 (b) DTLZ3

Figure 6.5: Performance of DDMOA on DTLZ1 and DTLZ3 using 10,000 evaluations.

This allows DDMOA to reach Pareto optimal region much faster than the other algorithms, especially on multimodal functions.

To reinforce these observations and to show the ability and efficiency of DDMOA in solving multimodal DTLZ1,3 problems, DDMOA is run on these two problems setting only the half of function evaluations as stopping criterion (10,000). Figure 6.5 presents approximation sets with the best values of the hypervolume indicator obtained by DDMOA after 30 runs. From Figures 6.5(a) and 6.5(b), it can be seen that DDMOA is able to find adequate approximations even within 10,000 function evaluations.

6.3.2 Intermediate Summary

In the performed experiments, DDMOA is tested on a set of problems and compared with state-of-the-art EMO algorithms. The obtained results reveal that DDMOA is able to solve a set of benchmark multiobjective problems. The proposed scheme to search for descent directions using local search combined with the strategy based on subpopulations is able to learn the fitness landscape and to generate new promising solutions spending a reasonable computational effort. When compared with other multiobjective solvers, DDMOA provides competitive results with respect to the epsilon and hypervolume indicators. Furthermore,

the experimental results show that DDMOA outperforms the other considered algorithms on multimodal problems due to its hybrid reproduction operator. Moreover, it is observed that DDMOA is able to find adequate approximations at much lower computational cost.

6.3.3 Further Experiments

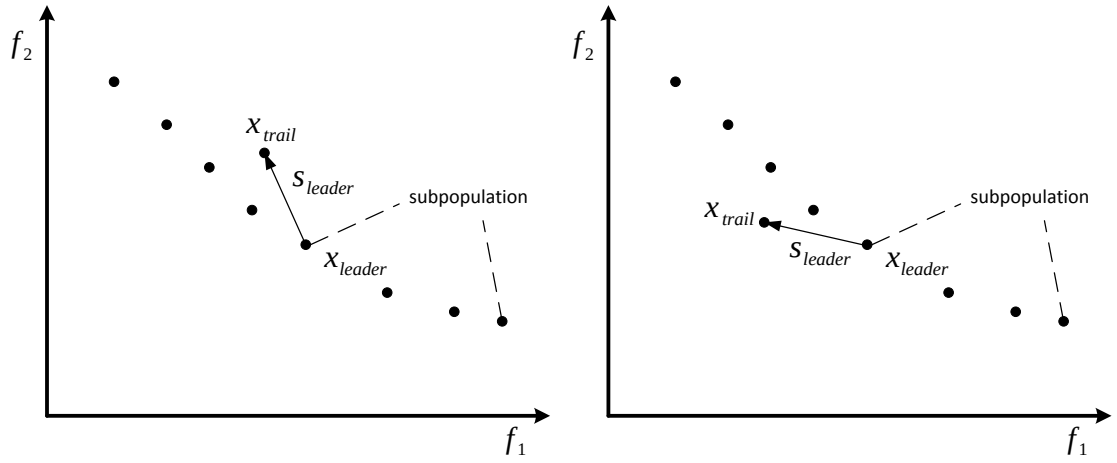
In this experimental study, the performance of DDMOA is further studied on a set of challenging problems. DDMOA is compared to popular EMO algorithms, namely, NSGA–II [46], IBEA [198] (in combination with the epsilon indicator), and MOEA/D [123]. Thus, DDMOA is compared against different algorithmic frameworks (dominance-based, indicator-based, and decomposition-based) employing different reproduction operators (genetic and DE operators). The performance of all algorithms is studied on the ZDT [197], DTLZ [50], and WFG [88] test suites.

Experimental Setup

DDMOA is implemented in the MATLAB® programming language, whereas the other algorithms are used within the jMetal framework [62]. All two and three-objective DTLZ and ZDT test problems except for ZDT4 are adopted with 30 decision variables, ZDT4 is tested using 10 decision variables. All two and three-objective WFG test problems are tested with 10 decision variables ($k = 4$ position-related parameters and $l = 6$ distance-related parameters). Additionally, WFG1 is used with a slight modification to the original definition presented in [88].

For all algorithms, 30 independent runs are performed on each test problem. The population size is set to 100 and the maximum number of function evaluations is set to 15,000 and 20,000 for the two-objective and three-objective test problems, respectively. All other parameter settings for NSGA–II, IBEA, and MOEA/D are default as defined in [62].

The parameter settings for DDMOA are the following: the initial step size for local search $\delta^{(0)} = 1$, the initial step size for reproduction $\sigma^{(0)} = 5$, the tolerance for step size $\delta_{\text{tol}} = 10^{-3}$, and the number of subpopulations $\alpha = 5$. In order to focus on promising re-



(a) without additional condition (b) with additional condition

Figure 6.6: Computation of descent directions for leaders.

gions of the search space, an additional condition for the acceptance of descent directions calculated for subpopulation leaders in `localSearch` procedure is introduced. Thus, descent direction $\boldsymbol{s}$ is accepted for subpopulation leader $\boldsymbol{x}_{\text{leader}}$ if two conditions are satisfied: (i) if $\boldsymbol{s}$ leads to a decrease in the value of the corresponding objective, and (ii) if a trail solution $\boldsymbol{x}_{\text{trail}} = \boldsymbol{x}_{\text{leader}} + \boldsymbol{s}$ is nondominated with respect to the current population (i.e. when a trail is added to the offspring population). To better understand this idea consider Figure 6.6, which presents a set of nondominated solutions in the objective space. Suppose that a descent direction needs to be computed with respect to f_1 for subpopulation leader $\boldsymbol{x}_{\text{leader}}$.

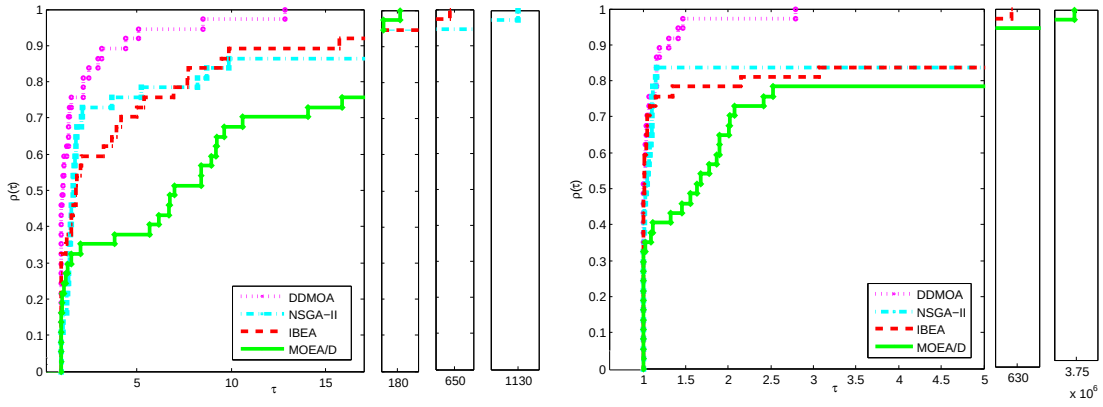
Figure 6.6(a) shows a possible descent direction $\boldsymbol{s}_{\text{leader}}$ that leads to the decrease in the value of f_1. It can be see that $\boldsymbol{x}_{\text{trail}}$ is dominated with respect to the population, so the descent direction shown in this figure may not lead to a promising region. On the other hand, Figure 6.6(b) shows a descent direction that can be obtained by introducing the condition that $\boldsymbol{x}_{\text{trail}}$ should be nondominated with respect to the population. This way, the possibility of obtaining a descent direction shown in Figure 6.6(a) is excluded, consequently the search pressure is increased. Thus, the DDMOA framework provides an additional feature to guide the search that can be easily exploited depending on the

characteristics of a given problem.

Furthermore, from Figure 6.5, one can see that DDMOA is able to converge to the Pareto front at low computational cost. However, the overall distribution of solutions along the Pareto front is not so good. Thus, to obtain a well-distributed set of solutions, in this study DDMOA uses `truncation` procedure (Algorithm 11) based on the nearest neighbor technique (truncation procedure used in SPEA2 [200]). This procedure works as follows. If the size of the new population is greater than population size μ, then individuals that have the minimum distance to the other individuals in the objective space are iteratively removed from the population until $|P^{(g+1)}| = \mu$. If there are several individuals with the minimum distance, the tie is broken by considering the second smallest distances, and so on.

Experimental Results

Figure 6.7 presents performance profiles on the median values of the epsilon and hypervolume indicators. The presented plots allow to observe and analyze the overall performance of the algorithms on all tested problems. Figure 6.7(a) shows that the most accurate algorithm is DDMOA, providing the best median values of the epsilon indicator in 46% of the



(a) median values of $I_{\epsilon+}$ (b) median values of I_H^-

Figure 6.7: Performance profiles on the median values of quality indicators.

	DDMOA	NSGA–II	IBEA	MOEA/D
Two-objective test problems				
ZDT1	**0.0063**$^{\text{II,III,IV}}$	0.013$^{\text{IV}}$	0.0082$^{\text{II,IV}}$	0.0997
ZDT2	**0.0074**$^{\text{II,III,IV}}$	0.0158$^{\text{III,IV}}$	0.029$^{\text{IV}}$	0.4135
ZDT3	**0.0066**$^{\text{III,IV}}$	0.0075$^{\text{III,IV}}$	0.0216$^{\text{IV}}$	0.1788
ZDT4	**0.0006**$^{\text{II,III,IV}}$	0.0029$^{\text{III,IV}}$	0.0087$^{\text{IV}}$	0.0675
ZDT6	**0.0002**$^{\text{II,III,IV}}$	0.0166$^{\text{IV}}$	0.0151$^{\text{II,IV}}$	0.0413
DTLZ1	**0.0174**$^{\text{II,III,IV}}$	0.1422	0.1206$^{\text{II}}$	0.1175
DTLZ2	**0.0088**$^{\text{II,III}}$	0.013$^{\text{III}}$	0.0138	0.0092$^{\text{II,III}}$
DTLZ3	**0.0183**$^{\text{II,III,IV}}$	0.1592	0.1731	0.1935
DTLZ4	**0.0091**$^{\text{II,III,IV}}$	0.0133$^{\text{III}}$	0.0149	0.0123$^{\text{III}}$
DTLZ5	**0.0091**$^{\text{II,III}}$	0.0125$^{\text{III}}$	0.0138	0.0094$^{\text{II,III}}$
DTLZ6	0.0009$^{\text{II,III}}$	0.5668	0.4662$^{\text{II}}$	**0.0007**$^{\text{I,II,III}}$
DTLZ7	**0.0051**$^{\text{II,III,IV}}$	0.0093$^{\text{III,IV}}$	0.0254$^{\text{IV}}$	0.2967
WFG1	0.0179$^{\text{III}}$	**0.0136**$^{\text{I,III,IV}}$	0.0242	0.0171$^{\text{III}}$
WFG2	0.2514	0.193$^{\text{I}}$	0.1931$^{\text{I}}$	**0.0196**$^{\text{I,II,III}}$
WFG3	0.0704	0.0148$^{\text{I}}$	**0.0083**$^{\text{I,II,IV}}$	0.0128$^{\text{I,II}}$
WFG4	0.0532$^{\text{IV}}$	0.016$^{\text{I,IV}}$	**0.012**$^{\text{I,II,IV}}$	0.0805
WFG5	0.085	0.0187$^{\text{I}}$	0.0169$^{\text{I,II}}$	**0.0167**$^{\text{I,II,III}}$
WFG6	0.0349	0.0409	0.0465	**0.0111**$^{\text{I,II,III}}$
WFG7	0.0242	0.0176$^{\text{I}}$	0.0121$^{\text{I,II}}$	**0.0098**$^{\text{I,II,III}}$
WFG8	0.1471$^{\text{III}}$	0.1838$^{\text{III}}$	0.2246	**0.1264**$^{\text{II,III}}$
WFG9	0.0331	0.0144$^{\text{I}}$	**0.0112**$^{\text{I,II,IV}}$	0.0128$^{\text{I,II}}$
Three-objective test problems				
DTLZ1	**0.0023**$^{\text{II,III,IV}}$	0.0824	0.0179$^{\text{II,IV}}$	0.1309
DTLZ2	0.0811$^{\text{II,IV}}$	0.1149$^{\text{IV}}$	**0.0775**$^{\text{I,II,IV}}$	0.5403
DTLZ3	**0.0063**$^{\text{II,III,IV}}$	0.1217$^{\text{IV}}$	0.0487$^{\text{II,IV}}$	0.3549
DTLZ4	0.1282$^{\text{III,IV}}$	**0.1175**$^{\text{I,III,IV}}$	0.632$^{\text{IV}}$	0.6643
DTLZ5	**0.0111**$^{\text{II,III,IV}}$	0.0154$^{\text{III,IV}}$	0.0407$^{\text{IV}}$	0.7908
DTLZ6	**0.0004**$^{\text{II,III,IV}}$	0.5041	0.1919$^{\text{II}}$	0.0688$^{\text{II,III}}$
DTLZ7	**0.0489**$^{\text{II,III,IV}}$	0.0802$^{\text{III,IV}}$	0.0904$^{\text{IV}}$	0.6863
WFG1	0.0763$^{\text{II,IV}}$	0.0902$^{\text{IV}}$	**0.0521**$^{\text{I,II,IV}}$	0.3201
WFG2	0.1511$^{\text{IV}}$	**0.0967**$^{\text{I,IV}}$	0.1949	0.1947
WFG3	0.0751$^{\text{III,IV}}$	**0.0541**$^{\text{I,III,IV}}$	0.1109$^{\text{IV}}$	0.4534
WFG4	0.1097$^{\text{II,IV}}$	0.1282$^{\text{IV}}$	**0.0794**$^{\text{I,II,IV}}$	0.7283
WFG5	0.1638$^{\text{IV}}$	0.1171$^{\text{I,IV}}$	**0.0752**$^{\text{I,II,IV}}$	0.6741
WFG6	0.0977$^{\text{II,IV}}$	0.1485$^{\text{IV}}$	**0.0887**$^{\text{I,II,IV}}$	0.854
WFG7	0.151$^{\text{IV}}$	0.1069$^{\text{I,IV}}$	**0.0697**$^{\text{I,II,IV}}$	0.587
WFG8	**0.1809**$^{\text{II,IV}}$	0.2328$^{\text{IV}}$	0.1832$^{\text{II,IV}}$	0.6951
WFG9	0.1085$^{\text{II,IV}}$	0.1267$^{\text{IV}}$	**0.0782**$^{\text{I,II,IV}}$	0.7224

Table 6.1: Median values of the epsilon indicator after 30 runs.

	DDMOA	NSGA–II	IBEA	MOEA/D
Two-objective test problems				
ZDT1	$\mathbf{0.2265}^{\text{II,III,IV}}$	0.2315^{IV}	$0.2287^{\text{II,IV}}$	0.3285
ZDT2	$\mathbf{0.5576}^{\text{II,III,IV}}$	0.5657^{IV}	0.5639^{IV}	0.7364
ZDT3	$\mathbf{0.2758}^{\text{II,III,IV}}$	$0.2787^{\text{III,IV}}$	0.2801^{IV}	0.4499
ZDT4	$\mathbf{0.0041}^{\text{II,III,IV}}$	$0.0048^{\text{III,IV}}$	0.0089^{IV}	0.0526
ZDT6	$\mathbf{0.2854}^{\text{II,III,IV}}$	0.2951^{IV}	$0.2941^{\text{II,IV}}$	0.3125
DTLZ1	$\mathbf{0.0007}^{\text{II,III,IV}}$	0.041	0.0289^{II}	0.0438
DTLZ2	$\mathbf{0.7776}^{\text{II,III,IV}}$	0.7791^{IV}	$0.7785^{\text{II,IV}}$	0.7799
DTLZ3	$\mathbf{0.0006}^{\text{II,III,IV}}$	0.0394	$0.0345^{\text{II,IV}}$	0.0684
DTLZ4	$\mathbf{0.7797}^{\text{II,III,IV}}$	0.7812^{IV}	0.7806	0.7858
DTLZ5	$\mathbf{0.779}^{\text{II,III,IV}}$	0.7801^{IV}	$0.7797^{\text{II,IV}}$	0.7814
DTLZ6	$\mathbf{0.0083}^{\text{II,III,IV}}$	0.5818	0.4125^{II}	$0.0083^{\text{II,III}}$
DTLZ7	$\mathbf{0.3398}^{\text{II,III,IV}}$	0.3442^{IV}	$0.3428^{\text{II,IV}}$	0.5286
WFG1	0.3476	$\mathbf{0.3382}^{\text{I,III,IV}}$	0.34^{I}	$0.3395^{\text{I,III}}$
WFG2	0.4645	$0.3746^{\text{I,III}}$	0.3766^{I}	$\mathbf{0.3283}^{\text{I,III}}$
WFG3	0.4828	0.4221^{I}	$\mathbf{0.4184}^{\text{I,II,IV}}$	$0.4202^{\text{I,II}}$
WFG4	0.7228	$0.6892^{\text{I,IV}}$	$\mathbf{0.6873}^{\text{I,II,IV}}$	0.7064^{I}
WFG5	0.645	$0.6291^{\text{I,IV}}$	$\mathbf{0.6287}^{\text{I,II,IV}}$	0.6301^{I}
WFG6	0.7477	0.732^{I}	0.7313^{I}	$\mathbf{0.7014}^{\text{I,II,III}}$
WFG7	0.6954	0.6775^{I}	$\mathbf{0.6753}^{\text{I,II,IV}}$	$0.6761^{\text{I,II}}$
WFG8	$\mathbf{0.5968}^{\text{II,III}}$	0.633	0.6415	$0.5971^{\text{II,III}}$
WFG9	0.6779	$0.6343^{\text{I,IV}}$	$\mathbf{0.6321}^{\text{I,II,IV}}$	0.6362^{I}
Three-objective test problems				
DTLZ1	$\mathbf{0}.08e-6^{\text{II,III,IV}}$	0.0026^{IV}	$0.05e-3^{\text{II,IV}}$	0.2962
DTLZ2	$0.4592^{\text{II,IV}}$	0.4813^{IV}	$\mathbf{0.4429}^{\text{I,II,IV}}$	0.893
DTLZ3	$\mathbf{0}.08e-5^{\text{II,III,IV}}$	0.0043^{IV}	$0.0002^{\text{II,IV}}$	0.5863
DTLZ4	$\mathbf{0.4666}^{\text{II,III,IV}}$	$0.4721^{\text{III,IV}}$	0.6256^{IV}	0.8852
DTLZ5	$\mathbf{0.8808}^{\text{II,III,IV}}$	0.8818^{IV}	$0.8817^{\text{II,IV}}$	0.9807
DTLZ6	$\mathbf{0.0047}^{\text{II,III,IV}}$	0.3865	$0.0438^{\text{II,IV}}$	0.1807^{II}
DTLZ7	$\mathbf{0.3763}^{\text{II,III,IV}}$	0.4017^{IV}	0.3876^{IV}	0.9499
WFG1	0.1143^{IV}	$\mathbf{0.0778}^{\text{I,III,IV}}$	$0.0816^{\text{I,IV}}$	0.4749
WFG2	0.2022^{IV}	$\mathbf{0.0725}^{\text{I,IV}}$	0.2223^{IV}	0.3724
WFG3	0.416^{IV}	$0.3514^{\text{I,IV}}$	$\mathbf{0.349}^{\text{I,II,IV}}$	0.7235
WFG4	0.5309^{IV}	$0.5104^{\text{I,IV}}$	$\mathbf{0.4587}^{\text{I,II,IV}}$	0.8674
WFG5	0.492^{IV}	$0.4602^{\text{I,IV}}$	$\mathbf{0.4263}^{\text{I,II,IV}}$	0.8553
WFG6	$0.5735^{\text{II,IV}}$	0.6106^{IV}	$\mathbf{0.5553}^{\text{I,II,IV}}$	0.9292
WFG7	0.4372^{IV}	$0.3732^{\text{I,IV}}$	$\mathbf{0.3376}^{\text{I,II,IV}}$	0.814
WFG8	$0.5204^{\text{II,IV}}$	0.5579^{IV}	$\mathbf{0.5075}^{\text{I,II,IV}}$	0.8991
WFG9	0.5372^{IV}	$0.5214^{\text{I,IV}}$	$\mathbf{0.4717}^{\text{I,II,IV}}$	0.8794

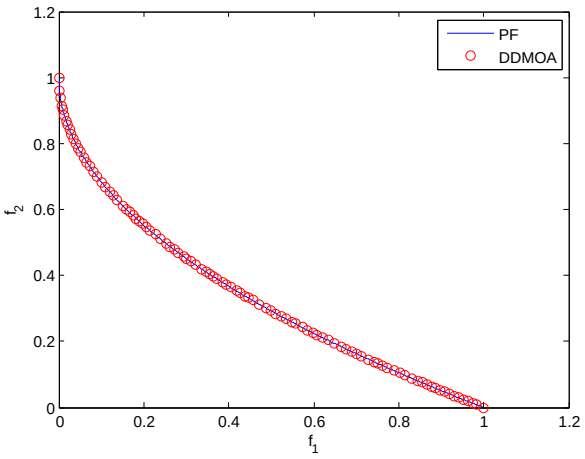
Table 6.2: Median values of the hypervolume indicator after 30 runs.

tested problems. In terms of robustness, the best algorithm is also DDMOA. Figure 6.7(b) shows that the most accurate algorithm is DDMOA, providing the best median values of the hypervolume indicator in 49% of the tested problems. In terms of robustness, the best algorithm is again DDMOA.

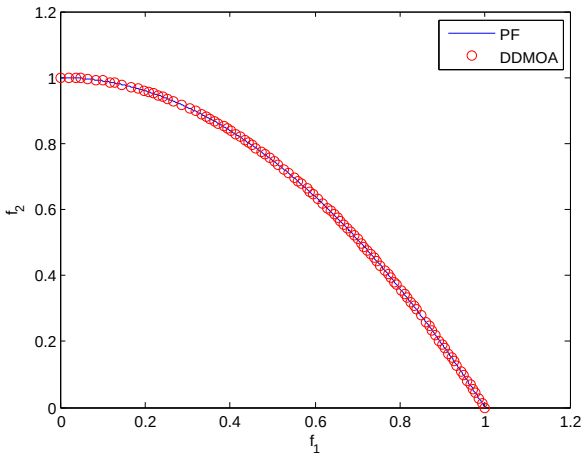
Table 6.1 presents the statistical comparison and the median values of the epsilon indicator. The superscripts I, II, III and IV indicate whether the respective algorithm performs significantly better than DDMOA, NSGA–II, IBEA, and MOEA/D, respectively. The best value in each row is marked bold (the lower the better). DDMOA outperforms the other algorithms on ZDT and the majority of the two and three-objective DTLZ test problems. On the two-objective WFG test problems, the best performance is achieved by MOEA/D that provides the best median values of the epsilon indicator in 5 out of 9 test problems. On the three-objective WFG test problems, IBEA has the best performance, providing the best median values of the epsilon indicator in 6 out of 9 test problems.

Table 6.2 shows the statistical comparison and the median values of the hypervolume indicator. The superscripts I, II, III and IV indicate whether the respective algorithm performs significantly better than DDMOA, NSGA–II, IBEA, and MOEA/D, respectively. The best value in each row is marked bold (the lower the better). DDMOA outperforms the other algorithms on the two-objective ZDT and DTLZ test problems, providing significantly better values of the hypervolume indicator than all other considered algorithms. IBEA performs significantly better than the other algorithms on the 5 two-objective WFG test problems. On the three-objective WFG3-9 test problems, IBEA gives significantly better results in terms of the hypervolume indicator than the other algorithms. The dominance of IBEA on WFG test problems in terms of the hypervolume can be explained by the fact that it is the only algorithm that attempts to maximize the cumulative hypervolume covered by nondominated solutions.

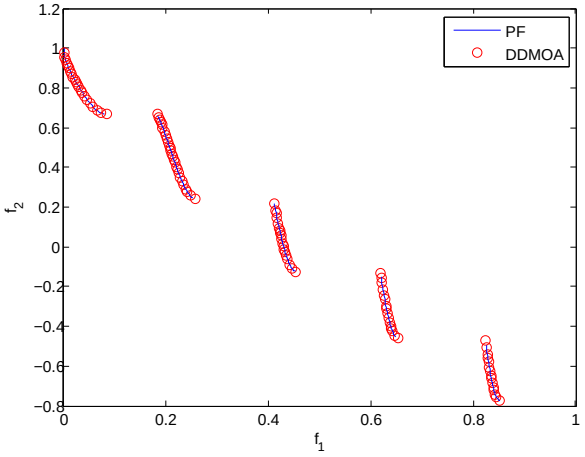
Figures 6.8–6.12 present approximation sets with the best hypervolume values found by DDMOA in 30 runs. In Figures 6.8, 6.9, and 6.10, it can be seen that DDMOA reaches the Pareto fronts for all the ZDT and DTLZ test problems, giving well-distributed sets of solutions in the objective space. From Figure 6.11, one can see that DDMOA performs



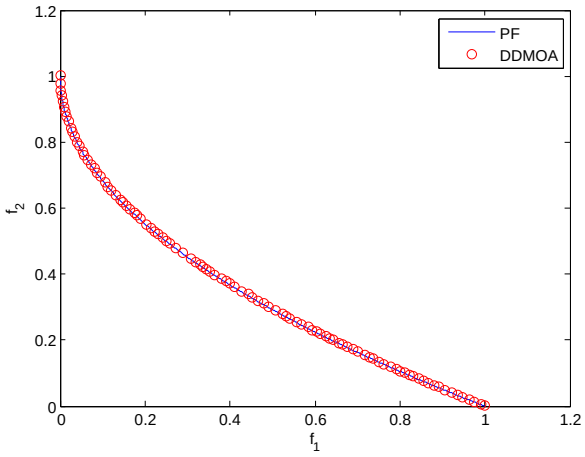
(a) ZDT1



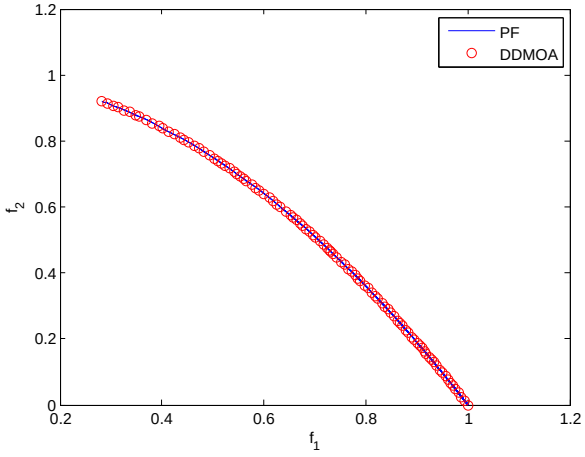
(b) ZDT2



(c) ZDT3

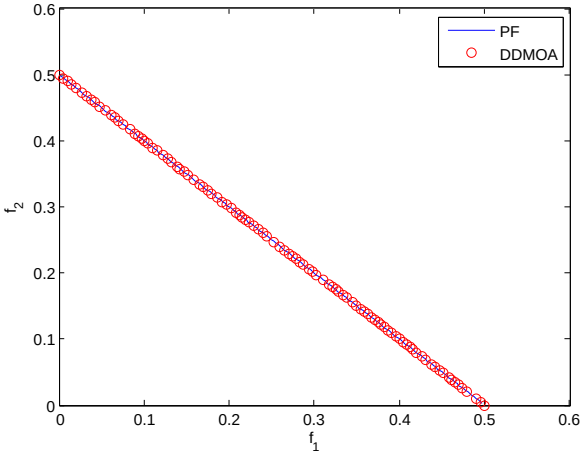


(d) ZDT4

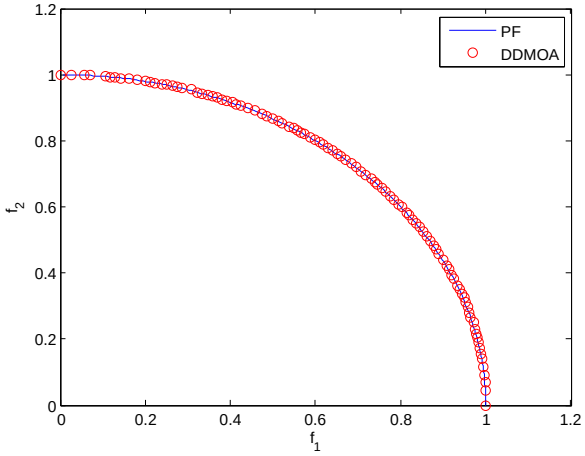


(e) ZDT6

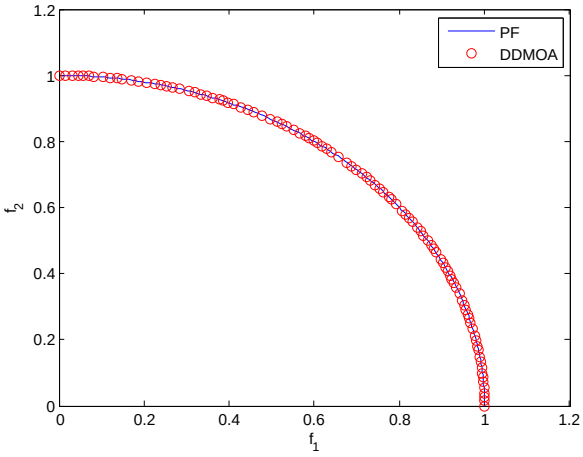
Figure 6.8: Performance of DDMOA on the ZDT test suite.



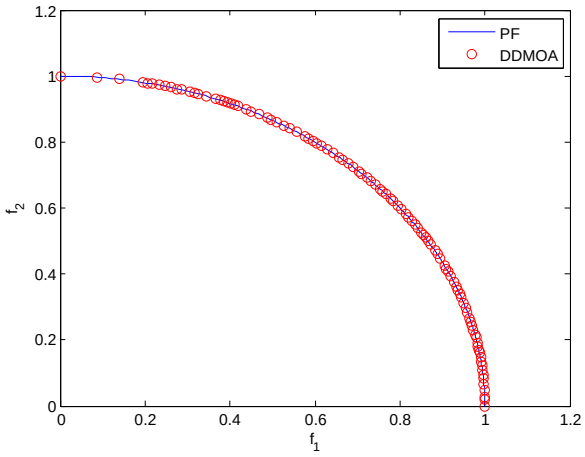
(a) DTLZ1



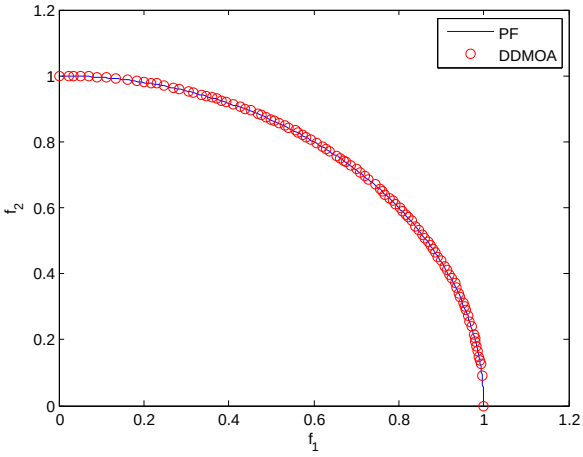
(b) DTLZ2



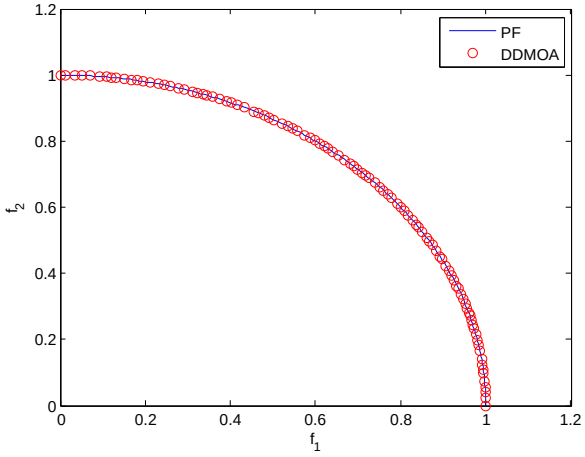
(c) DTLZ3



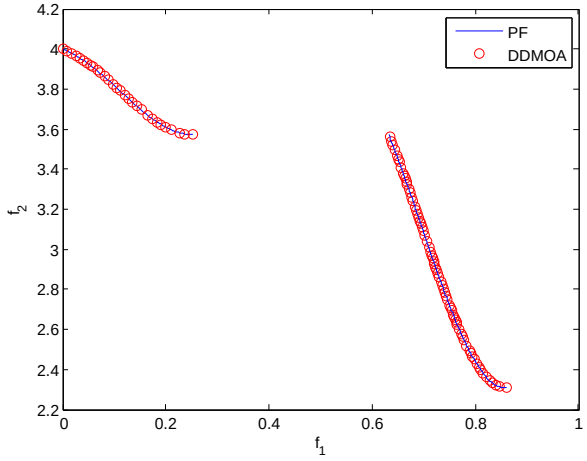
(d) DTLZ4



(e) DTLZ5

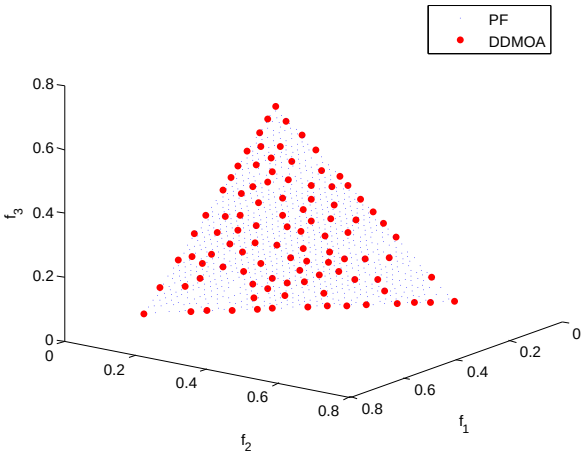


(f) DTLZ6

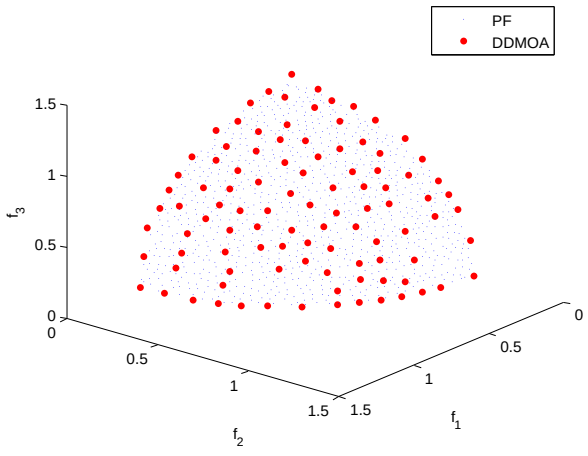


(g) DTLZ7

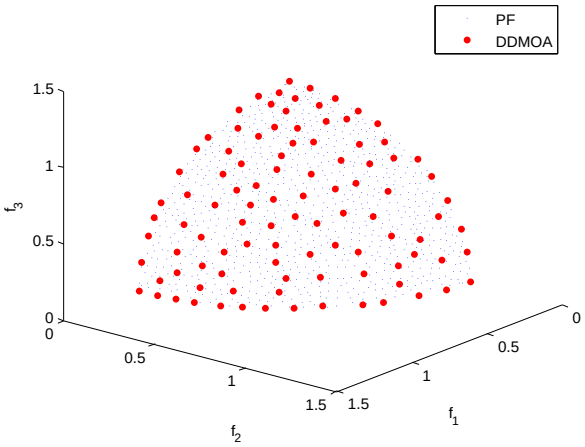
Figure 6.9: Performance of DDMOA on the two-objective DTLZ test suite.



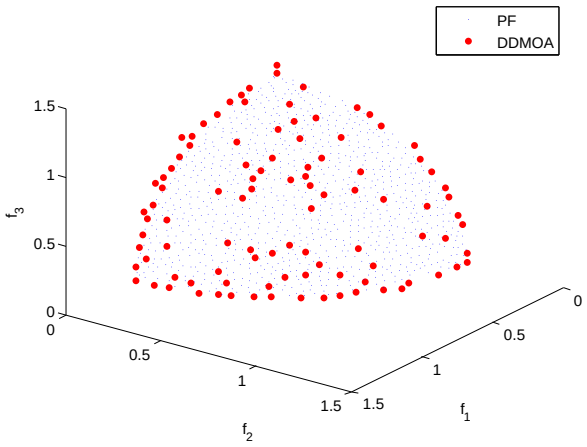
(a) DTLZ1



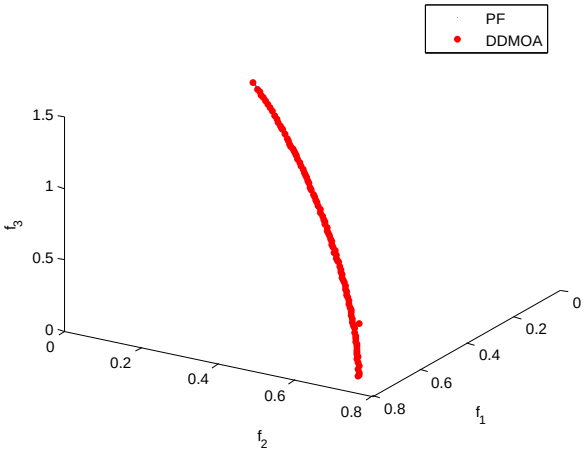
(b) DTLZ2



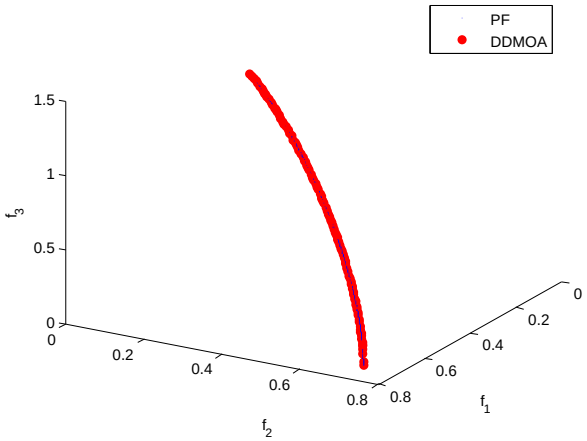
(c) DTLZ3



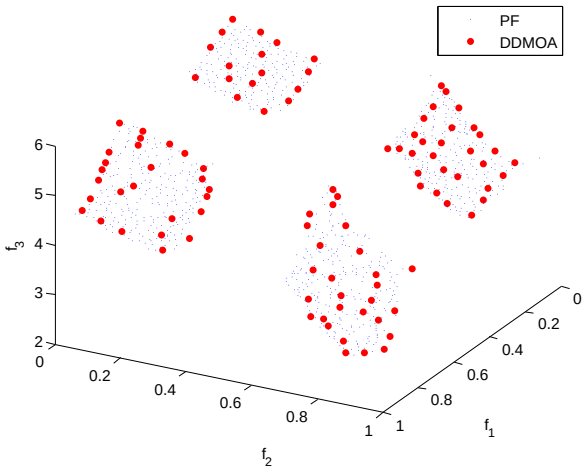
(d) DTLZ4



(e) DTLZ5

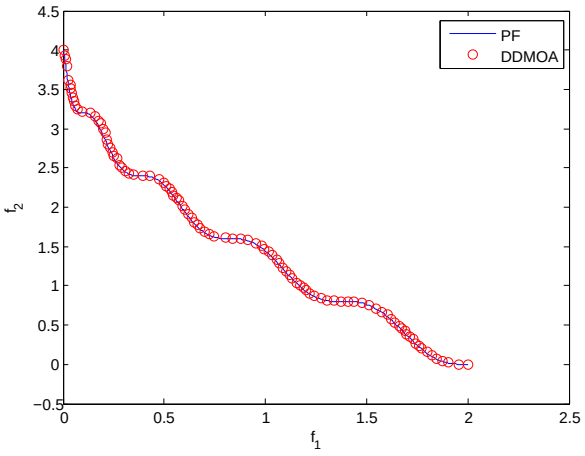


(f) DTLZ6

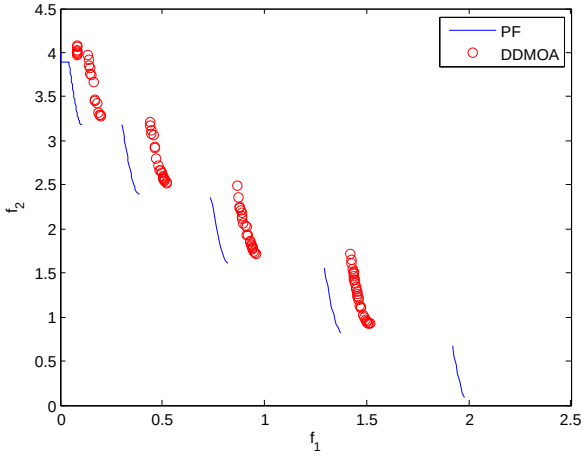


(g) DTLZ7

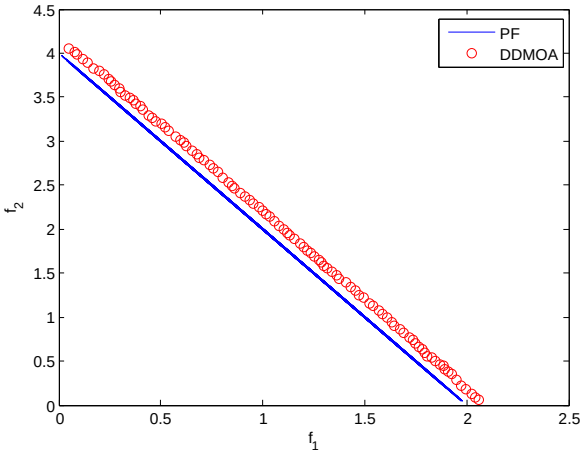
Figure 6.10: Performance of DDMOA on the three-objective DTLZ test suite.



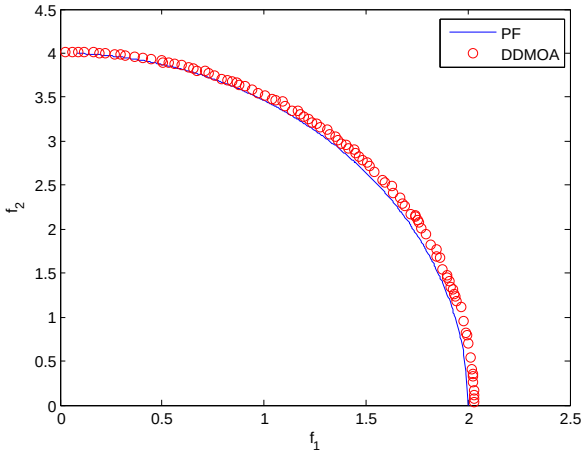
(a) WFG1



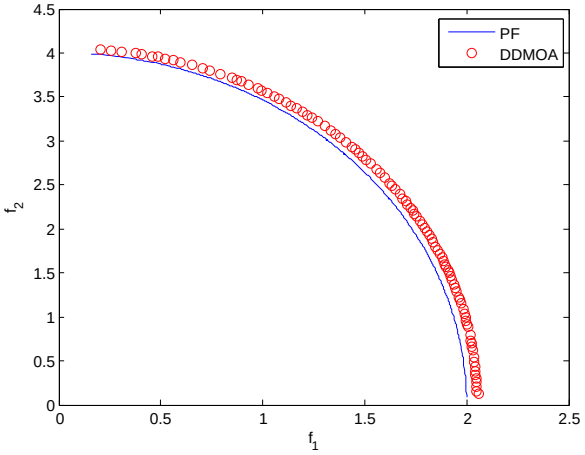
(b) WFG2



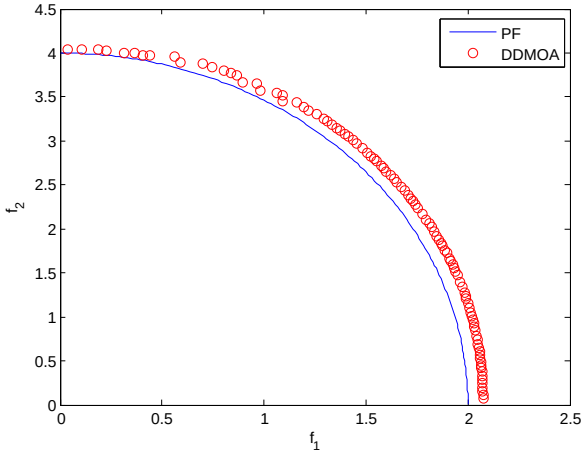
(c) WFG3



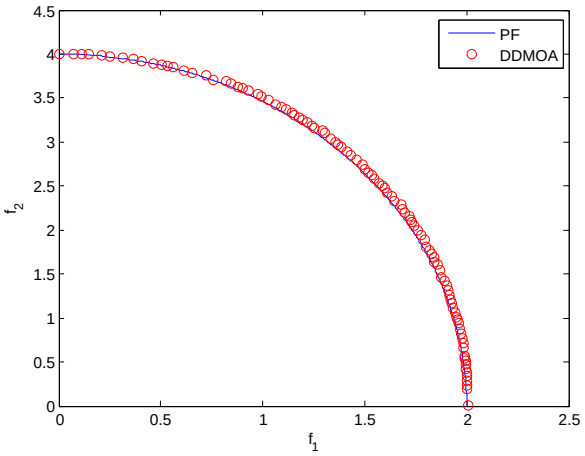
(d) WFG4



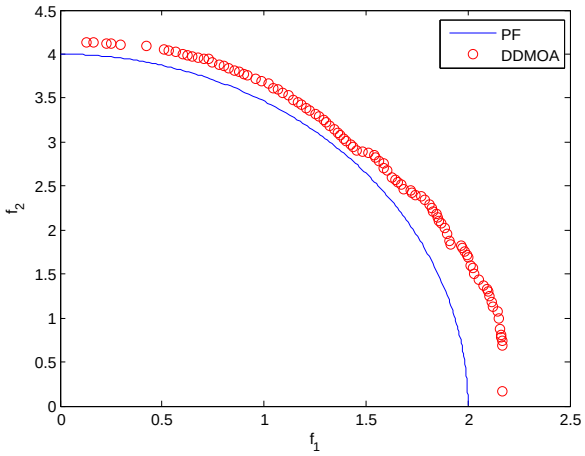
(e) WFG5



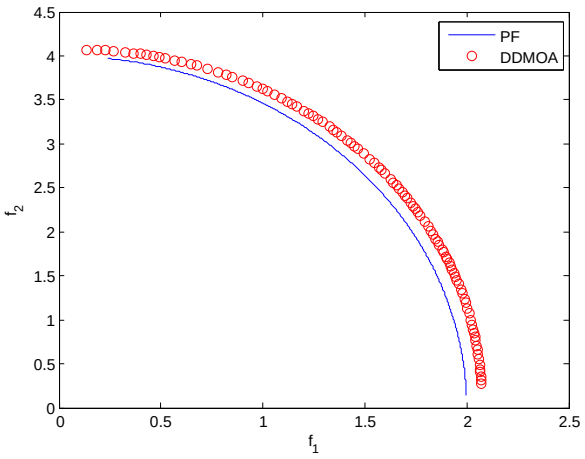
(f) WFG6



(g) WFG7

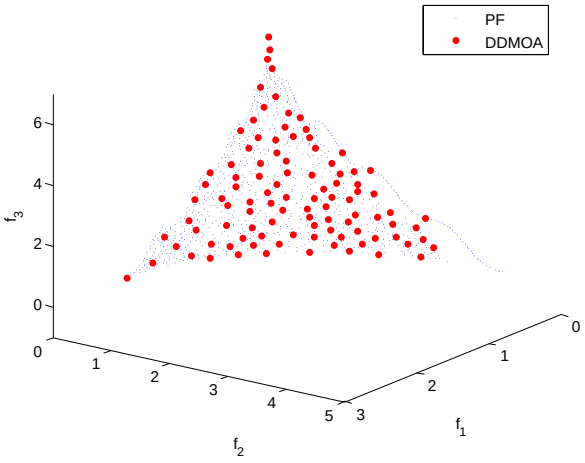


(h) WFG8

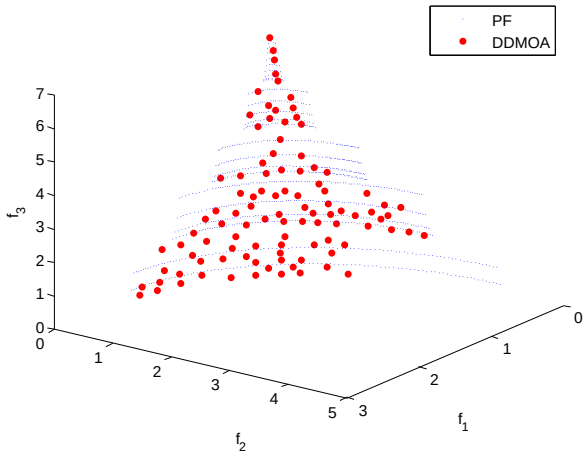


(i) WFG9

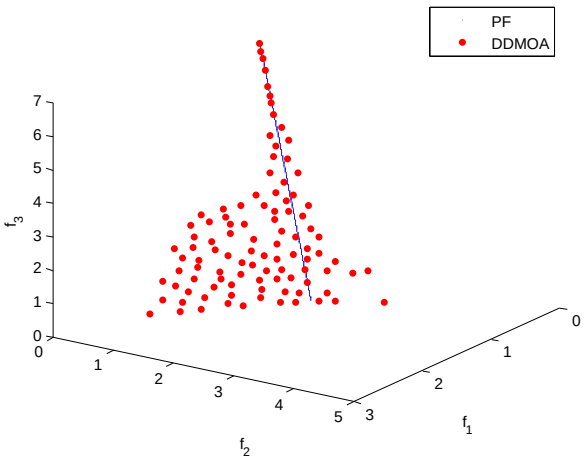
Figure 6.11: Performance of DDMOA on the two-objective WFG test suite.



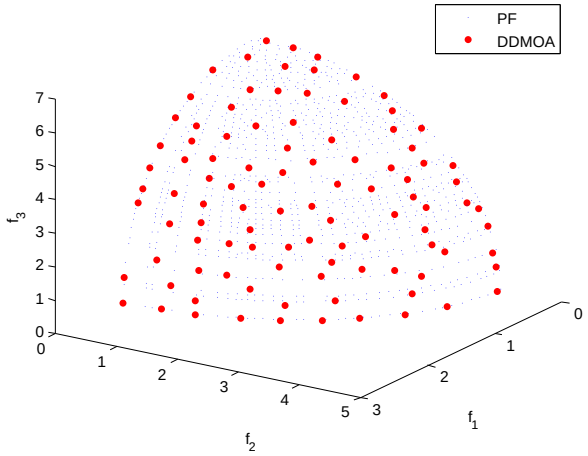
(a) WFG1



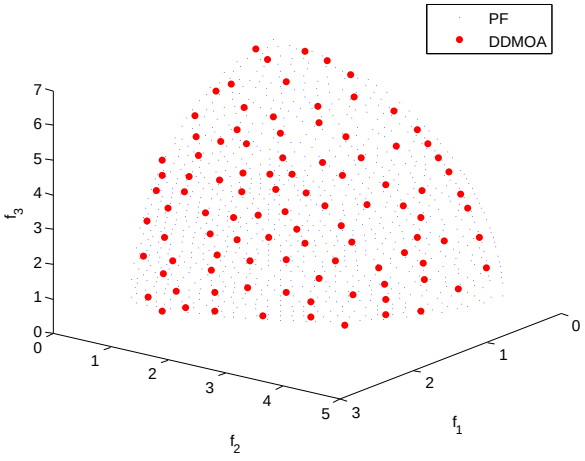
(b) WFG2



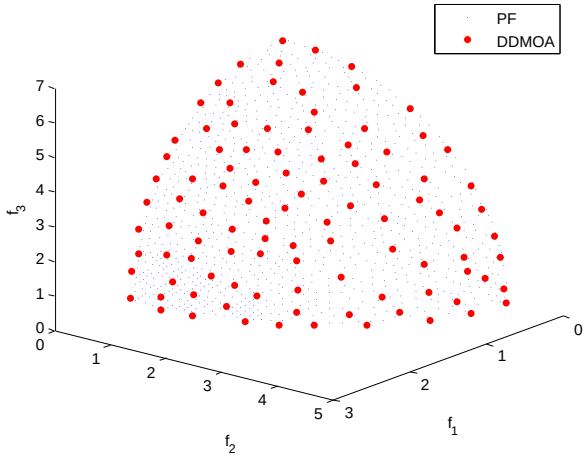
(c) WFG3



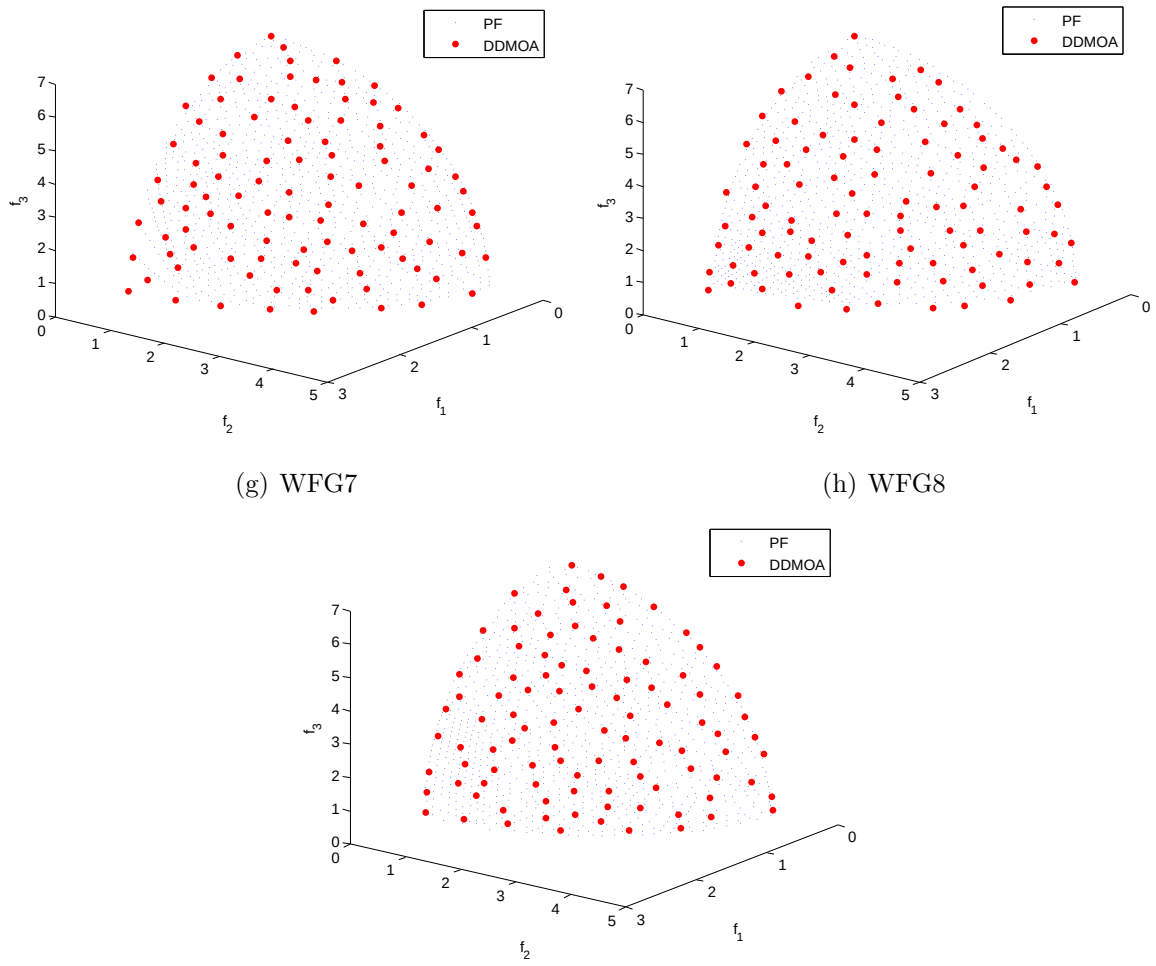
(d) WFG4



(e) WFG5



(f) WFG6



(g) WFG7 (h) WFG8

(i) WFG9

Figure 6.12: Performance of DDMOA on the three-objective WFG test suite.

poorly on the majority of the two-objective WFG test problems. However, adequate Pareto front approximations are achieved by DDMOA on WFG1,4,7. Although DDMOA performs poorly on the WFG8 test problem, from Table 6.2, one can see that DDMOA provides the best median value of the hypervolume indicator, being significantly better than NSGA-II and IBEA on this problem. In Figure 6.12, it can be observed that DDMOA is able to find adequate Pareto front approximations for all the three-objective WFG test problems, except for the WFG3 test problem.

6.4 Summary

The review of existing approaches for MO given in Chapter 3 shows that the majority of state-of-the-art EMO algorithms use variation operators initially designed for single-objective EAs. Since the particularities of multiobjective optimization are often not taken into consideration, some operators that are suitable for single-objective optimization may lead to a poor performance in the presence of multiple conflicting objectives. Although combinations of classical methods and evolutionary algorithms often outperform traditional EAs in single-objective optimization, such methods still remain to be an under-explored research area in multiobjective optimization.

This chapter introduces a new hybrid multiobjective evolutionary algorithm, termed DDMOA. DDMOA combines different concepts from single-objective and multiobjective optimization. The main idea of generating new candidate solutions is borrowed from Timmel's method for MO. To overcome the limitations related to the use of gradients, descent directions are calculated. Found descent directions are stored in the search matrix of each individual in the population. This way, search matrices store information about fitness landscape in the vicinity of each population member. The computation of descent directions is performed by a pattern search method. To perform the search in the objective space, the widely used Pareto dominance-based environmental selection is adopted from EMO algorithms.

The performed experimental studies show that DDMOA is a competitive multiobjective optimizer. The ability of DDMOA to outperform state-of-the-art EMO algorithms on multimodal problems is directly related to its hybrid reproduction operator. The obtained results clearly indicate that the proposed approach is a promising direction for further research.

Chapter 7

Generalized Descent Directions-Guided Multiobjective Algorithm

7.1 Introduction

The previous chapter introduces a multiobjective evolutionary algorithm with a hybrid reproduction operator, which is based on the idea of Timmel's method for generating new candidate solutions. Although the algorithm exhibits a competitive performance in the comparative studies with some state-of-the-art MOEAs, several issues are identified as potential weaknesses of DDMOA. In particular, in DDMOA the population consists only of nondominated individuals that can cause only a few or even one individual being present in the current population. That can severely deteriorate the performance of the algorithm. Since a few individuals are present in the population, it becomes difficult to escape from suboptimal regions of the search space. Another weakness is that the original DDMOA is not applicable to many-objective problems. Moreover, when the number of objectives is increased, finding descent directions for all objectives using local search may be computationally expensive.

This chapter addresses the aforementioned problems and presents a generalized descent directions-guided multiobjective algorithm (DDMOA2). DDMOA2 adopts the concepts used for reproduction from its predecessor. However, it differs from the original algorithm in other aspects. DDMOA maintains the population consisting only of nondominated individuals and calculates descent directions for all objectives. This scheme is similar to Timmel's method. On the other hand, in DDMOA2 the population size remains greater than or equal to the user-specified value. This way, nondominated as well as dominated individuals can be present in the current population. Moreover, each population member has a chance to produce offspring. Furthermore, DDMOA2 uses a scalarizing fitness assignment in order to deal with many-objective problems. To improve efficiency of the algorithm, descent directions are calculated only for two randomly chosen objectives. The introduction of all these modifications leads to a new algorithmic framework aiming to obtain an efficient and robust EMO algorithm.

7.2 DDMOA2

Algorithm 12 DDMOA2

1: $g \leftarrow 0$;
2: `initialize:` $P^{(g)}$, $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_\mu\}$;
3: **repeat**
4: $P^{(g)} \leftarrow$ `leaderSelection`$(P^{(g)})$;
5: $P^{(g)} \leftarrow$ `updateSearchMatrix`$(P^{(g)})$;
6: $P^{(g)} \leftarrow$ `updateStepSize`$(P^{(g)})$;
7: $P^{(g)} \leftarrow$ `parentSelection`$(P^{(g)})$;
8: $P^{(g)} \leftarrow$ `mutation`$(P^{(g)})$;
9: $P^{(g+1)} \leftarrow$ `environmentalSelection`$(P^{(g)})$;
10: $g \leftarrow g + 1$;
11: **until** the stopping criterion is met
12: output: $P^{(g)}$;

The main loop of DDMOA2 is given by Algorithm 12. DDMOA2 is a hybrid evolutionary algorithm with the $(\mu+\lambda)$ selection scheme. In each generation, the selection of leaders and the adaptation of the strategy parameters of all population members are followed by the successive application of parent selection, mutation, and environmental selection.

In DDMOA2, an individual a_i ($i \in \{1, \ldots, \mu\}$) in the current population $P^{(g)}$ in generation g is a tuple of the form $[\boldsymbol{x}_i, \delta_i, \boldsymbol{S}_i, \sigma_i]$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the decision vector, $\delta_i > 0$ is the step size used for local search, $\boldsymbol{S}_i \in \mathbb{R}^{n \times 2}$ is the search matrix, and $\sigma_i > 0$ is the step size used for reproduction.

The following subsections discuss the components of DDMOA2 in more detail.

7.2.1 Initialize Procedure

The algorithm starts by generating a set of weight vectors $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_\mu\}$ and initializing the population of size μ using Latin hypercube sampling [126]. The strategy parameters of each population member are initialized taking default values. The search matrix $\boldsymbol{S}^{(0)}$ is initialized by simply generating a zero matrix of size $n \times 2$.

7.2.2 Leader Selection Procedure

Each generation of DDMOA2 is started by selecting leaders of the current population. A leader is a population member that performs the best on at least one weight vector. Thus, leaders are selected as follows. First, the objective values of all individuals in the population are normalized:

$$\overline{f}_i = \frac{f_i - f_i^{\min}}{f_i^{\max} - f_i^{\min}}, \quad \forall i \in \{1, \ldots, m\} \tag{7.2.1}$$

where $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum values of the i-th objective in the current population, respectively, and $\overline{f}_i \in [0, 1], \forall i \in \{1, \ldots, m\}$ is the normalized objective value. For each weight vector, the fitness of each population member is calculated on a given weight vector using the weighted Chebyshev method, which after normalization of

objectives can be defined as:

$$f_{\text{fitness}} = \max_{1 \leq i \leq m} \{ w_i \overline{f}_i(\boldsymbol{x}) \} \tag{7.2.2}$$

where f_{fitness} is the fitness of the population member $\boldsymbol{x}$ on the weight vector $\boldsymbol{w}$. An individual having the best fitness on a given weight vector is a leader. It should be noted that one leader can have the best fitness value on several weight vectors.

7.2.3 Update Search Matrix Procedure

After selecting leader individuals, the search matrices of all individuals in the current population are updated in `updateSearchMatrix` procedure. In DDMOA2, descent directions are calculated only for two randomly chosen objectives, regardless of the dimensionality of the objective space. Thus, the search matrix of each population member contains two columns that store descent directions for these randomly chosen objectives. The motivation behind finding descend directions only for two instead of all objectives is that in the presence of a large number of objectives calculating descend directions for all objectives using local search may become computationally expensive.

In the beginning of `updateSearchMatrix` procedure, two objectives are chosen at random, and only leaders of the current population are considered while all the other individuals are temporarily discarded. Thereafter, for the first chosen objective, the resulting population is sorted in ascending order and partitioned into α equal parts. Thus, α subpopulations are defined in order to promote different reference points for the computation of descent directions. It follows that in each subpopulation, a representative individual a_{r} is selected. A representative of the subpopulation is a solution with the smallest value of the corresponding objective function among other solutions in the subpopulation and $\delta > \delta_{\text{tol}}$. Thus, if the solution with the smallest value of the corresponding objective has $\delta \leq \delta_{\text{tol}}$ then the solution with the second smallest value is selected as representative and so on. After that, a descent direction for the corresponding objective function is computed for the representative using coordinate search [175]. During coordinate search, the step size δ of the representative is reduced if no decrease in the objective function value is found. Each

time a trial solution is calculated, this solution is compared with the current population. If this trial solution has a smaller value for at least one objective compared with each member of the current population then it is added to the population, assuming the default values of the strategy parameters. When a descent direction $\boldsymbol{s}_{\mathrm{r}}$ for the subpopulation representative is found, descent directions for all other subpopulation members are computed as follows:

$$\boldsymbol{s}_i = \boldsymbol{x}_{\mathrm{r}} - \boldsymbol{x}_i + \boldsymbol{s}_{\mathrm{r}}, \tag{7.2.3}$$

where $\boldsymbol{s}_i$ is the descent direction for the i-th subpopulation member, $\boldsymbol{x}_i$ is the decision vector of the i-th subpopulation member, $\boldsymbol{x}_{\mathrm{r}}$ is the subpopulation representative, and $\boldsymbol{s}_{\mathrm{r}}$ is the descent direction for the representative. The calculated descent directions are stored in the first column of the search matrices of the corresponding individuals. Thereafter, the same procedure for finding descent directions is performed for the second chosen objective and the results are stored in the second column of the search matrices.

After the search matrices of leaders of the current population are updated, the search matrices of all other population members need to be updated. For this purpose, a simple stochastic procedure is used. For each non-leader individual, a leader is randomly chosen and this leader shares its search matrix, i.e., non-leader individual's search matrix is equal to the selected leader's search matrix. At the end of `updateSearchMatrix` procedure, the search matrices of all population members are updated. Since some promising solutions satisfying the aforementioned conditions are added to the population during coordinate search, at the end of this procedure, the population size is usually greater than μ.

7.2.4 Update Step Size Procedure

Before generating offspring, the step size of each population member needs to be updated. However, there is no common rule to update the step size σ, but it must be done carefully to ensure convergence to the Pareto set, starting from a larger value at the beginning and gradually reducing it during the generations. DDMOA2 uses the following rule for

updating the step size of each population member:

$$\sigma = \max\{\exp(\tau N(0,1))\, \sigma_0^{(1-\frac{3\, funEval}{maxEval})}, \delta_{\text{tol}}\} \tag{7.2.4}$$

where τ is the learning parameter ($\tau = 1/\sqrt{2n}$), $N(0,1)$ is a random number sampled from the normal distribution with mean 0 and standard deviation 1, σ_0 is the initial value of the step size, $funEval$ is the current number of function evaluations, and $maxEval$ is the maximum number of function evaluations. The idea here is to exponentially reduce the step size depending on the current number of function evaluations and multiply it by a scaling factor, thereby obtaining different step size values among the population members.

7.2.5 Parent Selection Procedure

In each generation, λ offspring individuals are generated by mutating the correspondent parent. The decision on how many offspring are produced from each population member is made in `parentSelection` procedure. At the beginning, it is assumed that none offspring is generated by each individual. Then, binary tournament selection based on scalarizing fitness is performed to identify which population members will be mutated in order to produce offspring. This selection process is based on the idea proposed in [97] and fosters promising individuals to produce more offspring.

The procedure starts by normalizing the objective values of all individuals in the population as defined in (7.2.1). Then, the selection process is performed in two stages. In the first stage, only leaders of the current population are considered. For each weight vector, two individuals are randomly selected and the fitness of the corresponding individual is calculated as defined in (7.2.2). The individual having smallest fitness value is a winner and the number of times it is mutated is augmented by one. Thereafter, all leaders are removed from the population and, for each weight vector, binary tournament selection is performed on the resulting population in the same way as it is done for leaders. It should be noted that some population members might be mutated several times, as a result of this procedure, while others will not produce any offspring at all.

7.2.6 Mutation Procedure

After identifying which population members have to be mutated, offspring are generated in `mutation` procedure. For the corresponding individual, the mutation is performed as follows:

$$\boldsymbol{x}' = \boldsymbol{x} + \sigma \boldsymbol{S}\, \boldsymbol{\nu} \tag{7.2.5}$$

where σ is the step size, $\boldsymbol{S}$ is the search matrix and $\boldsymbol{\nu}$ is a column vector of random numbers sampled from the uniform distribution ($\forall i \in \{1, 2\} : \nu_i \sim \mathbb{U}(0, 1)$). To guarantee that each new solution $\boldsymbol{x}' = (x'_1, \ldots, x'_n)^{\mathrm{T}}$ belongs to Ω, projection is applied to each component of the decision vector: $\boldsymbol{x}' = \min\{\max\{\boldsymbol{x}', \boldsymbol{l}\}, \boldsymbol{u}\}$. After offspring $\boldsymbol{x}'$ is repaired, it is evaluated and added to the population.

7.2.7 Environmental Selection Procedure

At the end of each generation, μ fittest individuals are selected from the enlarged population in `environmentalSelection` procedure. DDMOA2 uses the selection mechanism proposed in [92].

First, the objective values of all individuals are normalized as defined in (7.2.1). Next, the following steps are performed:

1. matrix $\boldsymbol{M}$ is calculated, which stores metrics for each population member on each weight vector (for each population member, a metric on a corresponding weight vector is computed as defined in (7.2.2));

2. for each column (weight vector), the minimum and second smallest metric value are found;

3. for each column, metric values are scaled by the minimum value found, except for the row which gave the minimum value. This result is scaled by the second lowest value;

4. for each row (population member), the minimum scaled value is found, this value represents individual's fitness;

5. the resulting column vector is sorted, and μ individuals with the smallest fitness values are selected.

Normalizing objective values allows to cope with differently scaled objectives, while the weighted Chebyshev method can find optimal solutions in convex and nonconvex regions of the Pareto front.

7.3 Performance Assessment

In this section, the performance of DDMOA2 is investigated. First, the algorithm is compared with its predecessor on a set of two and three-objective test instances in order to study the impact of the introduced modifications. Then, the performance of DDMOA2 on many-objective problems is studied and compared with some state-of-the-art many-objective optimizers. Once again, the main feature under concern is the proposed hybrid reproduction operator and its viability within a new algorithmic framework adopted by DDMOA2.

7.3.1 Preliminary Experiments

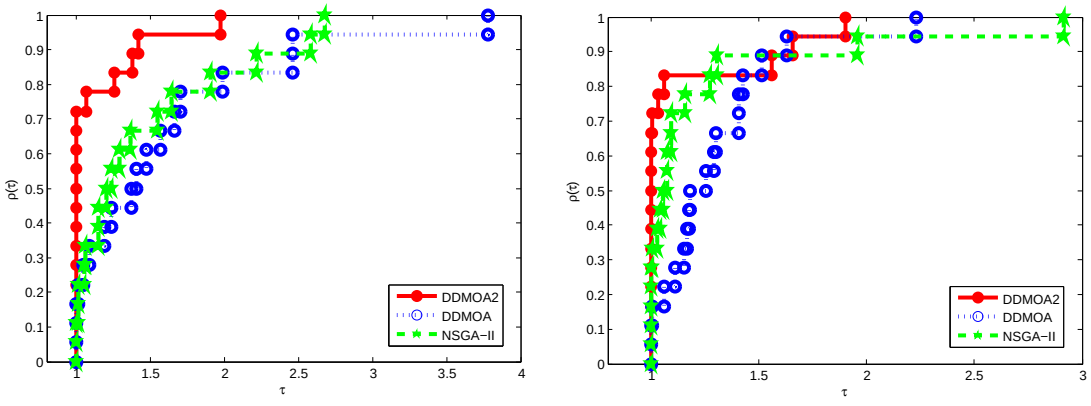
In this study, DDMOA2 is compared with its predecessor DDMOA and state-of-the-art EMO algorithm NSGA-II [46] on the WFG test suite [88]. The outcomes of the algorithms are assessed using the GD and IGD indicators. Furthermore, the overall performance of the algorithms is shown using performance profiles, and the pairwise statistical comparison is performed using the nonparametric Wilcoxon rank-sum test performed at the significance level of $\alpha = 0.05$.

Experimental Setup

All algorithms are implemented in the MATLAB® programming language and run on the two and three-objective WFG1-9 test problems with 10 decision variables. For each algorithm, 30 independent runs are performed on each problem with a population size of $\mu = 200$, running for 30,000 function evaluations. The further parameter settings for the algorithms are as follows. NSGA-II is adopted with the crossover and mutation distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively, as well as the crossover probability of $p_c = 0.9$ and the mutation probability of $p_m = 1/n$ (where n is the number of decision variables). The default parameters used in DDMOA and DDMOA2 are: the initial step size for local search $\delta^{(0)} = 1$, the initial step size for reproduction $\sigma^{(0)} = 5$, the number of subpopulations $\alpha = 5$, and the tolerance for step size $\delta_{\text{tol}} = 10^{-3}$.

Experimental Results

The overall performance with respect to I_{GD} and I_{IGD} on all considered test problems can be observed in Figure 7.1. The plots presented in this figure show that DDMOA2 is the best algorithm in terms of accuracy and robustness, regarding the median values of the GD indicator (Figure 7.1(a)) and the IGD indicator (Figure 7.1(b)).



(a) median values of I_{GD} (b) median values of I_{GD}

Figure 7.1: Performance profiles on the median values of quality indicators.

	DDMOA2	DDMOA	NSGA-II
	2-objectives		
WFG1	0.0052	**0.0048**$^{\text{I,III}}$	0.0051
WFG2	0.0183$^{\text{II}}$	0.0502	**0.0133**$^{\text{I,II}}$
WFG3	**0.0037**$^{\text{II,III}}$	0.0052	0.0047
WFG4	0.0191	0.0192	**0.0096**$^{\text{I,II}}$
WFG5	**0.0630**$^{\text{II,III}}$	0.0640	0.0639
WFG6	**0.0265**$^{\text{III}}$	0.0288$^{\text{III}}$	0.0709
WFG7	0.0051	**0.0040**$^{\text{I}}$	0.0041$^{\text{I}}$
WFG8	**0.1892**$^{\text{III}}$	0.1984	0.2272
WFG9	0.0177	**0.0125**$^{\text{I}}$	0.0132$^{\text{I}}$
	3-objectives		
WFG1	**0.0459**$^{\text{II,III}}$	0.0720	0.0709
WFG2	**0.0845**$^{\text{II,III}}$	0.2080	0.1388$^{\text{II}}$
WFG3	**0.2006**$^{\text{II,III}}$	0.4940$^{\text{III}}$	0.5185
WFG4	**0.0867**$^{\text{II,III}}$	0.1278	0.1186$^{\text{II}}$
WFG5	**0.0995**$^{\text{II,III}}$	0.1362	0.1228$^{\text{II}}$
WFG6	**0.0762**$^{\text{II,III}}$	0.1268$^{\text{III}}$	0.1688
WFG7	**0.0733**$^{\text{II,III}}$	0.0908	0.0843$^{\text{II}}$
WFG8	**0.2090**$^{\text{II,III}}$	0.3556$^{\text{III}}$	0.3978
WFG9	**0.0721**$^{\text{II,III}}$	0.0858	0.0829$^{\text{II}}$

Table 7.1: Statistical comparison in terms of I_{GD}.

	DDMOA2	DDMOA	NSGA-II
	2-objectives		
WFG1	**0.007**$^{\text{II,III}}$	0.0081	0.0076$^{\text{II}}$
WFG2	0.0169$^{\text{II}}$	0.0267	**0.0164**$^{\text{II}}$
WFG3	**0.0067**$^{\text{II,III}}$	0.0094	0.0085$^{\text{II}}$
WFG4	0.0149$^{\text{II}}$	0.0175	**0.0078**$^{\text{I,II}}$
WFG5	**0.0649**$^{\text{II,III}}$	0.0653	0.0653
WFG6	0.0243$^{\text{III}}$	**0.0243**$^{\text{III}}$	0.0709
WFG7	**0.0075**$^{\text{II,III}}$	0.0083	0.0080$^{\text{II}}$
WFG8	0.1373	**0.1366**	0.1495
WFG9	0.0125	0.0125	**0.0118**
	3-objectives		
WFG1	0.2252	0.1757$^{\text{I}}$	**0.1359**$^{\text{I,II}}$
WFG2	0.2469	0.2256$^{\text{I}}$	**0.1580**$^{\text{I,II}}$
WFG3	**0.0258**$^{\text{II,III}}$	0.0390$^{\text{III}}$	0.0504
WFG4	**0.1840**$^{\text{II,III}}$	0.2169	0.1980$^{\text{II}}$
WFG5	**0.2075**$^{\text{II,III}}$	0.2447	0.2189$^{\text{II}}$
WFG6	**0.1921**$^{\text{II,III}}$	0.2706	0.2512$^{\text{II}}$
WFG7	**0.1882**$^{\text{II,III}}$	0.2447	0.1938$^{\text{II}}$
WFG8	**0.4607**$^{\text{II,III}}$	0.5790	0.5331$^{\text{II}}$
WFG9	**0.1895**$^{\text{II,III}}$	0.2208	0.1974$^{\text{II}}$

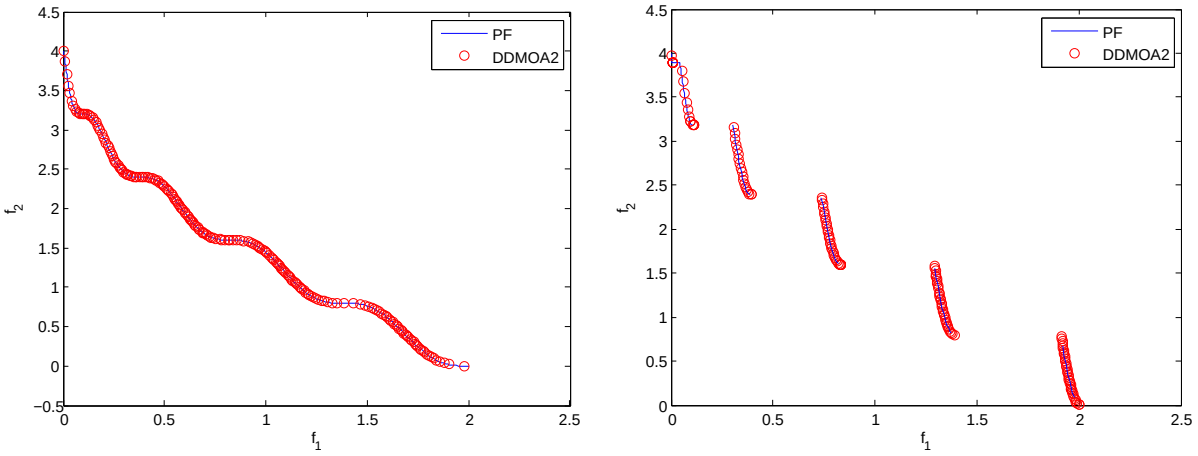
Table 7.2: Statistical comparison in terms of I_{IGD}.

Tables 7.1 and 7.2 show the median values of the generational distance indicator and the inverted generational distance indicator, respectively, over 30 runs. The superscripts I, II and III indicate whether the respective algorithm performs significantly better than DDMOA2, DDMOA, and NSGA–II, respectively. The best value in each row is marked bold (the lower the better). For the two-objective problems, DDMOA2 achieves the best

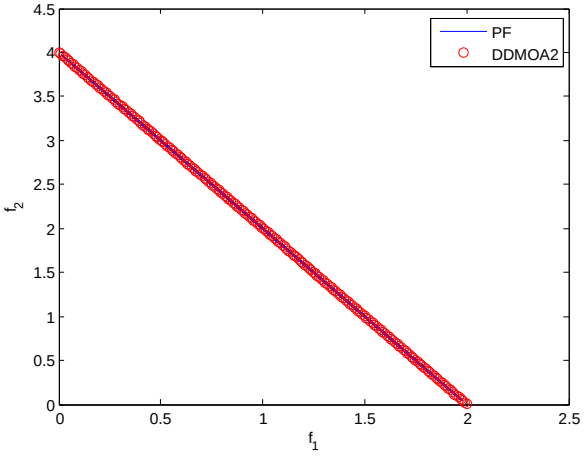
values of both quality indicators on the four test problems, always being significantly better than NSGA–II. On the majority of the problems DDMOA2 is significantly better than DDMOA. With respect to the three-objective problems, the new DDMOA2 completely outperforms the other algorithms regarding I_{GD}. Concerning I_{IGD}, DDMOA2 is significantly better on 7 out of 9 WFG test problems.

From the discussed results, it becomes clear that the new DDMOA2 performs generally better than its predecessor DDMOA. The achieved outperformance can be attributed to the fact that in DDMOA2 the population is constituted not only of nondominated solutions as well as all population members can generate offspring. Furthermore, it is shown that the proposed approach based on finding descent directions only for two randomly chosen objectives is able to guide efficiently the search.

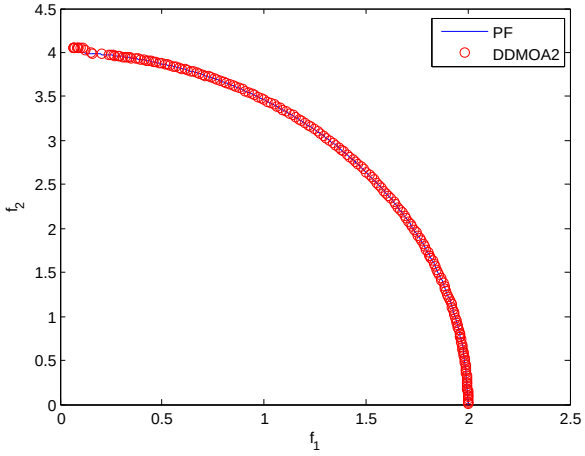
Additionally, Figures 7.2 and 7.3 show approximation sets with the best values of I_{IGD} obtained by DDMOA2 on the two and three-objective WFG test problems. It can be seen that DDMOA2 is able to obtain adequate Pareto front approximations on the majority of the two-objective test problems (Figures 7.2). However, on the WFG5 problem, the approximation set is not close enough to the true Pareto front (Figure 7.2(e)). Also, a poor performance is observed on the WGF8 test problem (Figure 7.2(h)). This is a difficult nonseparable problem [88], on which many algorithms fail to obtain a good Pareto



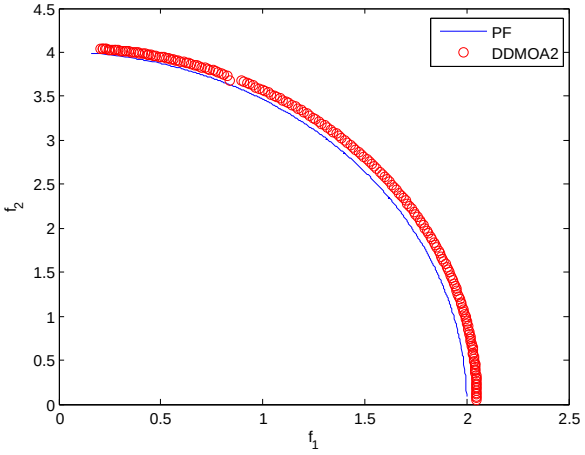
(a) WFG1 (b) WFG2



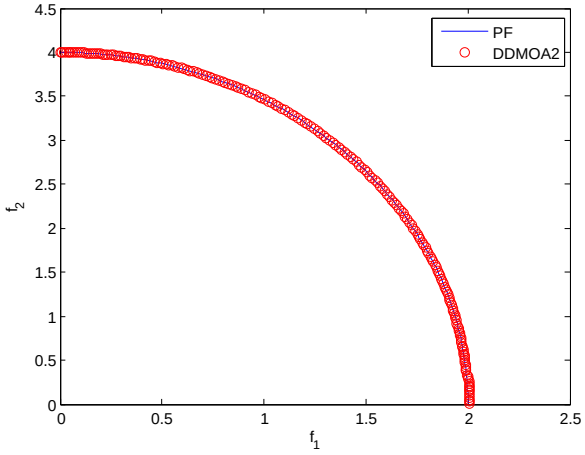
(c) WFG3



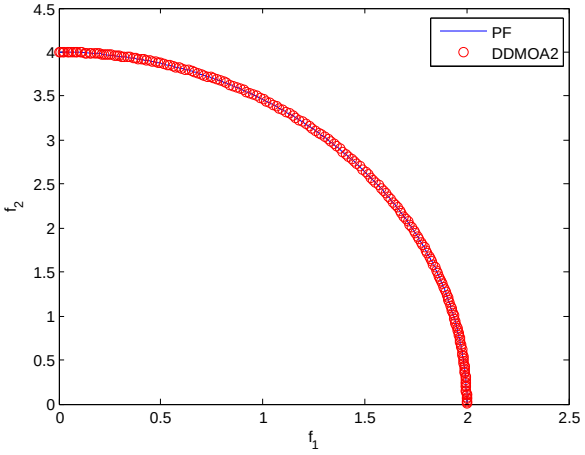
(d) WFG4



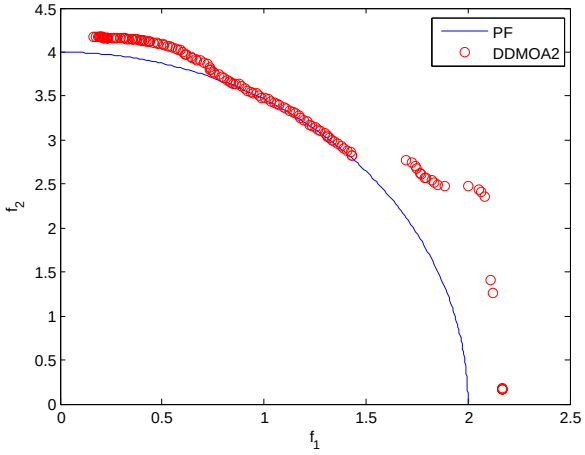
(e) WFG5



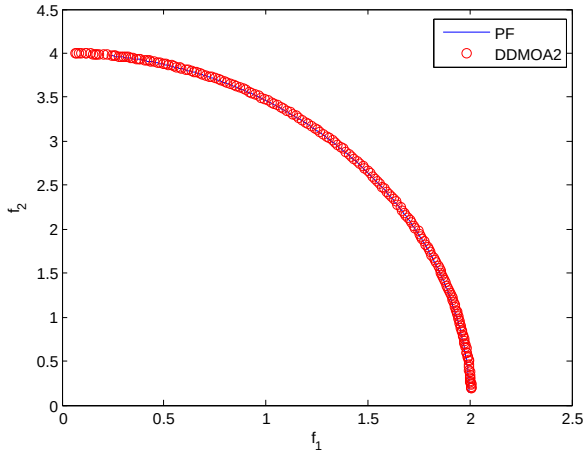
(f) WFG6



(g) WFG7



(h) WFG8

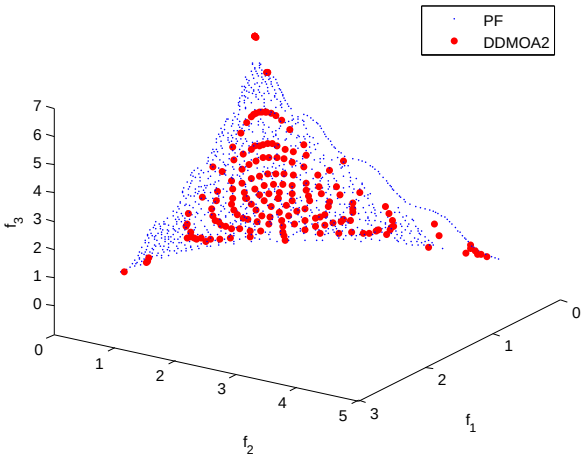


(i) WFG9

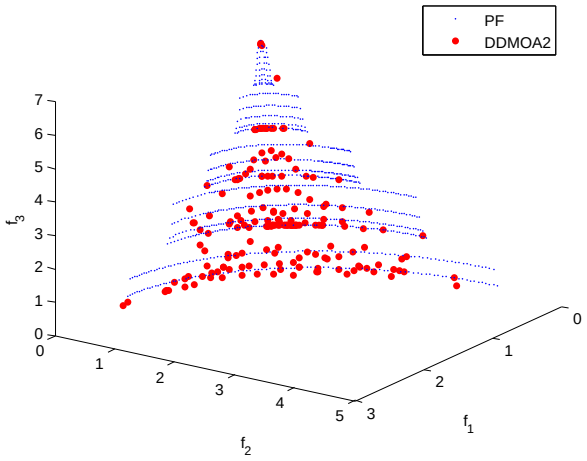
Figure 7.2: Performance of DDMOA2 on the two-objective WFG test suite.

front approximation.

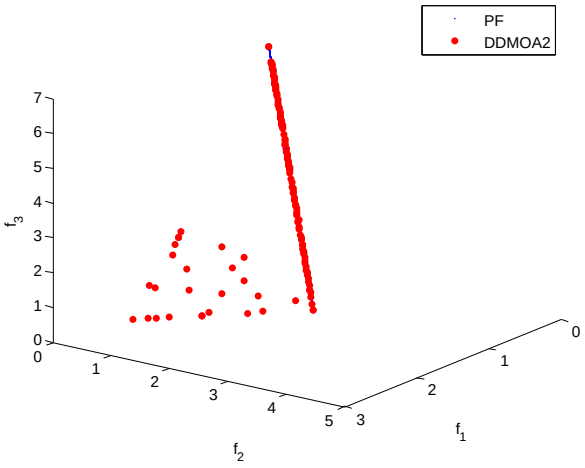
Concerning the three-objective problems (Figures 7.3), DDMOA2 is able to obtain adequate approximations on all of the test instances. However, from Figure 7.3(c), it can be seen that some solutions did not converge due to the presence of redundant nondominated solutions. This difficulty is often encountered on problems with a degenerate Pareto front. Also, there are uncovered regions of the Pareto front for WFG8 (Figure 7.3(h)).



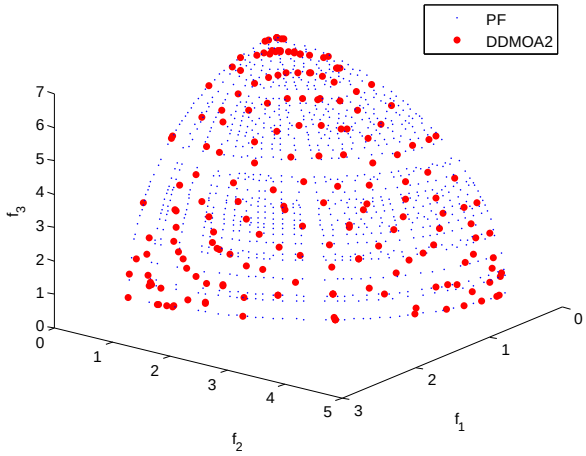
(a) WFG1



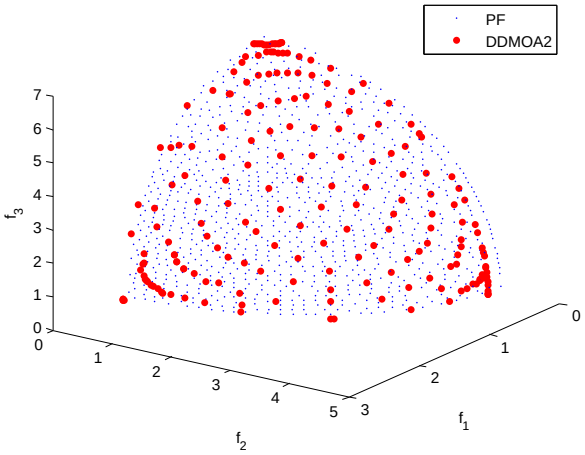
(b) WFG2



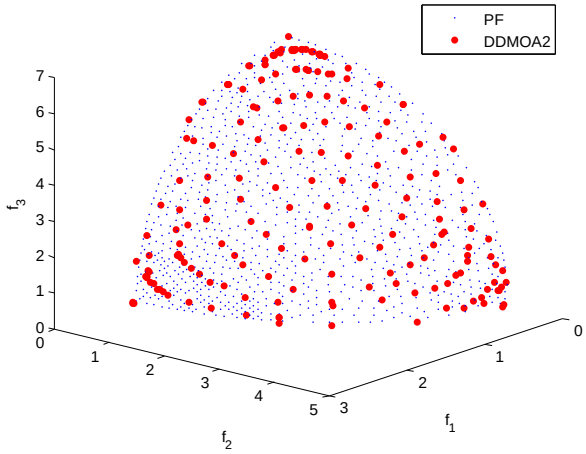
(c) WFG3



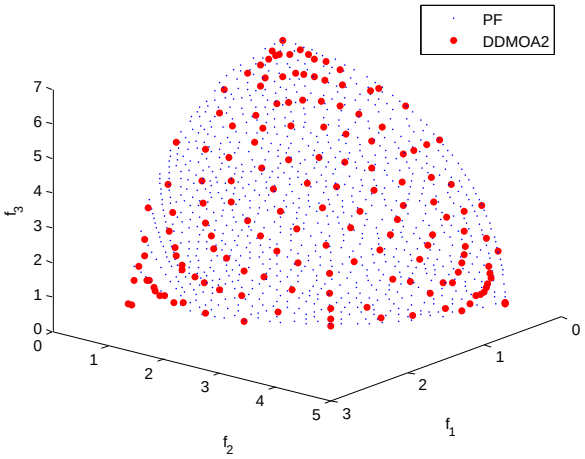
(d) WFG4



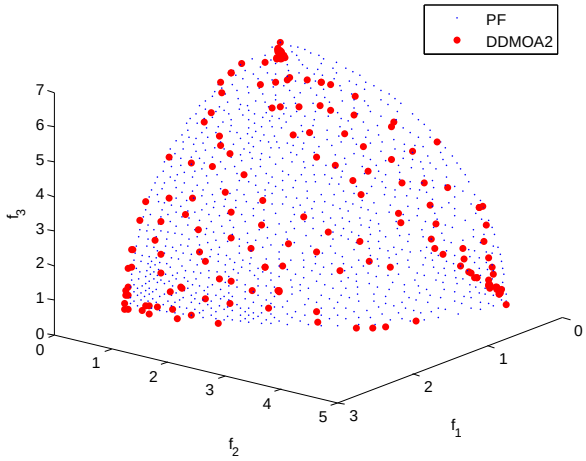
(e) WFG5



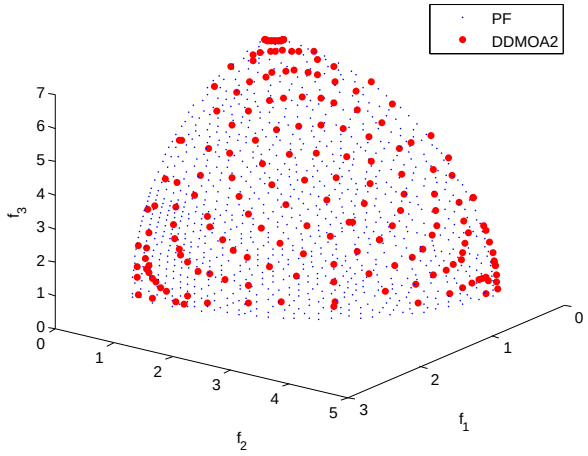
(f) WFG6



(g) WFG7



(h) WFG8



(i) WFG9

Figure 7.3: Performance of DDMOA2 on the three-objective WFG test suite.

7.3.2 Intermediate Summary

In the above discussed experiments, the performance of a generalized descent directions-guided multiobjective algorithm DDMOA2 is investigated and compared with its predecessor DDMOA. Both algorithms employ the same reproduction operator and general principles for computing descent directions. However, they differ significantly in some other components.

DDMOA2 does not rely on the concept of the Pareto dominance. Its parent and environmental selections are based on the scalarizing fitness assignment. DDMOA2 always maintains a number of individuals in the population that is not less than μ. This leads to the need for selecting promising individuals in the population to apply a local search procedure. Contrary to DDMOA, DDMOA2 uses the local search procedure to find descent directions only for two randomly chosen objectives. Furthermore, in DDMOA2 all individuals in the population have a chance to produce offspring.

The experimental results show that DDMOA2 generally outperforms its predecessor DDMOA, regarding the convergence to the Pareto front and the diversity of the obtained solutions. At the same time, DDMOA2 performs better than representative state-of-the-art NSGA–II in terms of both considered quality indicators.

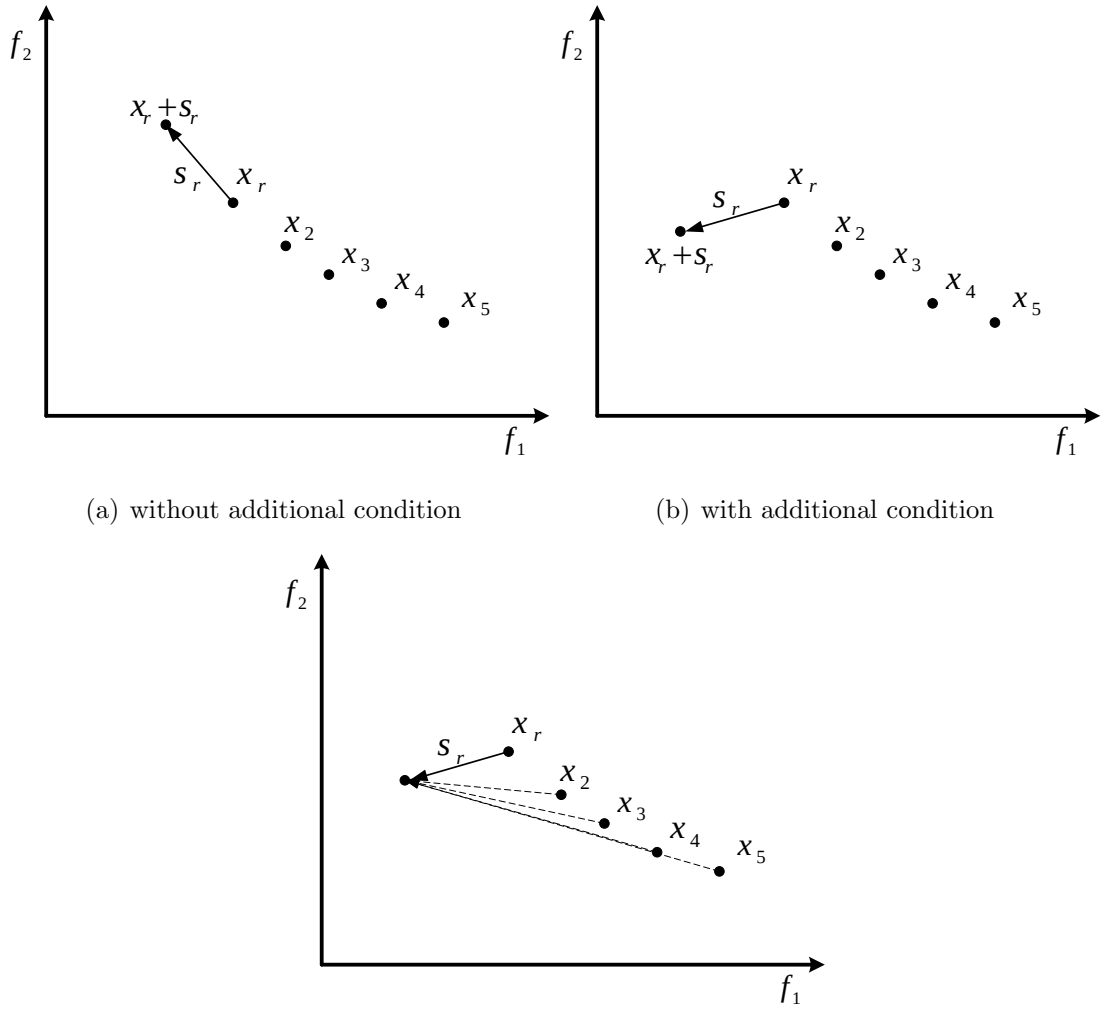
7.3.3 Further Experiments

In the following, the performance of DDMOA2 is studied on many-objective problems. The introduction of a scalarizing fitness assignment into the framework of descent directions-guided multiobjective algorithm is a major motivation for solving this sort of problems. The performance of DDMOA2 is compared with that produced by the state-of-the-art many-objective optimizers IBEA [198] and MOEA/D [123] on the DTLZ test suite [50] with 30 decision variables having between 2 and 8 objectives.

Experimental Setup

DDMOA2 and MOEA/D are implemented in the MATLAB® programming language, IBEA is used within the PISA [16] framework. For each algorithm, 30 independent runs are performed on each problem with a population size of $\mu = 200$, running for 60,000 function evaluations. The further parameter settings for the algorithms are as follows. IBEA is adopted with the crossover and mutation distribution indexes of $\eta_c = 20$ and $\eta_m = 20$, respectively, as well as the crossover probability of $p_c = 0.9$ and the mutation probability of $p_m = 1/n$ (where n is the number of decision variables). The control parameters used in MOEA/D are: the number of weight vectors in the neighborhood of each weight vector $T = 20$, the probability that the parent selected from the neighborhood $\delta = 0.9$, the maximum number of solutions replaced by each child solution $n_r = 2$. Settings for the DE operator are: $CR = 0.15$ and $F = 0.5$, the mutation probability of $p_m = 1/n$ and the mutation distribution index of $\eta_m = 20$ are used in the polynomial mutation operator. The parameters used in DDMOA2 are: the initial step size for local search $\delta^{(0)} = 0.4$, the initial step size for reproduction $\sigma^{(0)} = 5$, the number of subpopulations $\alpha = 5$, and the tolerance for step size $\delta_{\text{tol}} = 10^{-3}$.

Dealing with many-objective problems often requires an additional search pressure either in the decision or the objective space. The scalarizing fitness assignment used by DDMOA2 ensures the selection pressure during the evolutionary process. To increase the search pressure in the decision space, an additional condition for the acceptance of descent



(a) without additional condition

(b) with additional condition

(c) resulted descent directions

Figure 7.4: Computation of descent directions.

directions is introduced. Thus, descent direction $\boldsymbol{s}_{\mathrm{r}}$ for representative $\boldsymbol{x}_{\mathrm{r}}$ is accepted during local search if trail solution $\boldsymbol{x}_{\mathrm{r}} + \boldsymbol{s}_{\mathrm{r}}$ is not worse in all objectives than $\boldsymbol{x}_{\mathrm{r}}$, and direction $\boldsymbol{s}_{\mathrm{r}}$ leads to the decrease of the corresponding objective (objective for which descent direction is calculated).

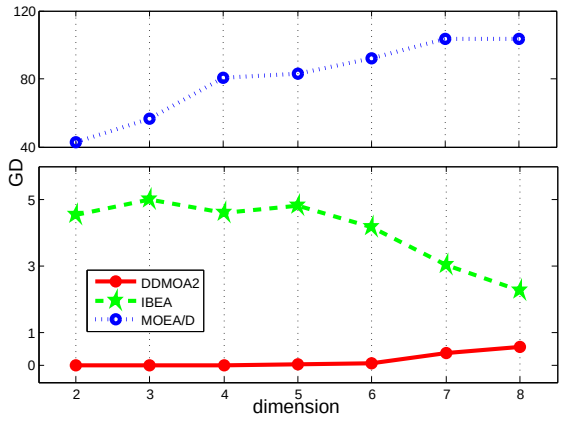
Figure 7.4 allows to better understand this procedure. In which a subpopulation is presented and local search is used to find descent directions with respect to f_1. Simply calculating the descent direction $\boldsymbol{s}_{\mathrm{r}}$ for subpopulation representative $\boldsymbol{x}_{\mathrm{r}}$, a descent direction

shown in Figure 7.4(a) can be found. Although this direction leads to a decrease with respect to f_1, it does not lead to a promising region of the search space. Imposing the condition that trail solution $\boldsymbol{x}_r + \boldsymbol{s}_r$ cannot be worse in all objectives and it must be better than $\boldsymbol{x}_r$ in f_1, the possibility of obtaining a descent direction shown in Figure 7.4(a) is excluded. An example of a descent direction that satisfies such condition is presented in Figure 7.4(b). Further, descent directions calculated for all the other individuals in the subpopulation are shown in Figure 7.4(c).

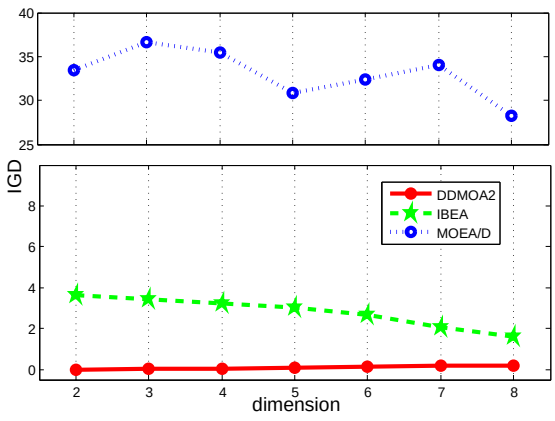
Experimental Results

Figure 7.5 presents the graphical representation of the median values of GD and IGD obtained by all the tested algorithms on the DTLZ1-7 test problems with varying dimensions. The plots allow to analyze the behavior of the algorithms on problems with different characteristics and distinct dimensions.

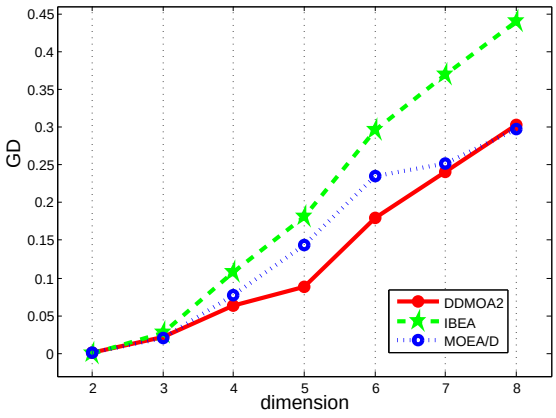
Observing Figures 7.5(a), 7.5(b), 7.5(e), and 7.5(f), one can conclude that DDMOA2 performs better with respect to both quality indicators on multimodal problems (DTLZ1 and DTLZ3). This result is achieved due to its reproduction operator, which allows to escape from locally optimal regions and efficiently generates promising individuals. This is especially noticeable in smaller dimensions. In higher dimensions, the performance curve of IBEA becomes closer to the one of DDMOA2, but from the plots it can be seen



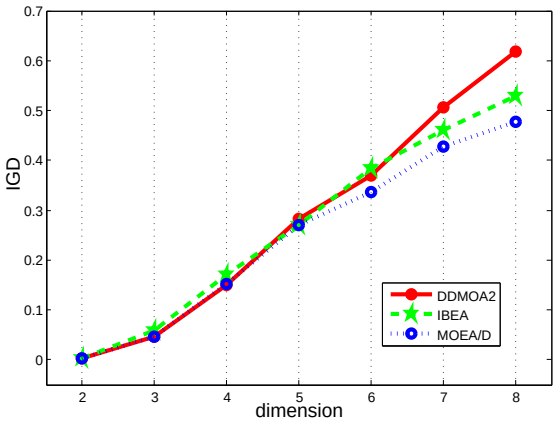
(a) DTLZ1



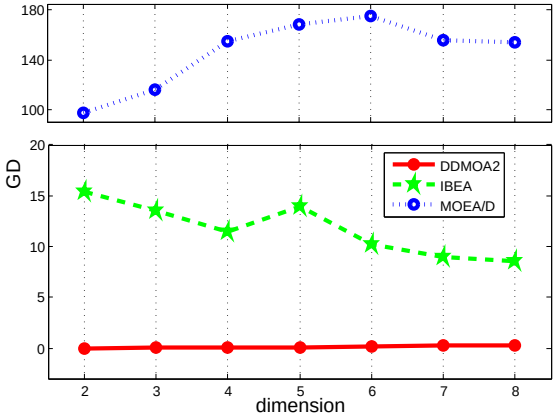
(b) DTLZ1



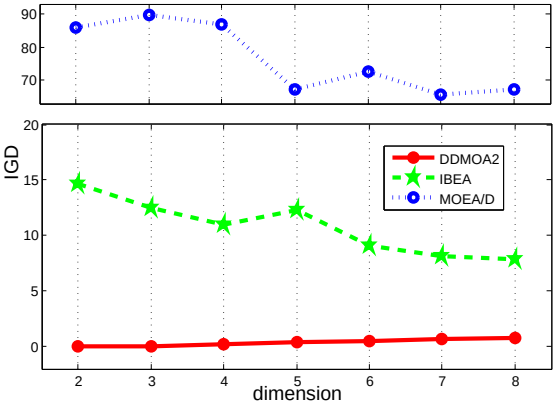
(c) DTLZ2



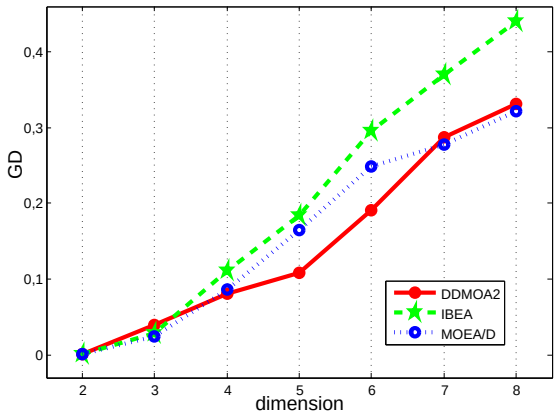
(d) DTLZ2



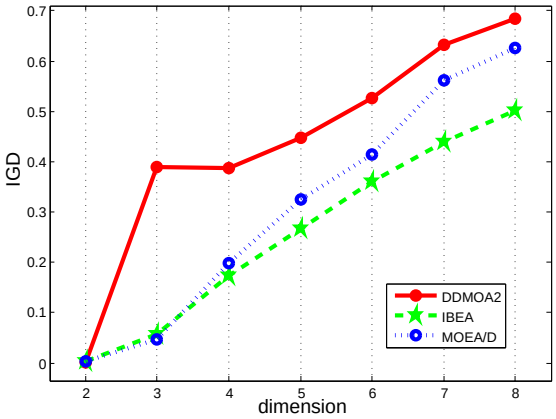
(e) DTLZ3



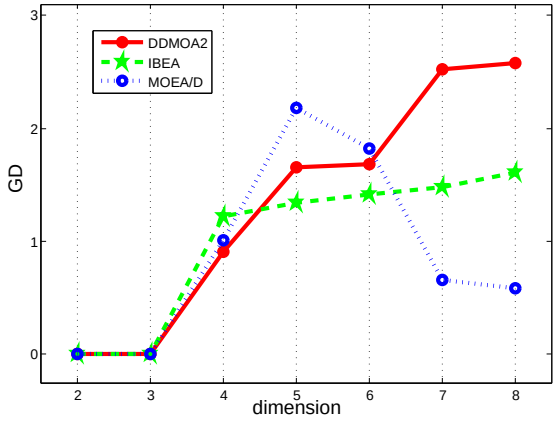
(f) DTLZ3



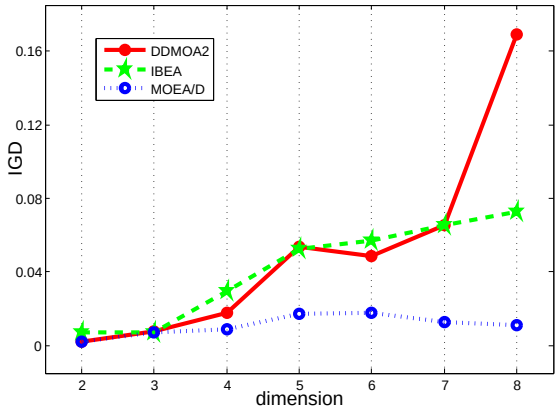
(g) DTLZ4



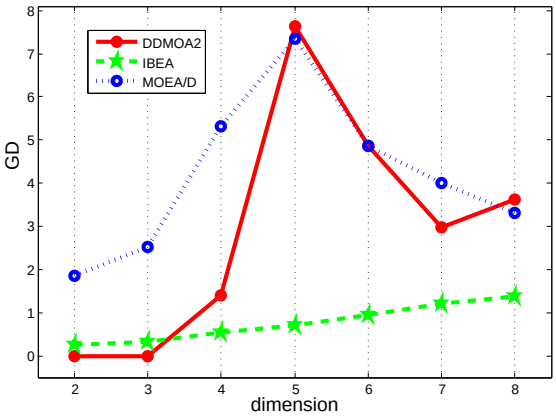
(h) DTLZ4



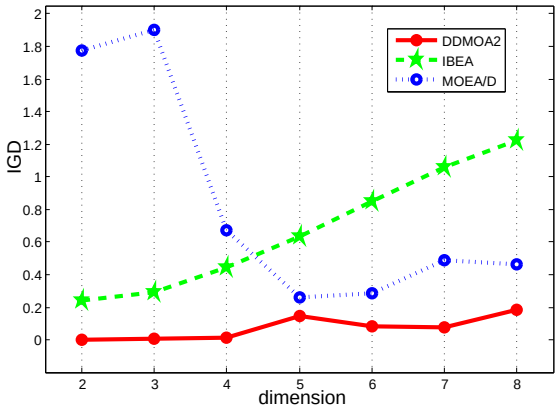
(i) DTLZ5



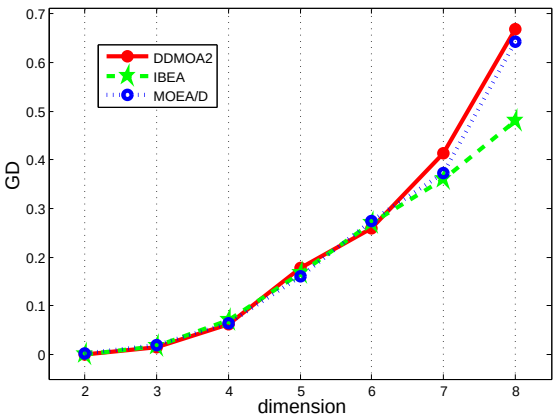
(j) DTLZ5



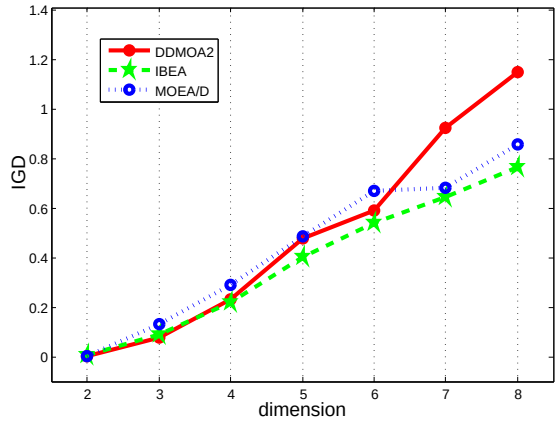
(k) DTLZ6



(l) DTLZ6



(m) DTLZ7



(n) DTLZ7

Figure 7.5: Performance comparison of DDMOA2, IBEA, and MOEA/D on the DTLZ test suite. The plots present the median values of GD (left hand side) and IGD (right hand side) over 30 runs.

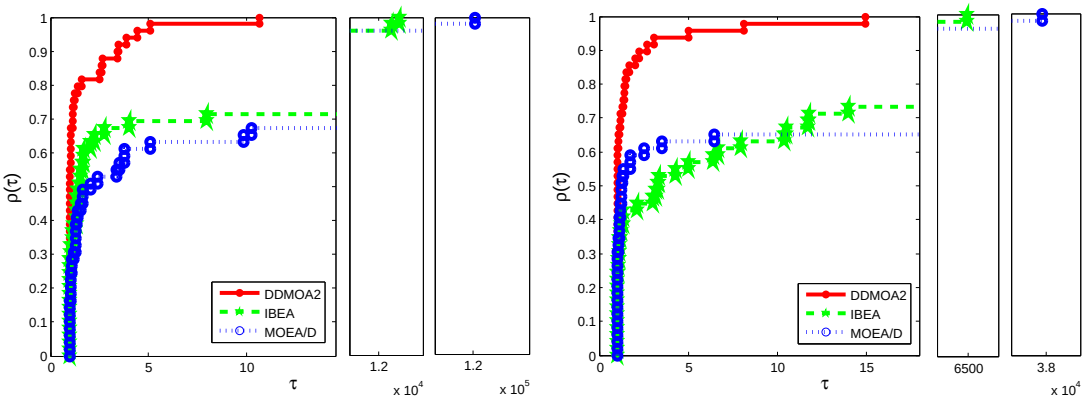
that DDMOA2 still clearly outperforms the other algorithms with respect to both quality indicators.

DTLZ2 and DTLZ4 test problems do not present as much difficulties in terms of convergence to the Pareto front as the above considered problems (DTLZ1 and DTLZ3). In Figures 7.5(c), 7.5(d), 7.5(g), 7.5(h), one can see that DDMOA2 provides competitive results with the representative state-of-the-art algorithms. However, the deterioration of performance can be observed in dimensions higher than 6, and a slight loss with respect to IGD.

In Figures 7.5(i), 7.5(j), 7.5(k), 7.5(l), abrupt changes in the performance of the algorithms can be observed. This fact can be explained by the presence of redundant objectives in DTLZ5 and DTLZ6 test problems. Dimensionality reduction techniques are suitable for solving such sort of problems in higher dimensions. Nevertheless, the competitive performance of DDMOA2 on DTLZ6 can be observed especially with respect to IGD.

Also, highly competitive results can be observed on DTLZ7 test problem with respect to both quality indicators up to 6 dimensions (Figures 7.5(m) and 7.5(n)). This problem does not present a lot of difficulties in terms of convergence, and its main characteristic is the discontinuity of the Pareto optimal front.

Figure 7.6 represents performance profiles on the median values of GD and IGD. The



(a) GD (b) IGD

Figure 7.6: Performance profiles on the median values of quality indicators.

	DDMOA2	IBEA	MOEA/D
2-objectives			
DTLZ1	II,III	III	
DTLZ2	III	I,III	
DTLZ3	II,III	III	
DTLZ4	III	I,III	
DTLZ5	III	I,III	
DTLZ6	II,III	III	
DTLZ7	III	I,III	
3-objectives			
DTLZ1	II,III	III	
DTLZ2	II		II
DTLZ3	II,III	III	
DTLZ4			II
DTLZ5	III	I,III	
DTLZ6	II,III	III	
DTLZ7	II,III	III	
4-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4	II,III		II
DTLZ5	II,III		II
DTLZ6	III	I,III	
DTLZ7	II,III		II
5-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4	II,III		II
DTLZ5		I,III	
DTLZ6		I,III	
DTLZ7		I	I
6-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4	II,III		II
DTLZ5		III	
DTLZ6		I,III	
DTLZ7	II,III	III	
7-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4	II		I,II
DTLZ5		I	I,II
DTLZ6	III	I,III	
DTLZ7		I,III	I
8-objectives			
DTLZ1	II,III	III	
DTLZ2	II		II
DTLZ3	II,III	III	
DTLZ4	II		I,II
DTLZ5		I	I,II
DTLZ6		I,III	
DTLZ7		I,III	

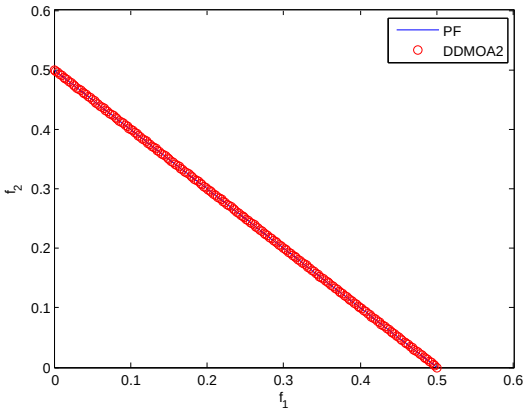
Table 7.3: Comparison in terms of GD.

	DDMOA2	IBEA	MOEA/D
2-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4	II,III		II
DTLZ5	II,III		II
DTLZ6	II,III	III	
DTLZ7	II,III		II
3-objectives			
DTLZ1	II,III	III	
DTLZ2	II		II
DTLZ3	II,III	III	
DTLZ4			I,II
DTLZ5		I,III	I
DTLZ6	II,III	III	
DTLZ7	II,III	III	
4-objectives			
DTLZ1	II,III	III	
DTLZ2	II,III		II
DTLZ3	II,III	III	
DTLZ4		I,III	I
DTLZ5	II		I,II
DTLZ6	II,III	III	
DTLZ7	III	I,III	
5-objectives			
DTLZ1	II,III	III	
DTLZ2		I	I
DTLZ3	II,III	III	
DTLZ4		I,III	I
DTLZ5			I,II
DTLZ6	II,III		II
DTLZ7		I,III	
6-objectives			
DTLZ1	II,III	III	
DTLZ2			I,II
DTLZ3	II,III	III	
DTLZ4		I,III	I
DTLZ5			I,II
DTLZ6	II,III		II
DTLZ7		I,III	
7-objectives			
DTLZ1	II,III	III	
DTLZ2		I	I,II
DTLZ3	II,III	III	
DTLZ4		I,III	I
DTLZ5			I,II
DTLZ6	II,III		II
DTLZ7		I	I
8-objectives			
DTLZ1	II,III	III	
DTLZ2		I	I,II
DTLZ3	II,III	III	
DTLZ4		I,III	I
DTLZ5		I	I,II
DTLZ6	II,III		II
DTLZ7		I,III	I

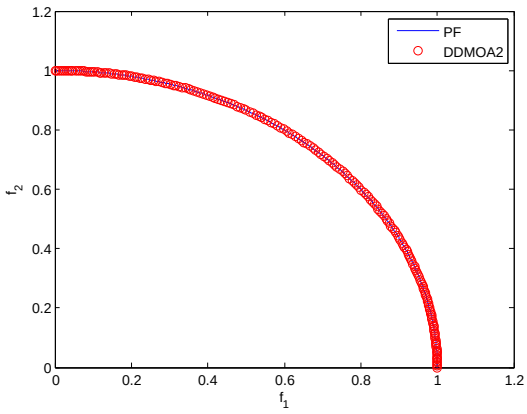
Table 7.4: Comparison in terms of IGD.

figure allows to get insights about the overall performance of the algorithms on all the tested problems. From Figures 7.6(a) and 7.6(b), one can conclude that DDMOA2 demonstrates the overall better performance when compared with the other algorithms, being the most robust algorithm with respect to both quality indicators.

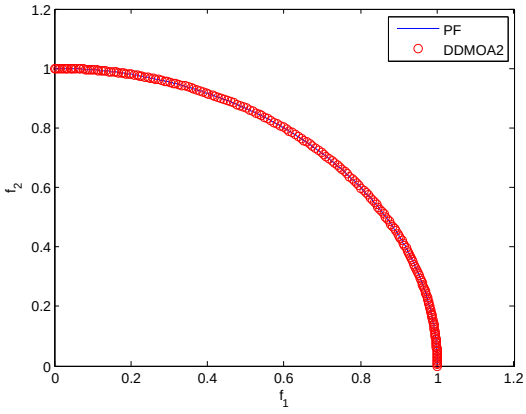
Furthermore, the pairwise statistical comparison of the algorithms with respect to GD and IGD is provided in Table 7.3 and Table 7.4, respectively. In the tables, the symbols I, II, and III indicate whether the respective algorithm performs significantly better than DDMO2, IBEA, and MOEA/D, respectively. Note that in all dimensions DDMOA2 performs statistically better on the DTLZ1 and DTLZ3 problems with regard to the GD and IGD indicators. This highlights the superiority of DDMOA2 on multimodal problems.



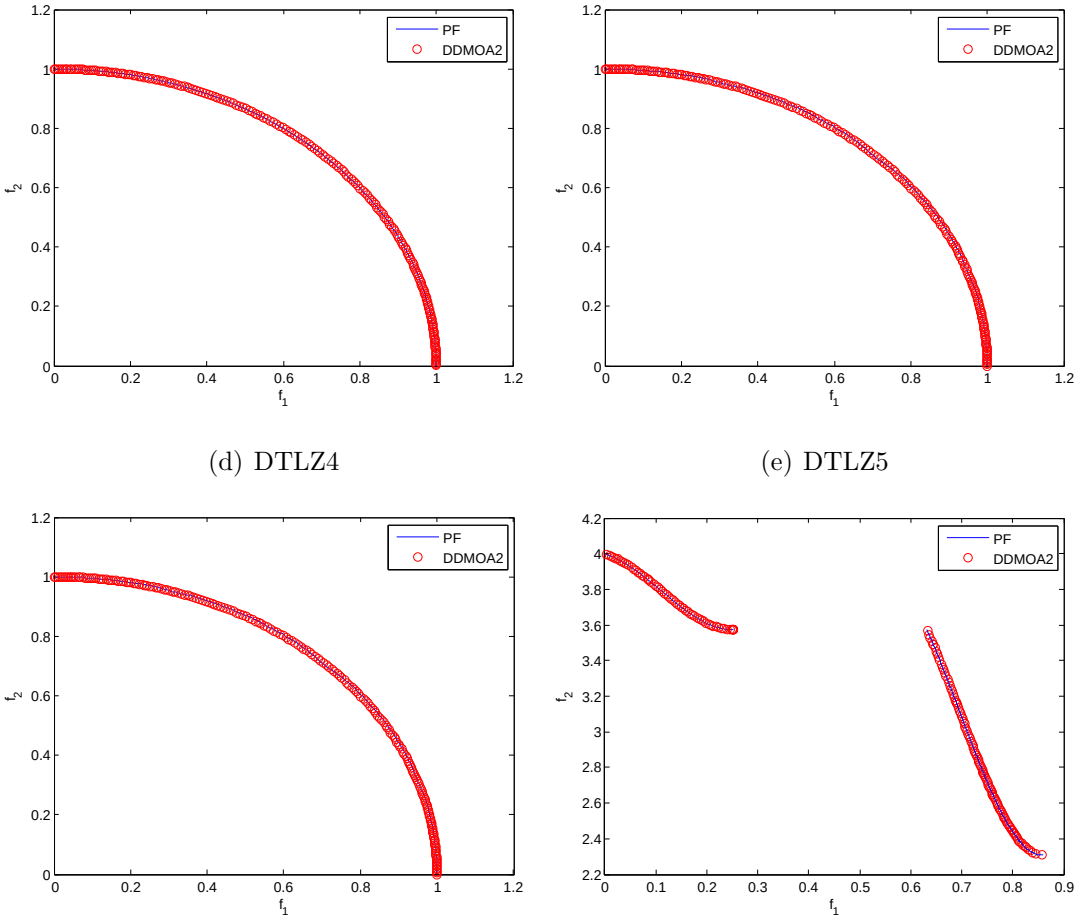
(a) DTLZ1



(b) DTLZ2



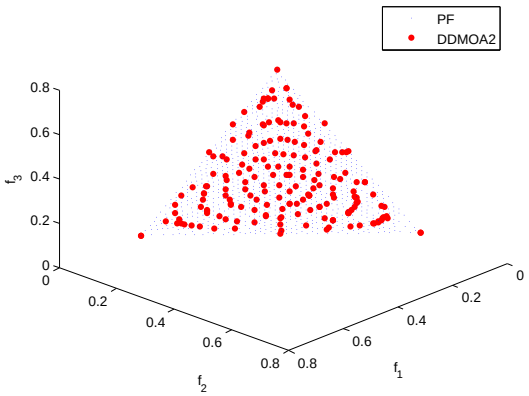
(c) DTLZ3



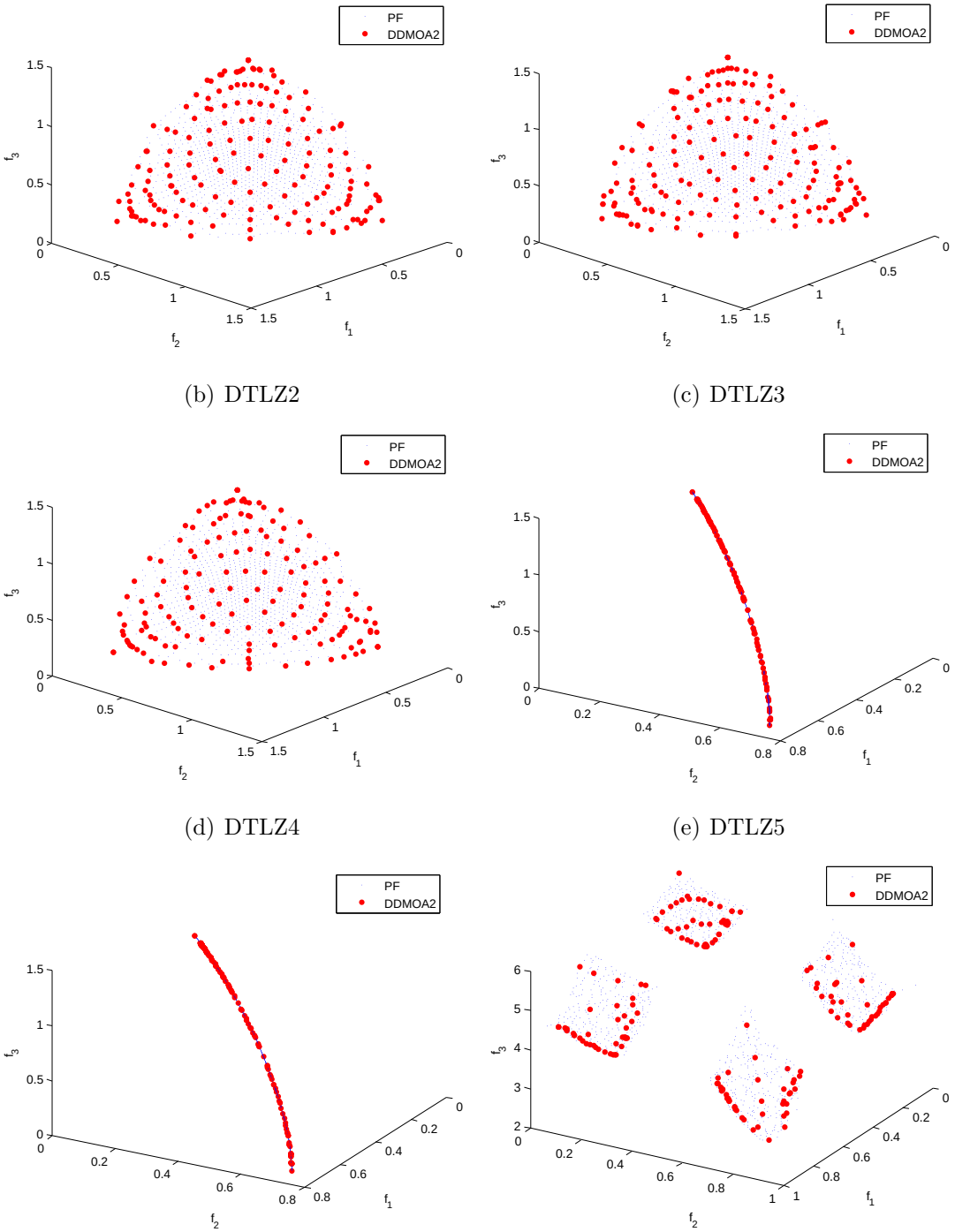
(d) DTLZ4 (e) DTLZ5

(f) DTLZ6 (g) DTLZ7

Figure 7.7: Performance of DDMOA2 on the two-objective DTLZ test suite.



(a) DTLZ1



(b) DTLZ2 (c) DTLZ3

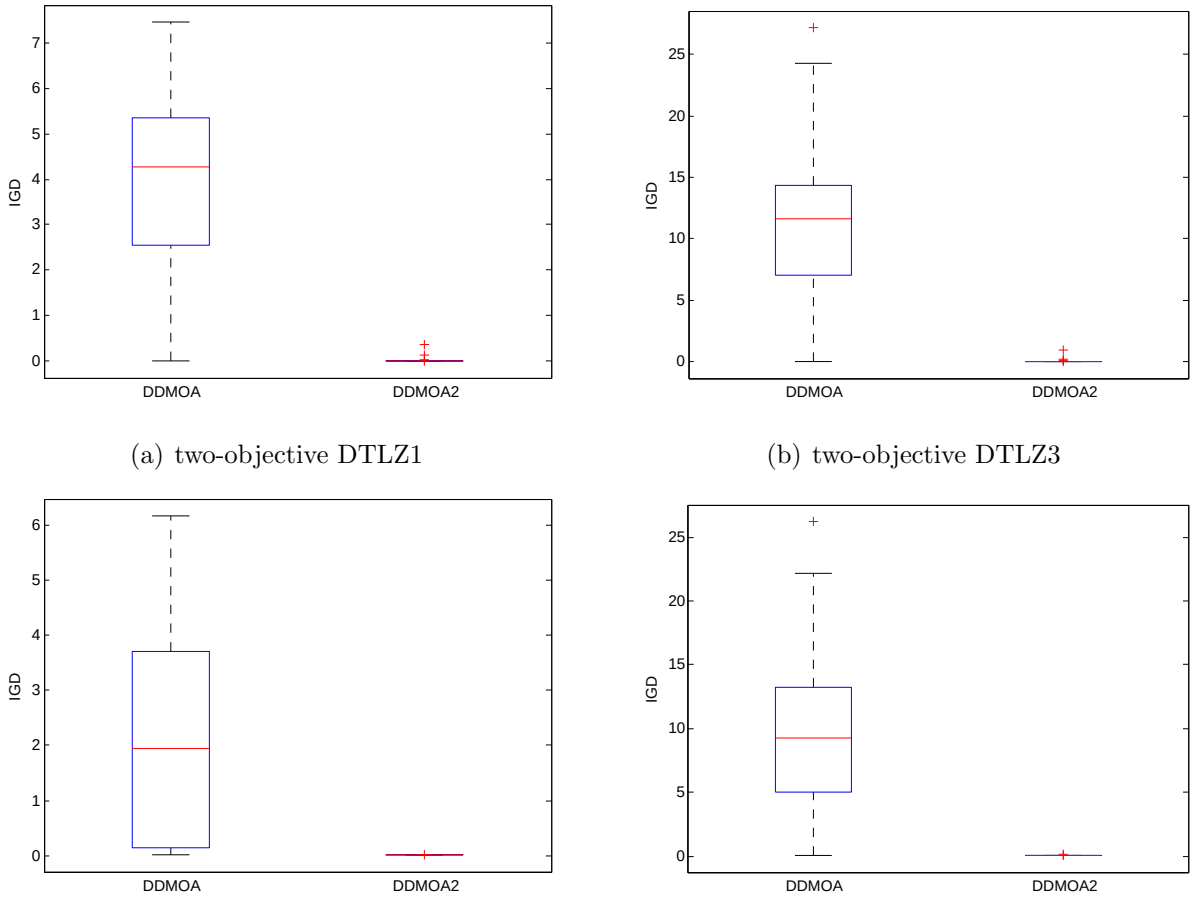
(d) DTLZ4 (e) DTLZ5

(f) DTLZ6 (g) DTLZ7

Figure 7.8: Performance of DDMOA2 on the three-objective DTLZ test suite.

Additionally, approximation sets with the best values of IGD obtained on the two and three-objective test problems are shown in Figures 7.7 and 7.8. From the presented plots it can be seen that DDMOA2 is able to converge and provide an adequate distribution of solutions along the Pareto front for all the two and three-objective DTLZ test problems.

To emphasize the difference between DDMOA and DDMOA2, the IGD indicator values obtained by both algorithms on multimodal DTLZ1 and DTLZ3 with two and three objectives are shown in Figure 7.9. From plots presented in this figure, a drastic performance improvement can be readily observed. Although DDMOA is able to obtain comparable indicator values on these problems, the variability of the distributions of these values are



(a) two-objective DTLZ1

(b) two-objective DTLZ3

(c) three-objective DTLZ1

(d) three-objective DTLZ3

Figure 7.9: Performance analysis of DDMOA and DDMOA2.

completely different. Although in some cases it may be interesting to compare the best indicator values, here this comparison is not carried out, as performance based on extreme values should be read with care, if not at all avoided [33]. Here the main focus is on the variability of the two distributions. As for real world problems, especially with computationally expensive functions, it is more important to have an algorithm that exhibits a higher average performance and small variability, than an algorithm which converges occasionally to the optimal region. From this perspective, it is clear that DDMOA2 is superior compared to its predecessor. The high improvement in the variability observed in the above plots is achieved by more extensive exploration of the search space performed by DDMOA2. In DDMOA2, this exploration is due to the fact that all individuals in the current population can produce offspring.

7.4 Summary

This chapter presents a generalized descent directions-guided multiobjective algorithm (DDMOA2), which is developed on the basis of its predecessor DDMOA. Although DDMOA2 inherits the basic principles of reproduction from its predecessor, it is generalized in some ways. DDMOA2 maintains the population that can consist of nondominated as well as dominated individuals. Each individual in the population has a chance to produce offspring. This way, a better exploration of the search space is achieved, as individuals being not promising at some stages of the search can significantly benefit to the search as the evolution goes on. This feature is especially desirable for dealing with multimodal problems. Furthermore, DDMOA2 does not rely on the Pareto dominance relation during the selection process, instead the scalarizing fitness assignment is incorporated into the parent and environmental selection procedures. The presence of nondominated solutions in the population leads to the need for differentiation of individuals in the population. Thus, in DDMOA2 the population consists of leader and non-leader individuals. The selection of leaders is based on the scalarizing fitness assignment.

The performed experimental studies show the viability of the proposed approach.

DDMOA2 demonstrates a significantly superior performance compared to its predecessor. This is achieved due to the introduced modifications. Moreover, DDMOA2 produces highly competitive performance when compared with some state-of-the-art EMO algorithms, even in high dimensional objective spaces. At the same time, DDMOA2 is often able to outperform other algorithms on multimodal problems. However, the performance deterioration is observed when the number of objectives is increased. This leaves open questions for further research.

In conclusion, DDMOA2 appears a robust and efficient approach for solving continuous multiobjective optimization problems, which is validated in several comparative studies with other MOEAs on a number of test problems. Further, DDMOA2 is applied to solve real-world optimization problems. Its hybrid reproduction operator is expected to be a crucial feature to successfully deal with problems under consideration.

Chapter 8

Many-Objective Optimization using Differential Evolution with Mutation Restriction

8.1 Introduction

So far in this part of the thesis the main concern has been about developing EMO algorithms with efficient reproduction operators for searching the Pareto optimal solutions. The design of such operators especially for MO can require some adaptations of the algorithmic framework. Thus, DDMOA2 differentiates the entire population into leader and non-leader individuals. Moreover, leaders are further divided into subpopulations in order to efficiently compute descent directions. In turn, DDMOA and MO-HGPSAL adopt existing approaches to perform the search in the objective space, taken advantage of their hybrid operators for generating offspring. Although an efficient reproduction operator is a crucial feature of either a single-objective or multiobjective optimization algorithm, the presence of two search spaces in MO requires to develop proper selection mechanisms to guide the search in the objective space. This becomes especially relevant to many-objective problems.

This chapter addresses the issue of fitness assignment and how to guide the search in a high-dimensional objective space. Two different selection procedures based on the nondominated sorting with IGD-based and clustering-based second sorting criteria are suggested. On the other hand, a strategy to control the mutation strength in a DE operator is proposed to handle efficiently multimodal problems.

8.2 DEMR for Multiobjective Optimization

8.2.1 NSDEMR

As it has already been stressed in this thesis, a multiobjective optimization problem inherits all properties of its single objectives. Differential evolution generates new individuals by using differences of randomly sampled pairs of individuals in the current population. This reproduction strategy allows the population to adapt to the fitness landscape of a given problem, and makes DE invariant under any rotations of the search space. This feature makes DE an attractive choice to solve continuous real-world optimization problems. However, DE often faces difficulties in solving multimodal problems. So this section addresses the issue of the convergence properties of DE on multiobjective optimization problems with a multimodal fitness landscape. A variant of a nondominated sorting differential evolution (NSDE) [96] is considered. The outline of NSDE is given by Algorithm 13.

The algorithm works as follows. First, an initial population is randomly generated. Next, for the predefined number of generations the evolutionary process is performed. In each generation, each individual in the population generates one offspring employing a DE operator (lines 9-18).

For each individual in the population, two different population members (r^1 and r^2) are selected. The difference vector $\boldsymbol{v}$ is calculated using the selected individuals. In the following, this vector is also referred as a mutation vector. Then, in order to introduce some variation to the difference vector, it is mutated using polynomial mutation with probability $p_m = 1/n$, where n is the dimensionality of the decision space. Thereafter, the resulting

Algorithm 13 NSDE

1: `initialize`: P;
2: **repeat**
3: $P' \leftarrow P$, $P'' \leftarrow \{\}$;
4: **for** $i = 1$ **to** μ **do**
5: randomly select two indexes $r^1 \neq r^2 \neq i$;
6: $\boldsymbol{v} \leftarrow \boldsymbol{x}^{r^1} - \boldsymbol{x}^{r^2}$;
7: apply polynomial mutation on $\boldsymbol{v}$;
8: restrict vector $\boldsymbol{v}$;
9: **for** $j = 1$ **to** n **do**
10: **if** $rand < CR$ **then**
11: $y_j \leftarrow x_j^i + v_j$;
12: **else**
13: $y_j \leftarrow x_j^i$;
14: **end if**
15: **end for**
16: **if** $\boldsymbol{x}^i = \boldsymbol{y}$ **then** go to 10;
17: **else** $P'' \leftarrow P'' \cup \boldsymbol{y}$;
18: **end if**
19: **end for**
20: $P \leftarrow$ `environmentalSelection` $(P' \cup P'')$;
21: **until** the stopping criterion is met
22: output: P;

difference vector is restricted (line 8) and is used to mutate the corresponding parent with probability CR.

The (If) statement in (line 16) is used to guarantee that at least one gene of the parent individual is mutated. Some preliminary experiments showed that this approach works better than the commonly used strategy in DE, where an integer jr is first generated and then the jr-th gene is mutated. Used reproduction operator is similar to the

DE/current-to-rand/1/bin variant with $K = 0$. If some genes of a newly generated offspring are out of the bounds of the variables of the problem, then the corresponding lower or upper values are assigned to these genes.

After generating the offspring population P'', this population is evaluated and combined with the parent population P'. Thereafter, `environmentalSelection` procedure is performed in order to create a new population. This procedure is the same as the one used in NSGA-II [46].

In the following, two different schemes to restrict the mutation vector and their impacts on the performance of the algorithm are investigated. Generally, EAs use some sort of step size adaptation scheme or restriction rule in order to control the mutation strength during the search. For instance, evolution strategies use 1/5th success rule [15] or cumulative step-size adaptation [80] for controlling the mutation strength. On the other hand, a variable-wise restriction mechanism was proposed to control the speed of particles in SMPSO [135]. In turn, in DE the difference vector is usually scaled by multiplying it by the parameter F, which is a constant or may be adapted during the search. This simple strategy to control the mutation strength may not be suitable for multimodal problems. The population can rapidly loss diversity and, as a result, get stuck in suboptimal regions.

The following two restriction mechanisms are considered:

1. The most commonly used strategy in DE, where difference vector $\boldsymbol{v}$ is scaled by multiplying it by the parameter $F = 0.5$.

2. The proposed strategy, where difference vector $\boldsymbol{v}$ is restricted as:

$$v_j = \begin{cases} -\delta_j & \text{if } v_j < -\delta_j \\ \delta_j & \text{if } v_j > \delta_j \\ v_j & \text{otherwise} \end{cases} \tag{8.2.1}$$

where

$$\delta_j = \frac{u_j - l_j}{2} \quad j \in \{1, \ldots, n\}, \tag{8.2.2}$$

u_j and l_j are the upper and lower bounds of the j-th variable, respectively.

It should be noted that the proposed restriction mechanism is the same as the one used in SMPSO [135] to control the speed of particles. Due to some similarities in the working principles of PSO and DE, this approach is used in the context of DE optimization. Differential evolution with the mutation restriction mechanism shown in (8.2.2) will be referred as DEMR. This mechanism is expected to improve the convergence properties of DE-based algorithms on multimodal problems.

8.2.2 Performance Assessment

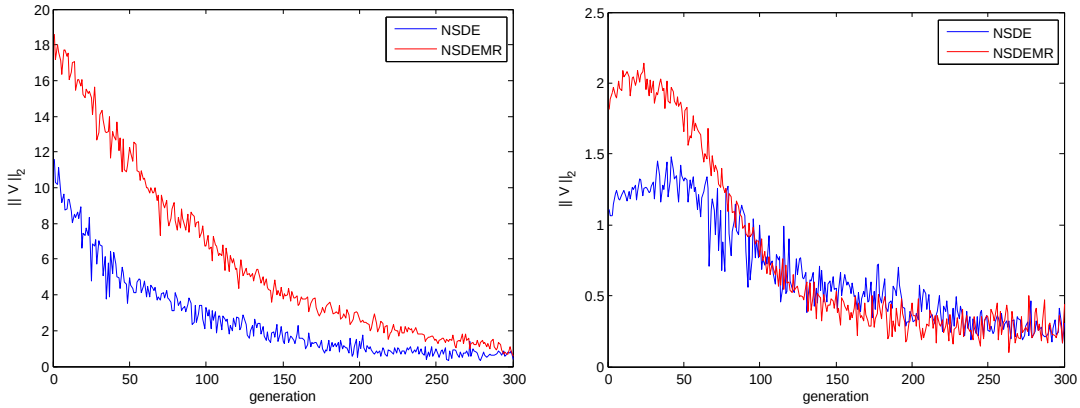
In the following, two variants of NSDE are under consideration: (i) uses the conventional DE mutation restriction strategy, where mutation vector $\boldsymbol{v}$ is multiplied by parameter F, referred as NSDE; (ii) employs the proposed variable-wise mutation restriction (8.2.2), referred as NSDEMR.

Experimental Setup

For each variant, 30 independent runs are performed on the ZDT4 and two-objective DTLZ1 test problems with 30 decision variables. Both problems are highly multimodal. For both approaches, the population size is set to 300, and the total number of generations is 300. The remaining parameters are: the crossover probability $CR = 0.15$, and the distribution index for polynomial mutation $\eta_m = 20$.

Experimental Results

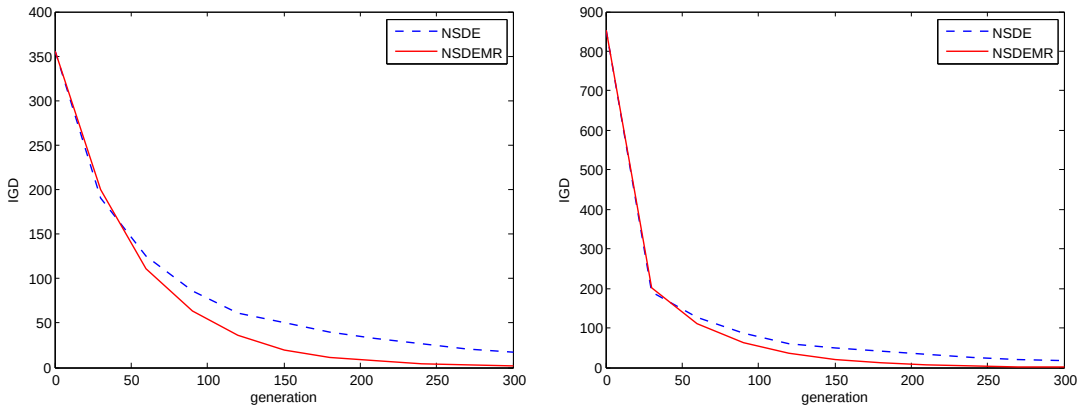
To illustrate the effect of the proposed mutation restriction mechanism, Figure 8.1 shows the length of mutation vector for randomly chosen parent in each generation. The two plots presented in this figure refer to the medians over 30 runs. From Figure 8.1(a), one can observe that the norm of mutation vector is larger in NSDEMR. The norms in both algorithms become equal only at the end of the evolutionary process. From Figure 8.1(b), it can be seen that in early generations the norm of $\boldsymbol{v}$ in NSDEMR is significantly larger. The norms in both algorithms become to have a comparable length after 100 generations.



(a) ZDT4 (b) DTLZ1

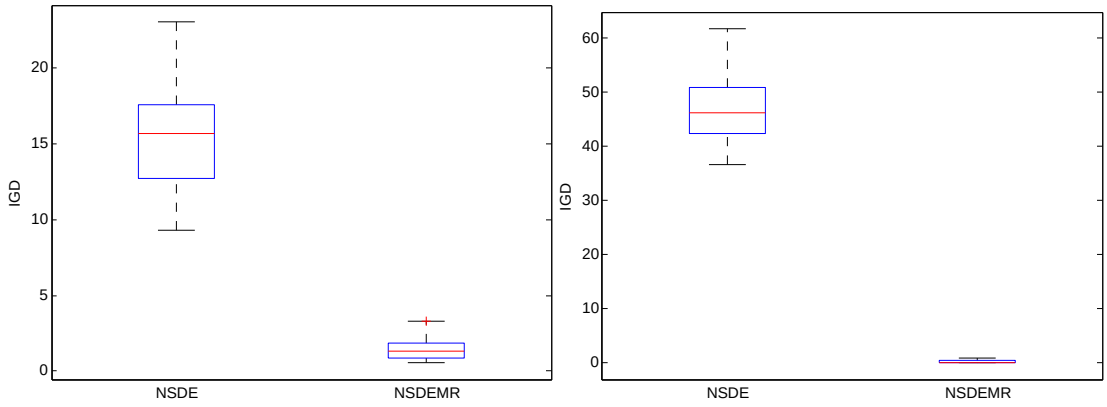
Figure 8.1: Length of mutation vector during the generations.

From the considered figures, it becomes clear that the proposed scheme to restrict the mutation vector results in higher lengths of mutation vectors over the generations. On the other hand, the components of mutation vector cannot be larger than the limits imposed by the restriction mechanism. This prevents high mutations of genes to take place during one generation. Next, the impact of increasing the length of mutation vector on the performance of the algorithm needs to be investigated. This impact can be readily observed



(a) ZDT4 (b) DTLZ1

Figure 8.2: Evolution of IGD during the generations.



(a) ZDT4 (b) DTLZ1

Figure 8.3: Comparison of NSDE and NSDEMR in terms of IGD.

in Figure 8.2. The plots presented in this figure show the median values of IGD over 30 runs during the generations. From both plots, it can be seen that NSDEMR demonstrates the faster convergence rate and the better final values of IGD on both ZDT4 and DTLZ1 test problems. The improved performance is directly related to larger lengths of mutation vectors in NSDEMR, preventing the algorithm from getting stuck in suboptimal regions of the search space.

Additionally, Figure 8.3 presents the distributions of IGD produced by both algorithms on the ZDT4 and DTLZ1 test problems. The two plots depicted in this figure clearly highlight the superiority of NSDEMR on both problems.

8.2.3 Intermediate Summary

This section discusses a restriction mechanism to control the mutation strength in the DE-based reproduction operator. The proposed strategy is implemented within the modified nondominated sorting differential evolution. The effect of using the proposed mutation restriction is studied on the multimodal ZDT4 and DTLZ1 test problems. The proposed mechanism to control mutation strength by means of imposing the lower and upper parameter values on the genes of mutation vector results in higher lengths of mutation vectors

during the evolution. Experimental results reveal that larger lengths of mutation vectors help to prevent the population from getting stuck in locally optimal regions. Although this section does not report on the performance of the algorithm without any restriction mechanism (i.e. the mutation vector is not restricted at all), preliminary results showed that this strategy performs poorly on the considered problems. In summary, the performed experiments demonstrate a significant improvement in terms of convergence and the superiority of the proposed restriction mechanism on multimodal problems. In the subsequent sections, the variation operator based on differential evolution with mutation restriction is used to develop efficient evolutionary many-objective optimization (EMyO) algorithms.

8.3 IGD-Based Selection for Evolutionary Many-Objective Optimization

8.3.1 EMyO-IGD

This section presents an evolutionary many-objective optimization algorithm with IGD-based selection (EMyO-IGD). The outline of EMyO-IGD is given by Algorithm 14.

EMyO-IGD adopts the same algorithmic framework and variation operator as used in NSDEMR. However, the significant difference between the two algorithms lies in `environmentalSelection` procedure. The selection process relies on two sorting crite-

Algorithm 14 EMyO-IGD

1: `initialize`: P and P^*;
2: **repeat**
3: $P' \leftarrow P$, $P'' \leftarrow \{\}$;
4: $P'' \leftarrow$ `variation`(P');
5: $P \leftarrow$ `environmentalSelection`$(P' \cup P'')$;
6: **until** the stopping criterion is met
7: output: P;

ria: (i) nondominated sorting, and (ii) distance metric. As it is discussed in this thesis, to successfully handle many-objective optimization problems, a proper secondary sorting criterion must be used. Actually, there is a trend in EMO community to use quality indicators in the selection process to guide the search in a high dimensional objective space. Following this current trend, a new selection mechanism based on the IGD indicator is developed. However, a reference set is required to calculate IGD. Thus, a set of evenly distributed points on an $(m-1)$-dimensional unit hyperplane must be provided before the search. In Algorithm 14, this hyperplane is denoted as P^*. In each generation, the hyperplane is used to approximate the Pareto front for calculating IGD.

Thus, in detail `environmentalSelection` procedure works as follows. First, the combined population $P' \cup P''$ is sorted according to different non-domination levels ($\mathcal{F}_1$, $\mathcal{F}_2$ and so on). Then, each non-domination level is selected one at a time, starting from $\mathcal{F}_1$, until no further level can be included without increasing the population size. Say the last accepted non-domination level is the l-th level. In general, the last accepted level cannot be completely accommodated. In such a case, only those solutions are kept, which maximize the second sorting criterion.

As the second sorting criterion, a distance metric based on the inverted generational distance is used. For choosing the remaining $k = |P| - \sum_{i=1}^{l-1} |\mathcal{F}_i|$ population members from the last accepted front $\mathcal{F}_l$, the following steps are performed:

1. The objectives of the solutions in $\mathcal{F}_l$ are normalized as:

$$\overline{f}_i = \frac{f_i - f_i^{\min}}{f_i^{\max} - f_i^{\min}}, \quad \forall i \in \{1, \ldots, m\} \tag{8.3.1}$$

   where $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum values of the i-th objective in $\mathcal{F}_l$, respectively. $\overline{f}_i \in [0, 1], \forall i \in \{1, \ldots, m\}$ is the normalized objective value.

2. If there are normalized solutions in $\mathcal{F}_l$ that are below the unit hyperplane, then the hyperplane is moved until no solutions are below it. The resulting hyperplane is used as a reference set to calculate IGD for the normalized set of objective vectors in $\mathcal{F}_l$.

3. For each $a \in \mathcal{F}_l$, a distance metric $I[a]$ based on IGD is calculated as:

$$I[a] = \text{IGD}(P^*, A) \tag{8.3.2}$$

where $A = \mathcal{F}_l \backslash a$, and P^* is the reference set.

In (8.3.2), the quality of solution a is evaluated by calculating the IGD metric of the set $\mathcal{F}_l$ without the solution a. The metric is the resulting IGD for the set $\mathcal{F}_l$ removing a. The higher value of $I[a]$, the better quality of solution a is, as removing solution a leads to a higher value of IGD that, in turn, corresponds to a poorer performance of the resulting set. Thus, removing the solutions with high values of $I[a]$ is undesirable, so they should be kept in the population.

Therefore, after computing the distance metric I for each solution in $\mathcal{F}_l$, the set $\mathcal{F}_l$ is sorted in descending order with respect to I. Finally, k first elements of the sorting set are included into P.

8.3.2 Performance Assessment

In the following, EMyO-IGD is compared with IBEA [198], GDE3 [116], SMPSO [135], and MOEA/D [123]. The main concern of the study is the ability of the algorithms to guide the search in a high dimensional objective space. Although SMPSO and GDE3 are expected to perform poorly when the number of objectives is increased, they serve as an important reference as they are state-of-the-art EMO algorithms. Moreover, it is also interesting to compare the convergence property of stochastic direct search algorithms on multimodal problems, thereby validating the proposed mutation restriction mechanism. The DTLZ test suite is used as the basis for comparison, since it can be scaled to an arbitrary number of objectives. Thus, the DTLZ1-4 test problems are adopted, each with 30 decision variables, ranging from 2 to 20 objectives.

EMyO-IGD	MOEA/D	IBEA	SMPSO	GDE3
$CR = 0.15$	$CR = 0.15$	$p_c = 0.9$	$\eta_m = 20$	$CR = 0.15$
$\eta_m = 20$	$F = 0.5$	$\eta_c = 20$	$p_m = 1/n$	$F = 0.5$
$p_m = 1/n$	$\eta_m = 20$	$\eta_m = 20$		
$\lvert P^* \rvert = 500$	$p_m = 1/n$	$p_m = 1/n$		

Table 8.1: Parameter settings for the algorithms.

Experimental Setup

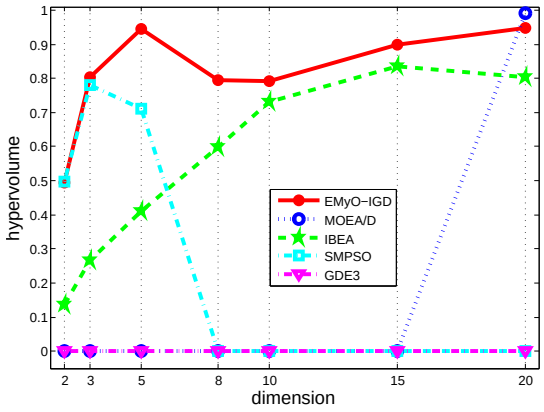
EMyO-IGD and MOEA/D are implemented in the MATLAB® programming language, IBEA is used within the PISA framework [16], whereas GDE3 and SMPSO are provided by the jMetal framework [62]. For each algorithm, 30 independent runs are performed on each problem with a population size of $\mu = 300$, running for 500 generations. The remaining parameter settings for the algorithms are presented in Table 8.1 (the parameters not specified use the default values as stated in the corresponding papers).

Experimental Results

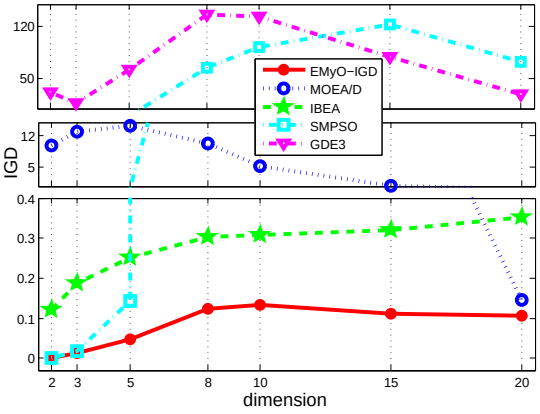
Figure 8.4 presents the graphical representation of the median values of the hypervolume indicator and IGD obtained by all the algorithms on the benchmark functions with varying dimensions.

Problems DTLZ1,3 are known to be hard to solve as they contain a very large number of local Pareto optimal fronts [50], so they are considered in more detail. For DTLZ1, EMyO-IGD achieves the best hypervolume for $m > 2$, except for $m = 20$, where the best median hypervolume is obtained by MOEA/D. However, in terms of the IGD indicator EMyO-IGD provides better results for $m > 2$ than the other four algorithms on DTLZ1. For DTLZ3, EMyO-IGD achieves the best hypervolume for $2 < m < 15$. Similar performance is observed in terms of the IGD indicator.

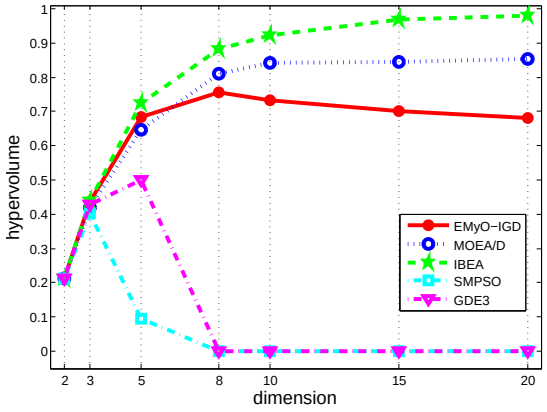
Concerning the other problems, EMyO-IGD is always better, concerning both quality indicators, than SMPSO and GDE3 for $m > 3$, except for DTLZ4 with five objectives,



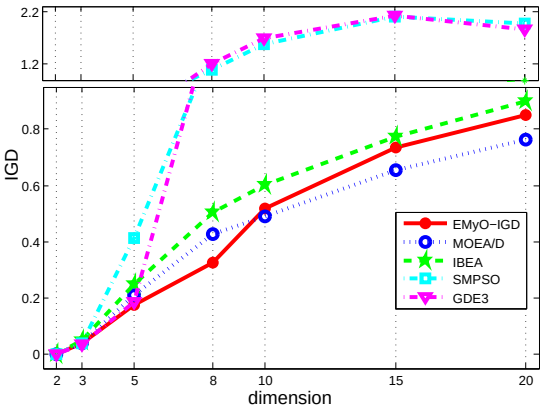
(a) DTLZ1 (the higher the better)



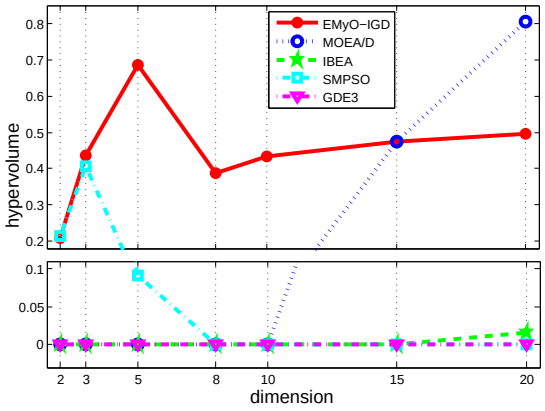
(b) DTLZ1 (the lower the better)



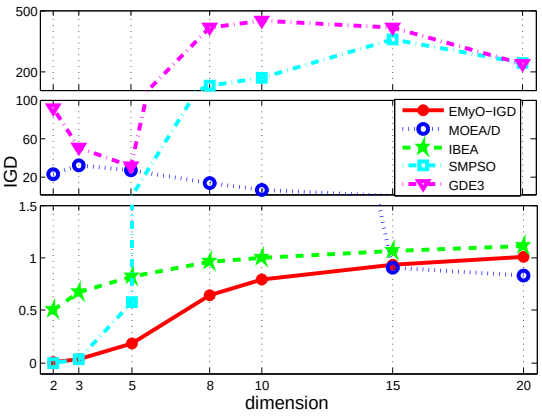
(c) DTLZ2 (the higher the better)



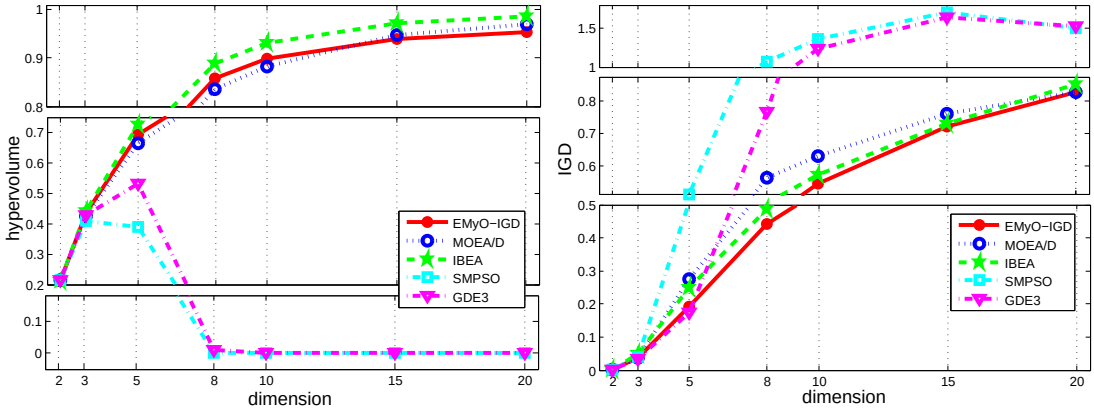
(d) DTLZ2 (the lower the better)



(e) DTLZ3 (the higher the better)



(f) DTLZ3 (the lower the better)



(g) DTLZ4 (the higher the better)

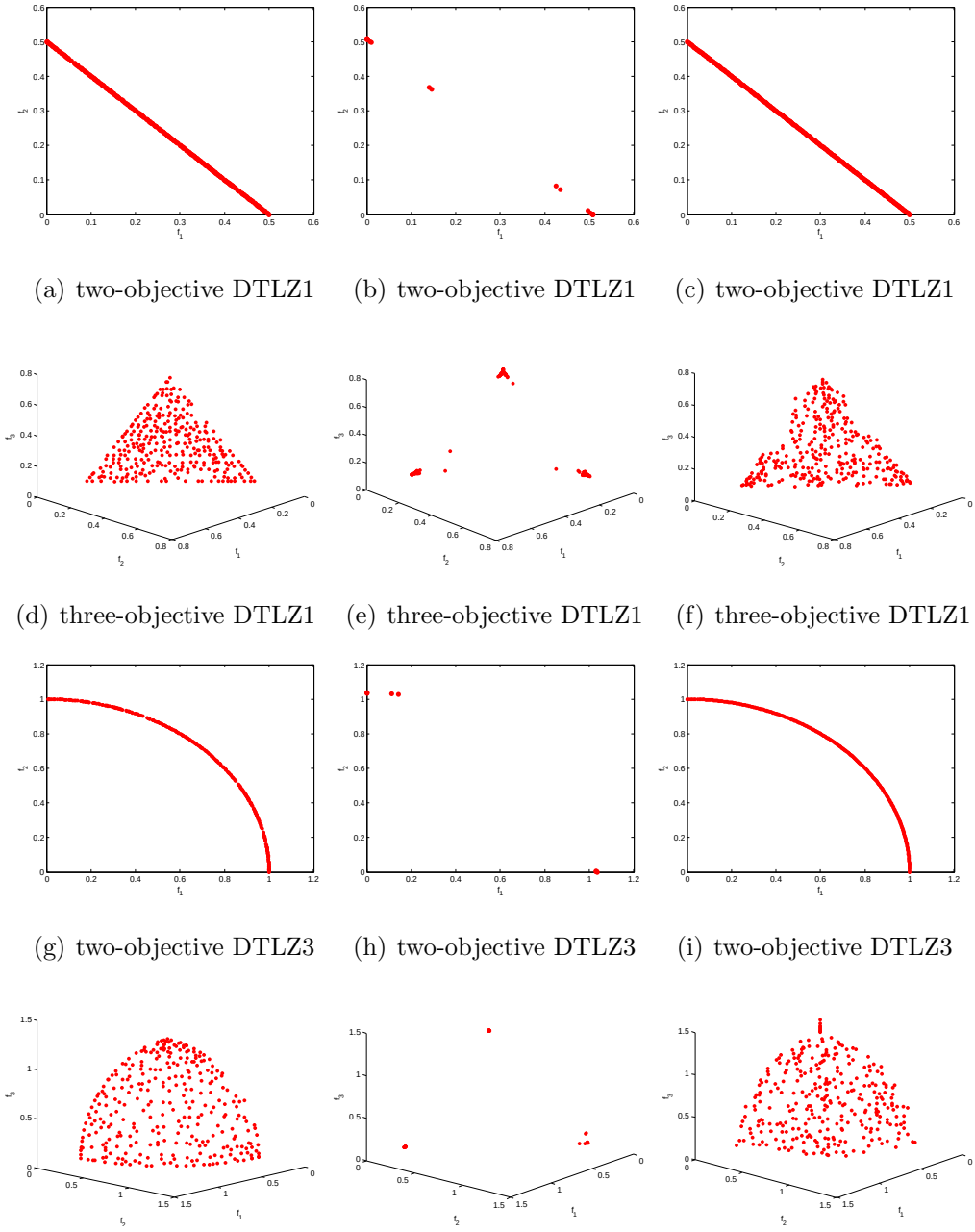
(h) DTLZ4 (the lower the better)

Figure 8.4: Performance comparison of EMyO-IGD, MOEA/D, IBEA, SMPSO, and GDE3 on the DTLZ1-4 test problems. The plots present the median values of the hypervolume (left hand side) and IGD (right hand side) over 30 runs.

where GDE3 yields the best results among all the algorithms in terms of IGD. Moreover, EMyO-IGD is always better than IBEA with respect to IGD, except for DTLZ4 with $m = 2$. Furthermore, IBEA provides the best hypervolume among all algorithms for $m > 2$ on DTLZ2 and DTLZ4. However, these problems do not present as much difficulties in terms of convergence to the Pareto front as the other considered problems, and IBEA is the only considered indicator based algorithm (the epsilon indicator was used) while the other algorithms use a diversity mechanism which attempts to uniformly distribute points along the Pareto front.

From the above discussion, it can be seen that EMyO-IGD performs the best on the multimodal problems (DTLZ1 and DTLZ3). Better results are only obtained by SMPSO for $m = 2$, which also outperforms all the algorithms in terms of IGD on all the problems with two objectives. MOEA/D performs better with regard to the hypervolume than EMyO-IGD on DTLZ1 with $m = 20$, and in terms of IGD on DTLZ3 with $m = 20$.

EMyO-IGD is the only algorithm which is able to converge to the Pareto front for all the problems, providing an adequate distribution of solutions along the Pareto front. Although IBEA converges to the Pareto front for all the problems too, it is observed



(a) two-objective DTLZ1 (b) two-objective DTLZ1 (c) two-objective DTLZ1

(d) three-objective DTLZ1 (e) three-objective DTLZ1 (f) three-objective DTLZ1

(g) two-objective DTLZ3 (h) two-objective DTLZ3 (i) two-objective DTLZ3

(j) three-objective DTLZ3 (k) three-objective DTLZ3 (l) three-objective DTLZ3

Figure 8.5: Performance of EMyO-IGD (left), IBEA (middle), SMPSO (right) on the two and three-objective DTLZ1 and DTLZ3 test problems.

that it is able to converge only to the corner points of the Pareto front for DTLZ1 and DTLZ3. Figure 8.5 displays approximation sets with the best values of IGD obtained on the multimodal DTLZ1 and DTLZ3 test problems. The figure does not present the performance of the algorithms that did not converge to the Pareto front for these problems.

A good convergence of EMyO-IGD on multimodal problems is due to the proposed restriction mechanism used in the DE reproduction operator. EMyO-IGD clearly outperforms all of the considered DE-based algorithms on multimodal problems, even in low-dimensional objective spaces where the selection mechanism of all the algorithms is expected to work well.

However, it is interesting to note that with the increasing number of objectives the performance of MOEA/D becomes notably better in terms of both quality indicators, especially on multimodal problems. Such behavior is related to the properties of the DTLZ test suite. The decision space of multiobjective problems consists of k position- and l distance-related parameters ($n = k + l$) [87]. For the DTLZ test suite, the number of position-related parameters is $k = m - 1$, and the number of distance-related parameters which account for the convergence is $l = n - m + 1$. Thus, when the number of objectives increases and the total number of decision variables remains constant, the number of distance-related parameters decreases. Such situation results in that a problem in question has a lower number of local Pareto fronts, consequently it is much easier to converge to the Pareto front.

Such scenario can be easily observed in the case of DTLZ1. In Figures 8.4(a) and 8.4(b), one can see that if $m \leq 15$, the number of distance-related parameters is $l = n-m+1 \geq 16$, MOEA/D is unable to converge that results in zero hypervolume and high values of IGD. However, if $m = 20$ then $l = 11$, and MOEA/D succeeds to converge to the Pareto front. Nevertheless, when the number of distance-related parameters is high, especially in the case of two and three dimensions, EMyO-IGD is the only among all of the tested DE-based algorithms that converges to the Pareto front. At the same time, EMyO-IGD provides highly competitive results compared to SMPSO on multimodal benchmarks with two and three objectives (Figure 8.5).

	EMyO-IGD	MOEA/D	IBEA	SMPSO	GDE3
2-objectives					
DTLZ1	II,III,V		II,V	I,II,III,V	
DTLZ2		I,III	I	I,II,III	I,II,III,IV
DTLZ3	II,III,V			I,II,III,V	
DTLZ4		I,III	I	I,II,III	I,II,III,IV
3-objectives					
DTLZ1	II,III,IV,V		II,V	II,III,V	
DTLZ2	II,IV,V	IV	I,II,IV,V		II,IV
DTLZ3	II,III,IV,V			II,III,V	
DTLZ4	II,IV,V	IV	I,II,IV,V		II,IV
5-objectives					
DTLZ1	II,III,IV,V		II,V	II,III,V	
DTLZ2	II,IV,V	IV,V	I,II,IV,V		IV
DTLZ3	II,III,IV,V			II,III,V	
DTLZ4	II,IV,V	IV,V	I,II,IV,V		IV
8-objectives					
DTLZ1	II,III,IV,V		II,IV,V		
DTLZ2	IV,V	I,IV,V	I,II,IV,V		IV
DTLZ3	II,III,IV,V				
DTLZ4	II,IV,V	IV,V	I,II,IV,V		IV
10-objectives					
DTLZ1	II,III,IV,V		II,IV,V		
DTLZ2	IV,V	I,IV,V	I,II,IV,V		
DTLZ3	II,III,IV,V				
DTLZ4	II,IV,V	IV,V	I,II,IV,V		IV
15-objectives					
DTLZ1	II,III,IV,V	IV,V	II,IV,V		
DTLZ2	IV,V	I,IV,V	I,II,IV,V		
DTLZ3	III,IV,V	III,IV,V		III	III
DTLZ4	IV,V	I,IV,V	I,II,IV,V		
20-objectives					
DTLZ1	III,IV,V	I,III,IV,V	IV,V		IV
DTLZ2	IV,V	I,IV,V	I,II,IV,V		
DTLZ3	III,IV,V	I,III,IV,V	IV,V		
DTLZ4	IV,V	I,IV,V	I,II,IV,V	V	

Table 8.2: Statistical comparison in terms of the hypervolume.

	EMyO-IGD	MOEA/D	IBEA	SMPSO	GDE3
2-objectives					
DTLZ1	II,III,V	V	II,V	I,II,III,V	
DTLZ2	III	I,III		I,II,III,V	I,II,III
DTLZ3	II,III,V	V	II,V	I,II,III,V	
DTLZ4		I,III	I	I,II,III,V	I,II,III
3-objectives					
DTLZ1	II,III,IV,V	V	II,V	II,III,V	
DTLZ2	III,IV	III,IV		III	I,II,III,IV
DTLZ3	II,III,V	V	II,V	II,III,V	
DTLZ4	II,III,IV	III,IV		III	I,II,III,IV
5-objectives					
DTLZ1	II,III,IV,V	V	II,V	II,III,V	
DTLZ2	II,III,IV,V	III,IV	IV		II,III,IV
DTLZ3	II,III,IV,V	V	II,V	II,V	
DTLZ4	II,III,IV	IV	II,IV		I,II,III,IV
8-objectives					
DTLZ1	II,III,IV,V	IV,V	II,IV,V	V	
DTLZ2	II,III,IV,V	III,IV,V	IV,V	V	
DTLZ3	II,III,IV,V	IV,V	II,IV,V	V	
DTLZ4	II,III,IV,V	IV,V	II,IV,V		IV
10-objectives					
DTLZ1	II,III,IV,V	IV,V	II,IV,V	V	
DTLZ2	III,IV,V	I,III,IV,V	IV,V	V	
DTLZ3	II,III,IV,V	IV,V	II,IV,V	V	
DTLZ4	II,III,IV,V	IV,V	II,IV,V		IV
15-objectives					
DTLZ1	II,III,IV,V	IV,V	II,IV,V		IV
DTLZ2	III,IV,V	I,III,IV,V	IV,V		
DTLZ3	III,IV,V	III,IV,V	IV,V	V	
DTLZ4	II,III,IV,V	IV,V	II,IV,V		IV
20-objectives					
DTLZ1	II,III,IV,V	III,IV,V	IV,V		IV
DTLZ2	III,IV,V	I,III,IV,V	IV,V		IV
DTLZ3	III,IV,V	I,III,IV,V	IV,V		
DTLZ4	III,IV,V	III,IV,V	IV,V		

Table 8.3: Statistical comparison in terms of IGD.

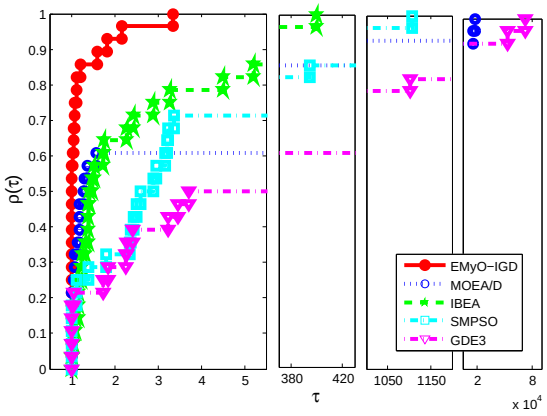


Figure 8.6: Performance profiles on the median values of IGD.

Furthermore, the pairwise statistical comparisons of the algorithms based on the hypervolume and IGD are provided in Table 8.2 and Table 8.3, respectively. Statistical testing is performed using the nonparametric Wilcoxon rank-sum test performed at the significance level of $\alpha = 0.05$. In both tables, the symbols I, II, II, IV, and V indicate whether the respective algorithm performs significantly better than EMyO-IGD, MOEA/D, IBEA, SMPSO, and GDE3, respectively.

Figure 8.6 presents the performance profiles on the median values of IGD. This figure allows to get insights about the overall performance of the algorithms on all of the tested problems. For each dimension, each considered DTLZ test function is viewed as a distinct test problem. So testing DTLZ1-4 test functions in 7 different dimensions there is a total of 28 problems. Thus, this figure shows that the most accurate algorithm is EMyO-IGD, which yields the best median values of the IGD indicator in 50% of the tested problems, the second best is MOEA/D (gives the best median values of IGD in 21% of the problems). In terms of robustness, the best algorithm is EMyO-IGD, the second best is IBEA.

8.3.3 Intermediate Summary

This section presents an evolutionary many-objective optimization algorithm, which incorporates the IGD indicator in the selection process. The algorithm is tested on problems with up to 20 objectives. The obtained results reveal that the proposed selection

scheme is able to effectively guide the search in a high-dimensional objective space. The algorithm produces highly competitive results compared with some state-of-the-art many-objective optimization algorithms. EMyO-IGD shows ability to produce well-covered and well-distributed approximations to the set of Pareto optimal solutions for all the considered problems. Therefore, it can be used as a viable alternative to solve optimization problems with a large number of objectives.

8.4 Clustering-Based Selection for Evolutionary Many-Objective Optimization

8.4.1 EMyO-C

The previous section introduces an approach based on the IGD indicator to handle many-objective problems, where a reference set is used to approximate the Pareto front for calculating distance metrics for each solution in the last non-domination level. The experimental results show that this strategy allows to effectively guide the search in a high-dimensional objective space. The idea of using a set of evenly distributed reference points has been adopted in some other studies [3, 44, 104]. All these proposed approaches significantly improve the scalability of EMO algorithms and appear to be suitable for dealing with many-objective problems. However, the major drawback of these approaches is that all of them require the user to specify a set of evenly distributed points on the unit hyperplane. In turn, it is not always an easy task especially for higher dimensions. Furthermore, changing the population size often requires to change the number of points on the hyperplane. These issues make EMO algorithms less flexible and less attractive to practitioners.

This section presents an evolutionary many-objective optimization algorithm with clustering-based selection (EMyO-C), which is developed to overcome the aforementioned difficulties. The outline of EMyO-C is given by Algorithm 14. The only difference between EMyO-IGD and EMyO-C is in `environmentalSelection` procedure. EMyO-C performs the selection process relying on the nondominated sorting to sort the combined population

according to different non-domination levels and a clustering-based procedure to select individuals from the last non-domination level.

In order to select the remaining $k = |P| - \sum_{i=1}^{l-1} |\mathcal{F}_i|$ population members from the last accepted non-domination front $\mathcal{F}_l$, the following steps are performed:

1. The distances from each individual in $\mathcal{F}_l$ to the reference point are calculated in the objective space.

2. Each objective of an individual in $\mathcal{F}_l$ is translated by substracting objective f_i by $f_i^{\min}$, where $f_i^{\min}$ is the minimum value of the i-th objective in $\mathcal{F}_l$. As a result, the ideal point of $\mathcal{F}_l$ is a zero vector, and objectives in $\mathcal{F}_l$ are nonnegative.

3. Each individual in $\mathcal{F}_l$ is projected on the unit hyperplane.

4. Clustering technique is performed using projected individuals in order to form k clusters.

5. From each cluster, a representative is selected, where a cluster representative is an individual having the smallest distance to the reference point.

Using this procedure, k individuals are selected and added to the population of the next generation. It should be be noted that this procedure is easy to implement, and it is completely self-adaptive.

The reference point $\boldsymbol{z}$ used in the above described procedure is initialized by setting the minimum values of each objective in the initial population ($z_j = \min_{1 \leq i \leq \mu} f_j(\boldsymbol{x}^i)$). At the end of each generation, the components of the reference point are updated if there are smaller objective values in the offspring population. The clustering algorithm is as follows:

Step 1 Initially, each point belongs to a separate cluster ($C_i = \{i\}$), so that $C = \{C_1, C_2, \ldots, C_{|\mathcal{F}_l|}\}$.

Step 2 If $|C| \leq k$, stop. Otherwise, go to Step 3.

Step 3 For each pair of clusters, calculate the distance between two cluster d_{12} as:

$$d_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i,j),$$

where $d(i,j)$ is the Euclidean distance between two points i and j. Find the pair (i_1, i_2) which corresponds to the minimum cluster-distance.

Step 4 Merge the clusters C_{i_1} and C_{i_2}. This reduces the size of C by one.
Go to Step 2.

If implemented in a proper way, the complexity of the above clustering algorithm is $O(m|\mathcal{F}_l|^2)$ [42]. So the complexity is mainly governed by the number of points in $\mathcal{F}_l$, while it is polynomial in the number of objectives. This feature is especially attractive to solve problems with a large number of objectives.

8.4.2 Performance Assessment

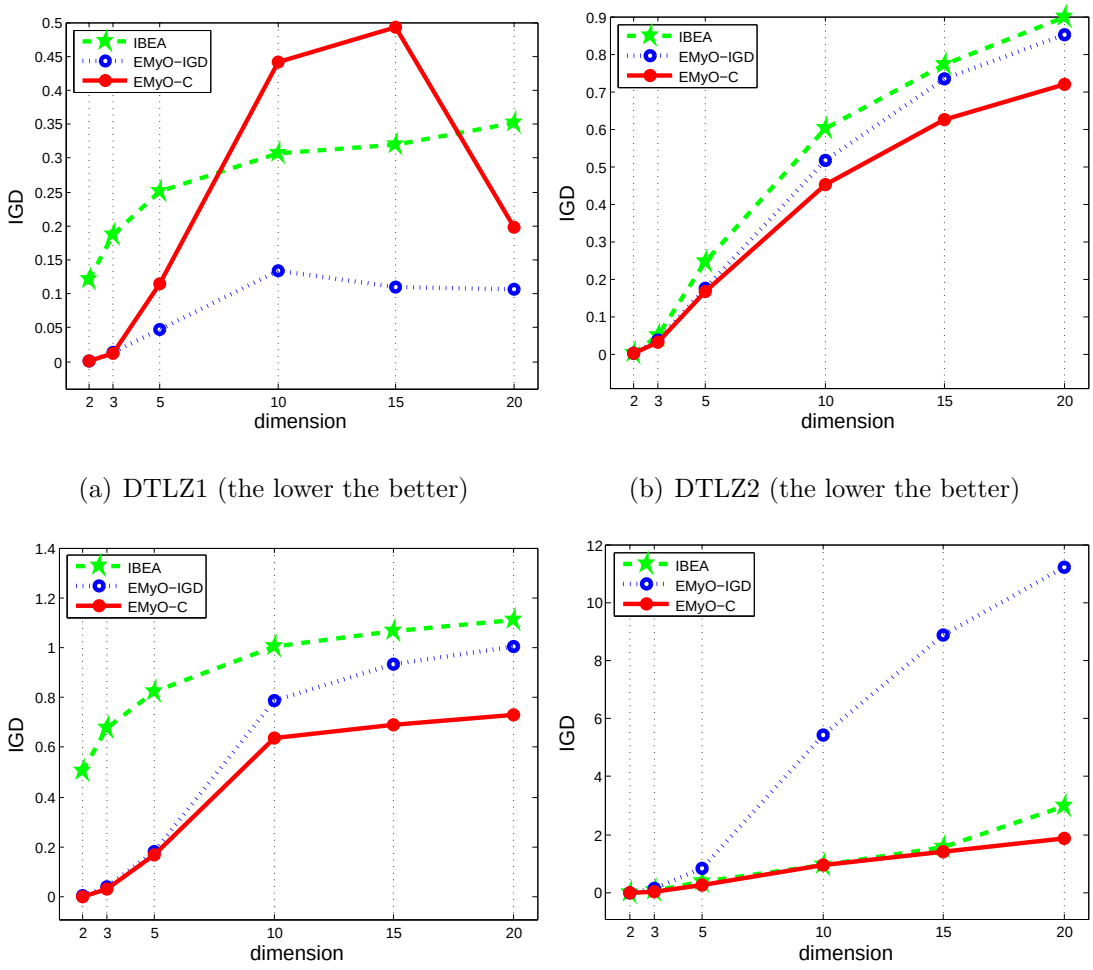
In the following, EMyO-C is compared with EMyO-IGD and IBEA. The main concern of this study is to investigate the ability of the clustering-based selection scheme to guide the search in a high-dimensional objective space. The performance of the algorithms is studied on the DTLZ1-3,7 test problems with 30 decision variables. Since these problems have different Pareto front geometries, including linear, concave, and disconnected geometries, strengths and weakness of different selection schemes used by the algorithms can be identified.

Experimental Setup

For each algorithm, 30 independent runs are performed on each problem with a population size of 300, running for 500 generations. Further parameters for IBEA and EMyO-IGD are the same as the ones used in the previous section. EMyO-C uses the same parameter settings as EMyO-IGD.

Experimental Results

Figure 8.7 shows the median values of the IGD indicator obtained by three algorithms on the DTLZ1-3,7 test problems with varying dimensions. From Figure 8.7(a), it can be seen that EMyO-C has the worst performance on DTLZ1 with 10 and 15 objectives. However, in other dimensions the performance of EMyO-C is very close to that produced by EMyO-IGD, which generally performs the best on this problem. It is not surprising,



(a) DTLZ1 (the lower the better)

(b) DTLZ2 (the lower the better)

(c) DTLZ3 (the lower the better)

(d) DTLZ7 (the lower the better)

Figure 8.7: Performance comparison of EMyO-C, EMyO-IGD, and IBEA on the DTLZ1-3,7 test problems.

since EMyO-IGD uses unit hyperplane in its selection process, which coincides with the Pareto front geometry of this problem. EMyO-C produces similar performance as EMyO-IGD on DTLZ2 and DTLZ3 with up to 5 objectives (Figures 8.7(b) and 8.7(c)). Both algorithms outperform IBEA on these two problems with respect to the IGD indicator. However, EMyO-C performs the best on DTLZ2 and DTLZ3 when the dimensionality of the objective space is larger than 5 (Figures 8.7(b) and 8.7(c)). In Figure 8.7(d), one can see that EMyO-C outperforms all the other algorithms in higher dimensions on DTLZ7. It is interesting to note that, when the number of objectives grows, the performance of EMyO-IGD on this problem becomes increasingly poorer. This is due to the fact that the unit hyperplane used in EMyO-IGD cannot approximate the Pareto front geometry of DTLZ7 properly. In this case, the selection mechanism used in EMyO-C becomes particularly useful.

Furthermore, in order to show the scalability of the proposed approach, EMyO-C is run on DTLZ2 with 50 and 100 objectives, having 60 and 110 decision variables, respectively. Parameter settings for EMyO-C are the same as in the previous experiment. To measure the convergence of solutions in the final population P to the Pareto front, the proximity indicator [142] is used, which is defined as:

$$I_p = median\left\{\left(\sum_{i=1}^{m}(f_i(\boldsymbol{x}))^2\right)^{\frac{1}{2}} - 1\right\}. \tag{8.4.1}$$

Smaller values of I_p indicate that the population P is closer to the Pareto optimal front.

Figure 8.8 shows the distributions of the proximity indicator for the final populations returned by EMyO-C on the DTLZ2 test problem with 50 and 100 objectives over 30 runs. The small values of I_p presented in the plot clearly indicate that EMyO-C converges to the Pareto front in the presence of such a large number of objectives. Moreover, the relatively small variability of the values of I_p in both 50 and 100-dimensional objective spaces emphasizes the robustness of the approach.

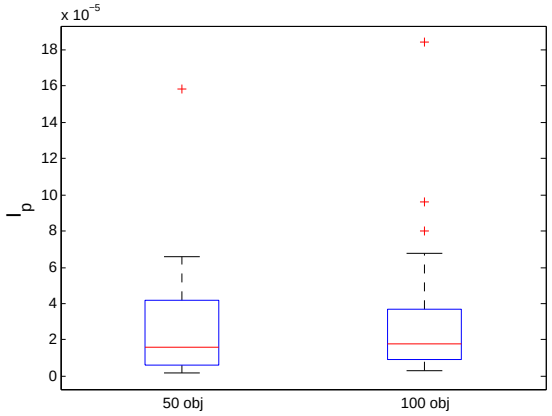


Figure 8.8: Distribution of the proximity indicator on DTLZ2 for EMyO-C.

8.5 Summary

This chapter addresses evolutionary many-objective optimization adopting a DE operator to generate offspring individuals and the widely used two-stage fitness assignment mechanism for multiobjective optimization as the base. The main contributions are: (i) an improved DE-based reproduction operator for multimodal problems, and (ii) new selection schemes to guide the search in high dimensional objective spaces.

The proposed mutation restriction mechanism introduces the lower and upper parameter values for the components of mutation vector instead of multiplying it by a scaling parameter. The performed experiments show that the restriction mechanism results in the increase of the length of mutation vector during the search. The results obtained on multimodal problems reveal that this allows the population to escape from suboptimal regions of the search space. A significant performance improvement is observed compared to a traditional restriction technique.

In order to deal with many-objective problems using the nondominated sorting in the selection process, a proper second sorting criterion must be utilized. Thus, two different second sorting criteria are developed. The suggested IGD-based second sorting criterion requires a set of evenly distributed points on the unit hyperplane to be provided in advance. The algorithm using this selection scheme is tested on problems with up to 20 objectives.

EMyO-IGD demonstrates the ability to effectively guide the search in a high-dimensional objective space producing highly competitive results, when compared with other state-of-the-art many-objective algorithms. However, drawbacks due to the need to specify a set of uniformly distributed points and relatively poor performance on problems with a complex Pareto front geometry are identified. To overcome these drawbacks, the clustering-based selection is suggested. The performed experiments show that this selection mechanism is able to guide the search in a high-dimensional objective space. Although some limitations of the IGD-based selection are overcome, it is observed that EMyO-C produces sightly poorer performance on some test instances. However, the results obtained on problems with a large number of objectives clearly indicate a promising direction for future research.

Part III

Applications

Chapter 9

Dengue Disease Transmission

During the last decades, the global prevalence of dengue progressed dramatically. It is a disease which is now endemic in more than one hundred countries of Africa, America, Asia, and the Western Pacific. This chapter presents a mathematical model for the dengue disease transmission and uses a multiobjective approach to find an optimal strategy to control the disease.

9.1 Disease Background

Dengue is a vector-borne disease transmitted from an infected human to a female *Aedes* mosquito by a bite. Then, the mosquito, that needs regular meals of blood to feed their eggs, bites a potentially healthy human and transmits the disease, turning it into a cycle.

There are two forms of dengue: dengue fever (DF) and dengue hemorrhagic fever (DHF). The first one is characterized by a sudden high fever without respiratory symptoms, accompanied by intense headaches, pain in joints and muscles and lasts between three to seven days. Humans may only transmit the virus during the febrile stage. DHF initially exhibits a similar, if more severe pathology as DF, but deviates from the classic pattern at the end of the febrile stage. Additionally, the hemorrhagic form is characterized by nose bleeding, skin or mouth and gums bruising, nausea, vomiting and fainting due to

Figure 9.1: Mosquito *Aedes aegypti* (adapted from [147]).

low blood pressure and fluid leakage. It usually lasts between two to three days and may have lethal outcome. Nowadays, dengue is the mosquito-borne infection that has become a major international public health concern. According to the World Health Organization (WHO), 50 to 100 million dengue fever infections occur yearly, including 500000 dengue hemorrhagic fever cases and 22000 deaths, mostly among children [186].

There are four distinct, but closely related, viruses that cause dengue. The four serotypes, named DEN-1 to DEN-4, belong to the *Flavivirus* family, but they are antigenically distinct. Recovery from infection by one virus provides lifelong immunity against that virus but provides only partial and transient protection against subsequent infection by the other three viruses. There are strong evidences that a sequential infection increases the risk of developing DHF.

The spread of dengue is attributed to the geographic expansion of the mosquitoes responsible for the disease: *Aedes aegypti* and *Aedes albopictus*. The *Aedes aegypti* mosquito (Figure 9.1) is a tropical and subtropical species widely distributed around the world, mostly between latitudes 35^oN and 35^oS. In urban areas, *Aedes* mosquitoes breed on water collections in artificial containers such as cans, plastic cups, used tires, broken bottles and flower pots. Due to its high interaction with humans and its urban behavior, the *Aedes aegypti* mosquito is considered the major responsible for the dengue transmission.

Dengue is spread only by adult females that require blood consumption for the development of their eggs, whereas male mosquitoes feed on fruit nectar and other sources of sugar. In this process, the female mosquitoes acquire the virus while feeding from the blood of an infected person. After virus incubation from eight to twelve days (*extrinsic* period) and for the rest of its life, an infected mosquito is capable of transmitting the virus to

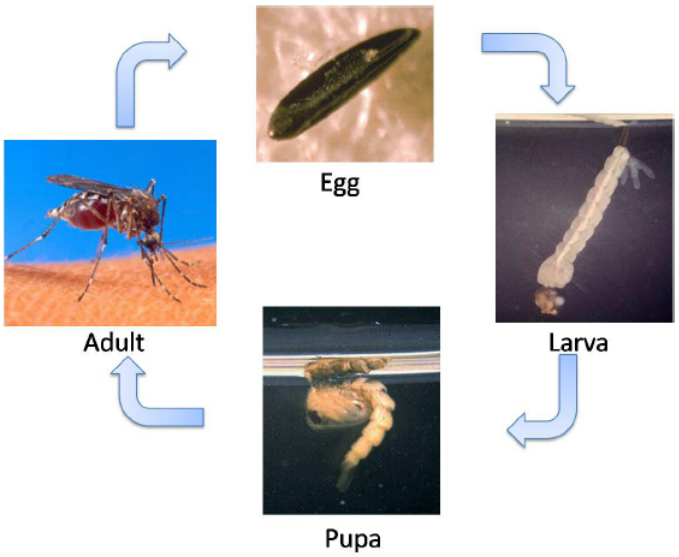


Figure 9.2: Life cycle of *Aedes aegypti* (adapted from [147]).

susceptible humans during probing and blood feeding, and the *intrinsic* period for humans varies from 3 to 15 days.

The life cycle of a mosquito has four distinct stages: egg, larva, pupa and adult, as it is possible to see in Figure 9.2. In the case of *Aedes aegypti*, the first three stages take place in, or near, the water, whereas the air is the medium for the adult stage [139]. Female mosquitoes lay their eggs by releasing them in different places, and therefore increasing the probability of new births.

It is very difficult to control or eliminate *Aedes aegypti* mosquitoes due to their resiliency, fast adaptation to changes in the environment and their ability to rapidly bounce back to initial numbers after disturbances resulting from natural phenomena (e.g., droughts) or human interventions (e.g., control measures).

Primary prevention of dengue resides mainly in mosquito control. There are two primary methods: larval control and adult mosquito control, depending on the intended target. *Larvicide* treatment is done through long-lasting chemical in order to kill larvae and preferably have WHO clearance for use in drinking water [56]. *Adulticides* is the most common measure, its application can have a powerful impact on the abundance of adult

mosquito vector. However, the efficacy is often constrained by the difficulty in achieving sufficiently high coverage of resting surfaces [58].

The most recent approach for fighting the disease is *biological control*. It is a natural process of population regulation through natural enemies. There are techniques that combine some parasites that kill partially the larval population; however, the operational difficulty stems from the lack of expertise in producing this type of parasites as well as cultural objections in introducing external agents into the water aimed for human consumption [25].

Another way of insect control is by changing the reproduction process, in particular, releasing sterile insects. This technique, known as *sterile insect technique*, consists of releasing sterile insects in natural environment, so that as a result of mating produces non-viable eggs are released, which can lead to drastic reduction of the species. This way of control faces two types of challenges: the cost of producing and releasing insects, as well as possible social objection, because an uninformed population may fail to understand how the addition of insects can represent a good solution [66].

Mathematical modeling is critical to understand how epidemiological diseases spread. It can help to explain the nature and dynamics of infection transmissions and can be used to devise effective strategies for fighting them. Therefore, a multiobjective optimization approach is applied to a mathematical model for the dengue transmission in order to find an optimal strategy to reduce economical costs incurred by the dengue disease.

9.2 ODE SEIR+ASEI Model with Insecticide Control

In the following, a mathematical model for the dengue disease transmission proposed in [147] is described. The model consists of eight mutually-exclusive compartments representing the human and vector dynamics. It also includes a control parameter, an adulticide spray, as a measure to fight the disease. Furthermore, the presented model uses real data of a dengue disease outbreak that occurred in the Cape Verde archipelago in 2009.

The notation used in the mathematical model includes four epidemiological states for

humans:

$S_h(t)$: susceptible
$E_h(t)$: exposed
$I_h(t)$: infected
$R_h(t)$: resistant

It is assumed that the total human population (N_h) is constant, so, $N_h = S_h + E_h + I_h + R_h$.

There are also four other state variables related to the female mosquitoes (as discussed earlier, male mosquitoes do not affect the dynamics of the disease). The considered state variables for mosquitoes are:

$A_m(t)$: aquatic phase
$S_m(t)$: susceptible
$E_m(t)$: exposed
$I_m(t)$: infected

Similarly, it is assumed that the total adult mosquito population is constant, which means $N_m = S_m + E_m + I_m$.

The model includes a control variable, which represents the amount of insecticide that is continuously applied during a considered period, as a measure to fight the disease:

$c(t)$: level of insecticide campaigns

The control variable is an adimensional value that is considered in relative terms varying from 0 to 1.

In the following, for the sake of simplicity, the independent variable t is omitted when writing the dependent variables (for instance, S_h is used instead of $S_h(t)$).

The following parameters are necessary to completely describe the model, and include the real data related to the outbreak of dengue disease occurred in the Cape Verde in 2009:

$N_h = 480000$:	total population
$B = 1$	average daily bites (per mosquito per day)
$\beta_{mh} = 0.375$:	transmission probability from I_m (per bite)
$\beta_{hm} = 0.375$:	transmission probability from I_h (per bite)
$\mu_h = 1/(71 \times 365)$:	average human lifespan (in day)
$\eta_h = 1/3$:	mean viremic period (in days)
$\mu_m = 1/11$:	average lifespan of adult mosquitoes (in day)
$\varphi = 6$:	number of eggs at each deposit per capita (per day)
$\mu_A = 1/4$:	natural mortality of larvae (per day)
$\eta_A = 0.08$:	rate of maturing from larvae to adult (per day)
$\eta_m = 1/11$:	extrinsic incubation period (in days)
$\nu_h = 1/4$:	intrinsic incubation period (in days)
$m = 6$:	number of female mosquitoes per human
$k = 3$:	number of larvae per human

Furthermore, in order to obtain a numerically stable problem, all the state variables are normalized as follows:

$$s_h = \frac{S_h}{N_h} \qquad e_h = \frac{E_h}{N_h} \qquad i_h = \frac{I_h}{N_h} \qquad r_h = \frac{R_h}{N_h}$$

$$a_m = \frac{A_h}{kN_h} \quad s_m = \frac{S_m}{mN_h} \quad e_m = \frac{E_m}{mN_h} \quad i_m = \frac{I_m}{mN_h}$$

Thus, the dengue epidemic is modeled by the following nonlinear time-varying state

equations:

$$\begin{cases} \frac{ds_h}{dt} = \mu_h - (B\beta_{mh} m i_m + \mu_h) s_h \\ \frac{de_h}{dt} = B\beta_{mh} m i_m s_h - (\nu_h + \mu_h) e_h \\ \frac{di_h}{dt} = \nu_h e_h - (\eta_h + \mu_h) i_h \\ \frac{dr_h}{dt} = \eta_h i_h - \mu_h r_h \\ \frac{da_m}{dt} = \varphi \frac{m}{k}(1 - a_m)(s_m + e_m + i_m) - (\eta_A + \mu_A) a_m \\ \frac{ds_m}{dt} = \eta_A \frac{k}{m} a_m - (B\beta_{hm} i_h + \mu_m) s_m - c s_m \\ \frac{de_m}{dt} = B\beta_{hm} i_h s_m - (\mu_m + \eta_m) e_m - c e_m \\ \frac{di_m}{dt} = \eta_m e_m - \mu_m i_m - c i_m \end{cases} \tag{9.2.1}$$

with the initial conditions

$$\begin{array}{llll} s_h(0) = 0.99865, & e_h(0) = 0.00035, & i_h(0) = 0.001, & r_h(0) = 0, \\ a_m(0) = 1, & s_m(0) = 1, & e_m(0) = 0, & i_m(0) = 0. \end{array}$$

Since any mathematical model is an abstraction of a complex natural system, additional assumptions are made to make the model mathematically treatable. This is also the case for the above epidemiological model, which comprises the following assumptions:

- the total human population (N_h) is constant;
- there is no immigration of infected individuals into the human population;
- the population is homogeneous, which means that every individual of a compartment is homogeneously mixed with other individuals;
- the coefficient of transmission of the disease is fixed and does not vary seasonally;
- both human and mosquitoes are assumed to be born susceptible, i.e., there is no natural protection;
- there is no resistant phase for mosquitoes, due to their short lifetime.

9.3 Multiobjective Approach

The most effective ways to reduce the costs related to the infected human population and the prevention measures to control the dengue disease transmission are investigated. In general, such problems are formulated and solved using optimal control (OC) theory, where a control problem includes a cost functional that is a function of state and control variables. The optimal control is calculated using the Pontryagin maximum principle. As the objective functional includes several different components, weights are often associated to each component that represent different decision-maker's preferences. Thus, attributing a larger weight to the cost of infected humans one can focus on the medical perspective to obtain as small as possible percentage of the affected population. On the other hand, attributing a larger weight to the insecticide cost, a solution representing economical perspective of savings for the prevention campaign can be obtained.

The aforementioned approach allows to obtain a single optimal solution that minimizes the cost functional formulated from some specific perspective. The most straightforward disadvantage of such approach is that not all optimal solutions can be obtained. This way, only a limited amount of information about a choice of the optimal strategy can be presented to the decision maker. Also, as it has been already emphasized in the present thesis, the goal of multiobjective optimization is to find a set of optimal solutions representing different trade-offs with respect to given objective functions. Much more information about optimal strategies can be obtained and presented to the decision maker solving a multiobjective optimization problem. Thus, a multiobjective approach to find optimal strategies for applying insecticide, taking into account the costs associated with the insecticide and infected humans, is formulated as follows:

$$\begin{aligned} \text{minimize:} \quad & f_1(c(t)) = \int_0^T i_h(t)\,dt \\ & f_2(c(t)) = \int_0^T c(t)\,dt \\ \text{subject to:} \quad & (9.2.1) \end{aligned} \tag{9.3.1}$$

where T is a period of time, f_1 and f_2 represent the total cost incurred in the form of infected population and the total cost of applying insecticide for the period T, respectively.

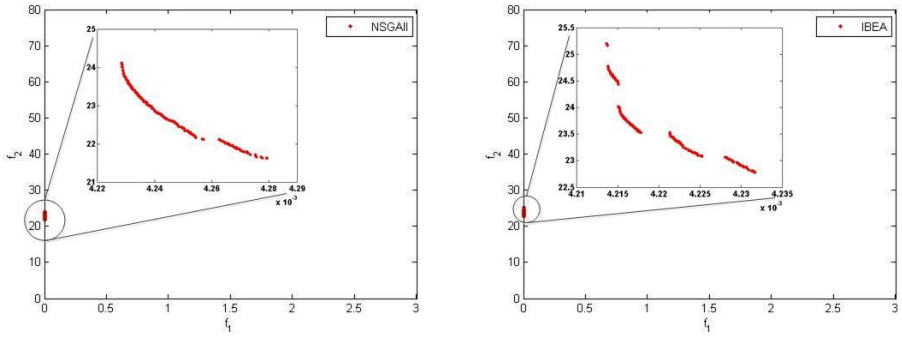
Experimental Setup

The constraints presented in (9.2.1) are numerically integrated using the fourth-order Runge-Kutta method with 1000 equally spaced time intervals over the period of 84 days. The decision variable $c(t)$ is also discretized ending up with a total of 1001 decision variables. Thus, the feasible decision space is $c \in [0,1]^{1001}$. Furthermore, the integrals used to determine objective function values in (9.3.1) are calculated using the trapezoidal rule. The problem is coded in the MATLAB® and Java™ programming languages. In order to solve this large-scale problem, the MATLAB® implementation of DDMOA2 is used, while NSGA-II [46], IBEA [198], GDE3 [116], MOEA/D [123], and SMPSO [135] are used within the jMetal framework [62].

For each algorithm, 30 independent runs are performed with a population size of 100, and 10^5 function evaluations. For each algorithm within the jMetal framework, the other parameters use the default settings. The parameter settings employed for DDMOA2 are: the initial step size for local search $\delta^{(0)} = 0.4$, the initial step size for reproduction $\sigma^{(0)} = 5$, the number of subpopulations $\alpha = 5$, and the tolerance for step size $\delta_{\text{tol}} = 10^{-3}$. Furthermore, descent direction $\boldsymbol{s}$ for subpopulation representative is accepted when trail solution $\boldsymbol{x} + \boldsymbol{s}$ is nondominated with respect to the current population.

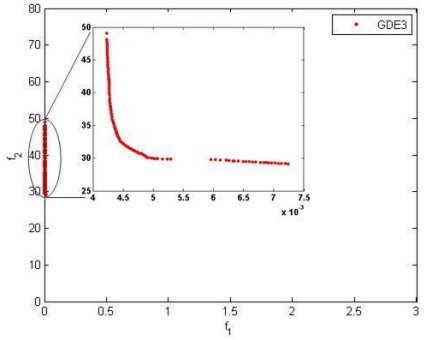
Experimental Results

Figure 9.3 depicts the sets containing all of the nondominated solutions obtained by each algorithm after 30 runs. The total infected human population is shown in the x-axis (f_1). The total cost of insecticide is shown in the y-axis (f_2). One can easily observe that all EMO algorithms, with the exception of DDMOA2, face significant difficulties in obtaining a set of well-distributed nondominated solutions in the objective space. The obtained solutions are located in very small regions, while the majority of the search space remains unexplored (Figures 9.3(a)–9.3(e)). However, DDMOA2 is the only algorithm able to provide a good spread in the obtained nondominated solutions (Figure 9.3(f)). DDMOA2 extensively explores the search space and provides a wide range of trade-off solutions.

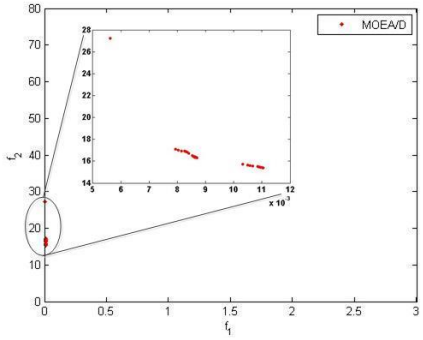


(a) NSGA-II

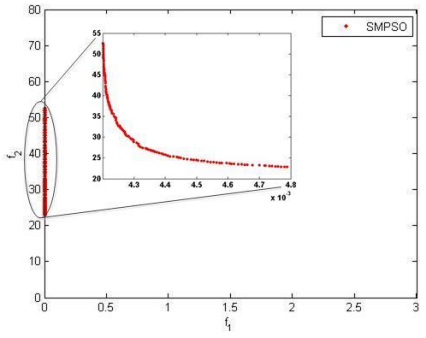
(b) IBEA



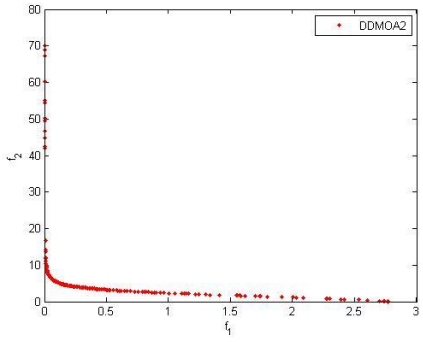
(c) GDE3



(d) MOEA/D

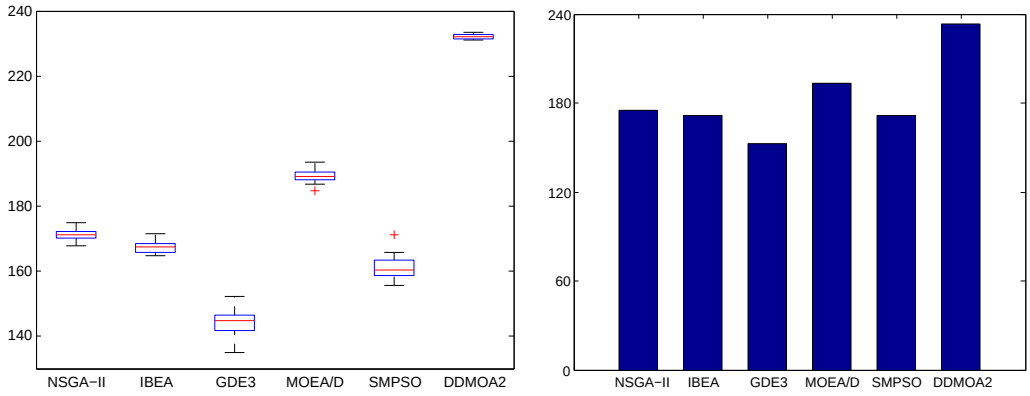


(e) SMPSO



(f) DDMOA2

Figure 9.3: Trade-off curves obtained by six different algorithms.



(a) distribution of hypervolume values (b) total hypervolume

Figure 9.4: Performance comparison of the algorithms in terms of the hypervolume on the dengue transmission model.

Thus, even visual comparison of the obtained results shows the superiority of DDMOA2 over the other EMO algorithms tested on the given real-world optimization problem.

To quantitatively assess the outcomes produced by the algorithms, the hypervolume is calculated using $[3, 80]$ as a reference point. Performance comparison of the algorithms with respect to the hypervolume is shown in Figure 9.4. The boxplots representing the distributions of the hypervolume values over the runs for each algorithm are depicted in Figure 9.4(a). It is interesting to note that the performance of genetic algorithms, namely NSGA-II (dominance-based) and IBEA (indicator-based), seems to be quite similar. However, NSGA-II gives a better spread of solutions that results in slightly higher values of the hypervolume. Two DE-based algorithms, namely GDE3 (dominance-based) and MOEA/D (scalarizing-based), perform differently. GDE3 has the worst performance, while MOEA/D behaves the best without considering DDMOA2. It can also be observed that the only PSO-based algorithm performs poorly. In turn, DDMOA2 yields the highest values of the hypervolume when compared to the other EMO algorithms. Moreover, the small variability of the achieved values highlights the robustness of DDMOA2. Performance comparison with respect to the total hypervolume achieved by all nondominated solutions obtained by each algorithm is presented in Figure 9.4(b). The data presented in this plot

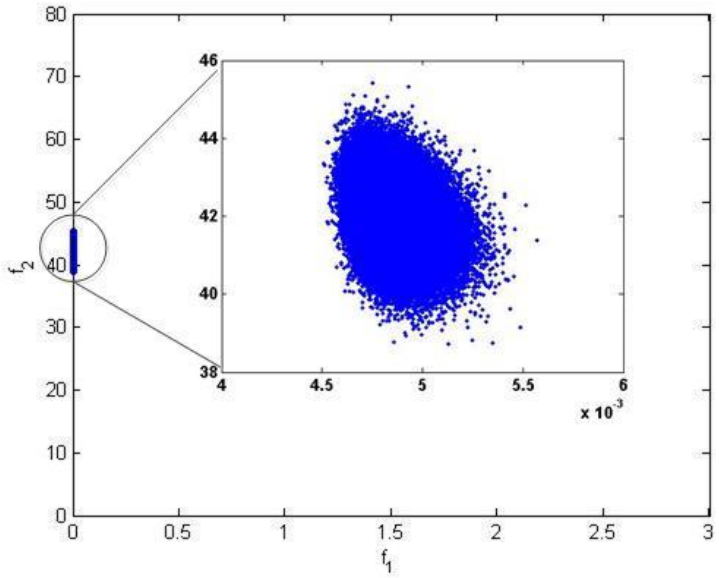


Figure 9.5: Mapping of uniformly sampled points in the decision space into the objective space defined by the dengue transmission model.

are consistent with the previous observations, being DDMOA2 again the algorithm with the better performance.

To better understand the difficulties in solving this multiobective optimization problem, 10^5 points are sampled within the 1001-dimensional unit hypercube (the feasible decision space) using uniform distribution. Figure 9.5 illustrates the mapping of these points into the objective space. It can be observed that the uniform distribution of solutions in the decision space does not correspond to a uniform distribution in the objective space. Solutions are mapped into a relatively small region of the objective space. Thus, the probability of getting solutions in the region shown in Figure 9.5 is much higher than anywhere in the objective space. This bias exacerbated by the high dimensionality of the decision space causes significant difficulties to the variation operators of MOEAs. As a result, they cannot embrace the whole range of trade-off solutions (Figure 9.3). On the other hand, DDMOA2 performs local search to find descent directions that allows to extensively explore the decision space. This fact appears to be extremely useful to deal with this problem.

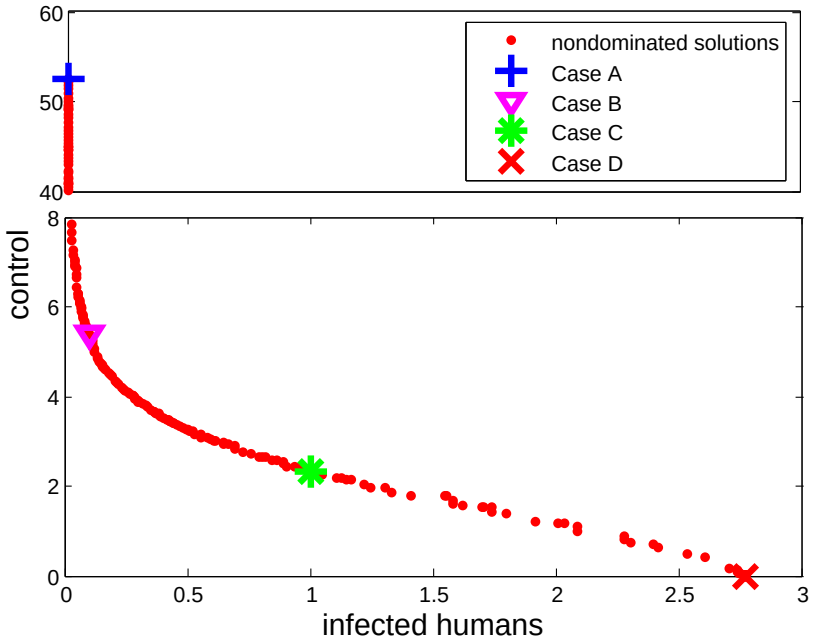
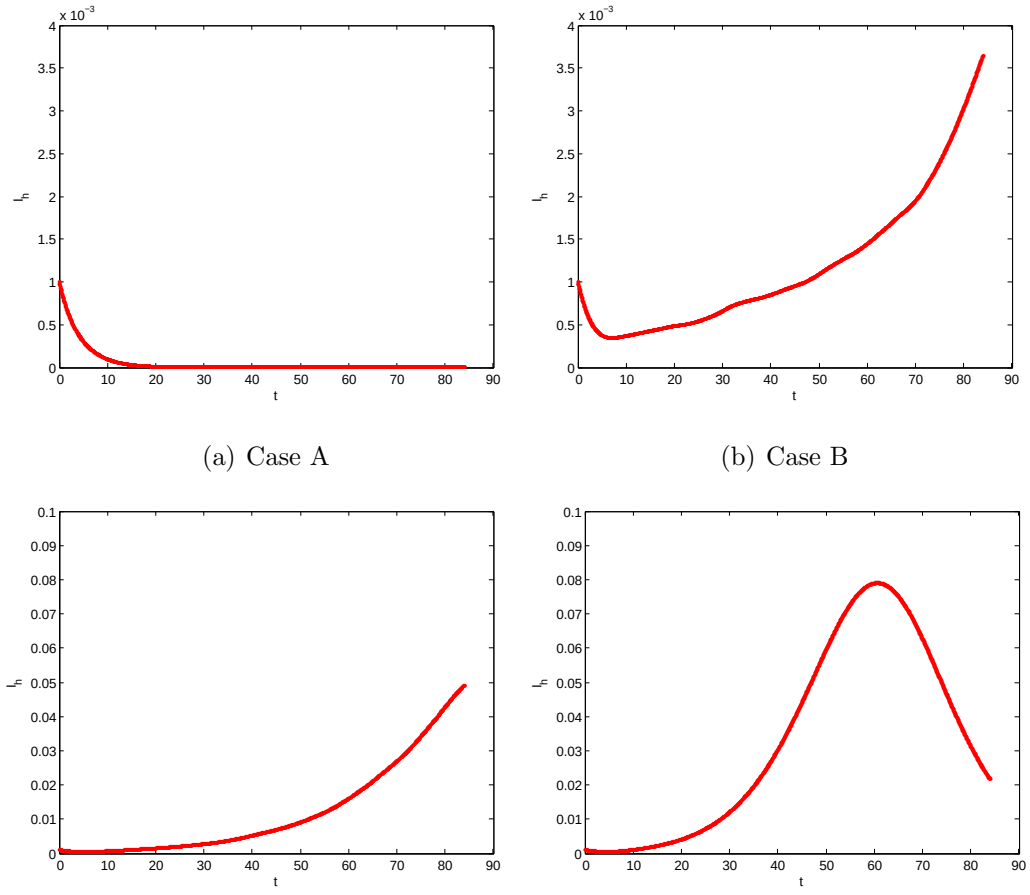


Figure 9.6: Trade-off curve for the infected population and the control.

For this particular problem, the above analysis allows to conclude that DDMOA2 works significantly better than the other tested algorithms. However, from a decision maker's perspective, instead of comparing the performance of the EMO algorithms, the main concern is a set of optimal solutions. Therefore, all obtained nondominated solutions produced by the algorithms are combined and a set of nondominated solutions is selected from the composed multiset. Figure 9.6 presents all nondominated solutions obtained afterwards. Observing this figure, one can see that for the insecticide cost in the range $0 \leq f_2 \leq 4$ there is almost linear dependency between the total infected human population (f_1) and the insecticide cost (f_2). Hence, reducing the number of infected humans from the worst scenario to 0.5 can be done at a relatively low cost. However, starting from some further point, say $f_1 = 0.5$, reducing the number of infected humans can be achieved through exponential increase in spendings for insecticide. Thus, even a small decrease in the number of infected humans corresponds to a high increase in expenses for insecticide. Scenarios represented by this part of the trade-off curve can be unacceptable from the economical point of view.



(a) Case A

(b) Case B

(c) Case C

(d) Case D

Figure 9.7: Different scenarios of the dengue epidemic.

Furthermore, Figure 9.6 depicts four distinct points (Case A, Case B, Case C, Case D) representing different parts of the trade-off curve, and, consequently, different scenarios of the dengue epidemic. For each case, Figure 9.7 plots the dynamics of the numbers of infected humans over the considered period of time.

Thus, Case A represents the medical perspective, when the number of infected humans is the lowest. From Figure 9.7(a), one can see that the number of infected people decreases from the very beginning. However, it is achieved through a huge expense for the insecticide.

Case B represents a point in the trade-off curve where an exponential dependency between the infected humans and the total control begins. From Figure 9.7(b), it can be

seen that at the beginning the number of infected humans decreases slightly. Thereafter, it grows steadily.

Case C represents a point located close to the part of the trade-off curve with seemingly linear dependency between the two objectives. Figure 9.7(b) shows that in this case the number of infected humans grows from the very first moment. A more rapid grow is observed in the second half of the considered period of time.

Finally, Case D represents the economical perspective, when the treatment for infected population is neglected and the main concern is the saving from not performing insecticide campaigns. Figure 9.7(d) illustrates that in Case D the number of infected humans grows rapidly from the very beginning, reaching its peak approximately on the 60-th day. After that the number of infected people decreases. This case corresponds to the worst scenario from the medical point of view.

9.4 Summary

This chapter presents a multiobjective approach to find an optimal control for reducing financial expenses caused by the outbreak of the dengue epidemic. The problem includes two clearly conflicting objectives. The first objective represents expenses due to the infected population. The second objective represents the total cost of applying insecticide in order to fight the disease.

Additionally, the performance comparison of DDMOA2 with other state-of-the-art EMO algorithms is carried out on this problem. The obtained results show that DDMOA2 significantly outperforms the other considered algorithms. DDMOA2 is able to present a wide range of trade-off solutions, whereas the other tested EMO algorithms face significant difficulties in solving this problem: the obtained solutions are located in small regions of the objective space, thus, presenting only a limited amount of alternatives.

The obtained nondominated solutions reveal different perspectives on applying insecticide: a low number of infected humans can be achieved spending larger financial recourses, whereas low spendings for prevention campaigns result in significant portions of the pop-

ulation affected by the disease. At the same time, a number of other solutions represent different trade-offs between the given objectives. Once the whole range of optimal solutions is obtained, the final decision on the control strategy can be made taking into consideration the available financial resources and goals of public health care. Hence, multiobjective optimization presents clear advantages over other approaches to find an optimal control strategy.

Chapter 10

Wastewater Treatment Plant Design

The high costs associated with the design and operation of wastewater treatment plants (WWTPs) motivate the research in the area of WWTP modelling and the water treatment process optimization. This chapter addresses different methodologies, which are based on defining and simultaneously optimizing several conflicting objectives, for finding the optimal values of the state variables in the WWTP design.

10.1 Activated Sludge System

A typical WWTP is schematically represented in Figure 10.1. There is a primary treatment, which is a physical process and aims to eliminate the gross solids and grease, so avoiding the blocking up of the secondary treatment. Although the dimensioning of such a unit is usually empirical and based on the wastewater to be treated, its cost is not affected by the (biological, chemical and biochemical) characteristics of the wastewater. The cost just corresponds to the civil engineering construction work of a tank. This is the reason why this process usually is not included in the optimization procedure. The next two units define the secondary treatment of the wastewater. This is the most important treatment in the plant because it eliminates the soluble pollutants. In the majority of the WWTPs, it is a biological process which, in the case herein studied, comprises an aeration tank and

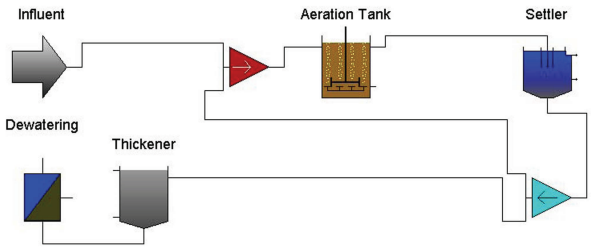


Figure 10.1: Schematic representation of a typical WWTP (adapted from [23]).

a clarifier that aims to separate the biological sludge from the treated water. There are other biological and chemical treatments but this is, by far, the most widely used. Finally, the last unit is used to treat the biological sludge that is wasted by the secondary settler. When the wastewater is very polluted and the secondary treatment does not provide the demanded quality, a tertiary treatment, usually a chemical process, can be included. There are many other possible WWTP layouts, but most of them are alike the above described.

The work herein presented focus solely on the secondary treatment, in particular on an activated sludge system that is represented in Figure 10.2. This system consists of an aeration tank and a secondary settler. The influent enters the aeration tank where the biological reactions take place, in order to remove the dissolved carbonaceous matter and nitrogen. The sludge that leaves this tank enters the secondary settler where suspended solids are removed. After this treatment, the treated final effluent leaves the settling tank and half of the thickened sludge is recycled back to the aeration tank and the rest of it is

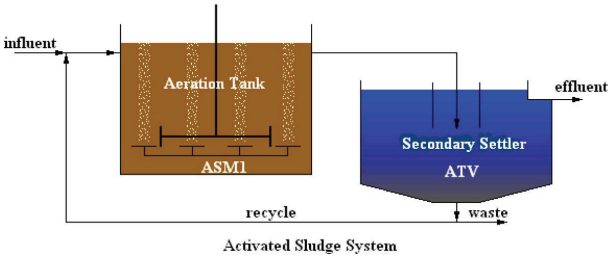


Figure 10.2: Schematic representation of the activated sludge system (adapted from the GPS-X simulator [23]).

wasted. The mathematical models used to describe the aeration tank and the settling tank are the ASM1 model [83], and the ATV model [63] combined with the double exponential model [172], respectively.

10.2 Mathematical Model

The mathematical model can be subdivided into seven types of equations. The system under study consists of an aeration tank, where the biological reactions take place, and a secondary settler for the sedimentation of the sludge and clarification of the effluent.

To describe the aeration tank, the activated sludge model n.1 (ASM1) described by Henze et al. [83] is used. ASM1 considers both the elimination of the carbonaceous matter and the removal of the nitrogen compounds. This model is widely accepted by the scientific community, as it produces good predictive values by simulation [2]. This means that all state variables keep their biological interpretation. The tank is considered a completely stirred tank reactor (CSTR) in steady state.

For the settling tank, a combination of the ATV design procedure [63] with the double exponential model model [172] is used. The ATV model is usually used as a design procedure to new WWTPs. It is based on empirical equations obtained by experiments and does not contain any solid balances, although it contemplates peak wet weather flow (PWWF) events. The double exponential model model is the most widely used in simulations and it produces results very close to reality. However, since it does not provide extra sedimentation area needed during PWWF events, the resulting design has to consider the use of security factors that yield an over-dimensioned and expensive unit.

In [151], it is shown that this combined model is prepared to overcome PWWF events without over dimensioning and provides the most equilibrated WWTP design when compared with the other two used separately. When these three designs were introduced in the GPS-X simulator [23] and a stress condition of a PWWF value of five times the normal flow was imposed, only the combined model was able to support this adverse condition maintaining the quality of the effluent under the values imposed by the portuguese law.

Mass Balances around Aeration Tank

The first set of equations come from the mass balances around the aeration tank. The Peterson matrix of the ASM1 model [83] is used to define the model for the mass balances. For a CSTR, it is assumed that the mass of a given component entering the tank minus the mass of the same compound in the tank, plus a reaction term (positive or negative) equals the accumulation in the tank of the same compound:

$$\frac{Q}{V_a}(\xi_{in} - \xi) + r_\xi = \frac{d\xi}{dt}. \tag{10.2.1}$$

It is convenient to refer that in a CSTR the concentration of a compound is the same at any point inside the reactor and at the effluent of that reactor. The reaction term for the compound in question, r_ξ, is obtained by the sum of the product of the stoichiometric coefficients, $\nu_{\xi j}$, with the expression of the process reaction rate, ρ_j, of the ASM1 Peterson matrix [83]

$$r_\xi = \sum_j \nu_{\xi j}\rho_j. \tag{10.2.2}$$

In steady state, the accumulation term given by $\frac{d\xi}{dt}$ is zero, because the concentration of a given compound is constant in time. The ASM1 model involves 8 processes incorporating 13 different components. The mass balances for the inert materials, S_I and X_I, are not considered because they are transport-only components.

All the symbols used in the WWTP modelling are listed in the Appendix. The process rates are the following:

- Aerobic growth of heterotrophs

$$\rho_1 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{OH} + S_O}\right) X_{BH}; \tag{10.2.3}$$

- Anoxic growth of heterotrophs

$$\rho_2 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{OH}}{K_{OH} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{BH}; \tag{10.2.4}$$

- Aerobic growth of autotrophs

$$\rho_3 = \mu_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}} \right) \left(\frac{S_O}{K_{OA} + S_O} \right) X_{BA}; \tag{10.2.5}$$

- Decay of heterotrophs

$$\rho_4 = b_H X_{BH}; \tag{10.2.6}$$

- Decay of autotrophs

$$\rho_5 = b_A X_{BA}; \tag{10.2.7}$$

- Ammonification of soluble organic nitrogen

$$\rho_6 = k_a S_{ND} X_{BH}; \tag{10.2.8}$$

- Hydrolysis of entrapped organics

$$\begin{aligned} \rho_7 \quad &= k_h \frac{\frac{X_S}{X_{BH}}}{K_X + \frac{X_S}{X_{BH}}} \left[\left(\frac{S_O}{K_{OH} + S_O} \right) \right. \\ &\left. + \eta_h \left(\frac{K_{OH}}{K_{OH} + S_O} \right) \left(\frac{S_{NO}}{K_{NO} + S_{NO}} \right) \right] X_{BH}; \end{aligned} \tag{10.2.9}$$

- Hydrolysis of entrapped organic nitrogen

$$\rho_8 = \rho_7 \frac{X_{ND}}{X_S}. \tag{10.2.10}$$

The unit adopted for concentration is g COD/m^3 and the equations obtained from the ASM1 model with mass balances are as follows:

- Soluble substrate (S_S)

$$\frac{Q}{V_a} (S_{S_{in}} - S_S) - \frac{1}{Y_H} \rho_1 - \frac{1}{Y_H} \rho_2 + \rho_7 = 0; \tag{10.2.11}$$

- Slowly biodegradable substrate (X_S)

$$\frac{Q}{V_a} (X_{S_{in}} - X_S) + (1 - f_P) \rho_4 + (1 - f_P) \rho_5 - \rho_7 = 0; \tag{10.2.12}$$

- Heterotrophic active biomass (X_{BH})

$$\frac{Q}{V_a}\left(X_{BH_{in}} - X_{BH}\right) + \rho_1 + \rho_2 - \rho_4 = 0; \tag{10.2.13}$$

- Autotrophic active biomass (X_{BA})

$$\frac{Q}{V_a}\left(X_{BA_{in}} - X_{BA}\right) + \rho_3 - \rho_5 = 0; \tag{10.2.14}$$

- Particulate products arising from biomass decay (X_P)

$$\frac{Q}{V_a}\left(X_{P_{in}} - X_P\right) + f_P\rho_4 + f_P\rho_5 = 0; \tag{10.2.15}$$

- Nitrate and nitrite nitrogen (S_{NO})

$$\frac{Q}{V_a}\left(S_{NO_{in}} - S_{NO}\right) - \frac{1 - Y_H}{2.86Y_H}\rho_2 + \frac{1}{Y_A}\rho_3 = 0; \tag{10.2.16}$$

- $NH_4^+ + NH_3$ nitrogen (S_{NH})

$$\frac{Q}{V_a}\left(S_{NH_{in}} - S_{NH}\right) - i_{X_B}\rho_1 - i_{X_B}\rho_2 - \left(i_{X_B} + \frac{1}{Y_A}\right)\rho_3 + \rho_6 = 0; \tag{10.2.17}$$

- Soluble biodegradable organic nitrogen (S_{ND})

$$\frac{Q}{V_a}\left(S_{ND_{in}} - S_{ND}\right) - \rho_6 + \rho_8 = 0; \tag{10.2.18}$$

- Particulate biodegradable organic nitrogen (X_{ND})

$$\frac{Q}{V_a}\left(X_{ND_{in}} - X_{ND}\right) + \left(i_{X_B} - f_P i_{X_P}\right)\rho_4 + \left(i_{X_B} - f_P i_{X_P}\right)\rho_5 - \rho_8 = 0; \tag{10.2.19}$$

- Alkalinity (S_{alk})

$$\begin{aligned}\tfrac{Q}{V_a}\left(S_{alk_{in}} - S_{alk}\right) \quad & -\tfrac{i_{X_B}}{14}\rho_1 + \left(\tfrac{1-Y_H}{14\times 2.86Y_H} - \tfrac{i_{X_B}}{14}\right)\rho_2 \\ & - \left(\tfrac{i_{X_B}}{14} + \tfrac{1}{7Y_A}\right)\rho_3 + \tfrac{1}{14}\rho_6 = 0;\end{aligned} \tag{10.2.20}$$

- Oxygen (S_O)

$$\frac{Q}{V_a}\left(S_{O_{in}} - S_O\right) + K_L a\left(S_{O_{sat}} - S_O\right) - \frac{1 - Y_H}{Y_H}\rho_1 - \frac{4.57 - Y_A}{Y_A}\rho_3 = 0, \tag{10.2.21}$$

where Y_A, Y_H, f_P, i_{X_B} and i_{X_P} are stoichiometric parameters, and $K_L a$ is the overall mass transfer coefficient.

For oxygen mass transfer, aeration by diffusion is considered:

$$K_L a = \frac{\alpha\ G_S\ \eta\ P_{O_2}\ 1333.3}{V_a S_{O_{sat}}} \theta^{(T-20)} \tag{10.2.22}$$

where

$$S_{O_{sat}} = \frac{1777.8 \beta\, \rho\, P_{O_2}}{HenryO_2}, \tag{10.2.23}$$

$$\rho = 999.96(2.29 \times 10^{-2} T) - (5.44 \times 10^{-3} T^2), \tag{10.2.24}$$

$$HenryO_2 = 708\ T + 25700, \tag{10.2.25}$$

G_S is the air flow rate and α, β, ρ, η, P_{O_2}, T and θ are operational parameters [22].

Composite Variables

In a real system, some state variables are, most of the time, not available from direct measurements. Thus, readily measured composite variables are used instead. They are defined as follows:

- Particulate chemical oxygen demand

$$X = X_I + X_S + X_{BH} + X_{BA} + X_P; \tag{10.2.26}$$

- Soluble chemical oxygen demand

$$S = S_I + S_S; \tag{10.2.27}$$

- Chemical oxygen demand

$$COD = X + S; \tag{10.2.28}$$

- Volatile suspended solids

$$VSS = \frac{X}{icv}; \tag{10.2.29}$$

- Total suspended solids

$$TSS = VSS + ISS; \tag{10.2.30}$$

- Biochemical oxygen demand

$$BOD = f_{BOD}\left(S_S + X_S + X_{BH} + X_{BA}\right); \tag{10.2.31}$$

- Total nitrogen of Kjeldahl

$$TKN = S_{NH} + S_{ND} + X_{ND} + i_{X_B}\left(X_{BH} + X_{BA}\right) + i_{X_P}\left(X_P + X_I\right); \tag{10.2.32}$$

- Total nitrogen

$$N = TKN + S_{NO}; \tag{10.2.33}$$

where icv and f_{BOD} define ratios to convert units.

Quality Constraints

Quality constraints are usually derived from environmental law restrictions. The most used are related with limits in COD, N, and TSS at the effluent. In mathematical terms, these constraints are defined as:

$$\begin{aligned} COD_{ef} &\leq COD_{law} \\ N_{ef} &\leq N_{law} \\ TSS_{ef} &\leq TSS_{law} \end{aligned} \tag{10.2.34}$$

where the subscript "ef" stands for effluent.

Secondary Settler Constraints

Traditionally, the importance of the secondary settler is underestimated when compared with the aeration tank. However, it plays a crucial role in the activated sludge system. For example, the clarification efficiency of the settling tank has great influence on the treatment plant efficiency because the particulate fraction arising from biomass contributes to the

major portion of the effluent COD. Further, it has been observed that the investment cost of a typical settling tank in a WWTP context could reach 25% of the total [73]. Thus, when trying to reduce both investment and operation costs, the importance of the secondary settler is by far emphasized.

When the wastewater leaves the aeration tank, where the biological treatment took place, the treated water should be separated from the biological sludge, otherwise, the COD would be higher than it is at the entry of the system. The most common way of achieving this purpose is by sedimentation in tanks.

A good settling tank has to accomplish three different functions. As a thickener, it aims to produce a continuous underflow of thickened sludge to return to the aeration tank; as a clarifier, it produces a good quality final effluent; and as a storage tank it allows the conservation of the sludge in peak flow events. None of these functions could fail. If that happens the effluent will be of poor quality and the overall behavior of the system can be compromised.

The behavior of a settling tank depends on its design and operation, namely the hydraulic features, as the flow rate, the physical features, as inlet and sludge collection arrangements, site conditions, as temperature and wind, and sludge characteristics. The factors that most influence the size of the tank are the wastewater flow and the characteristics of the sludge. As the influent flow is known, the optimization of the sedimentation area and depth must rely on the sludge characteristics, which in turn are related with the performance of the aeration tank. So, the operation of the biological reactor influences directly the performance of the settling tank and for that reason, one should never be considered without the other.

The ATV design procedure contemplates the PWWF events, in which the sludge mass transferred from the biological reactor is $\Delta X\, V_a$, where ΔX is the change in the sludge concentration within the aeration tank. A reduction of 30% on the sludge concentration for a PWWF event is considered. A higher reduction of the sludge concentration into the biological reactor may compromise the entire process.

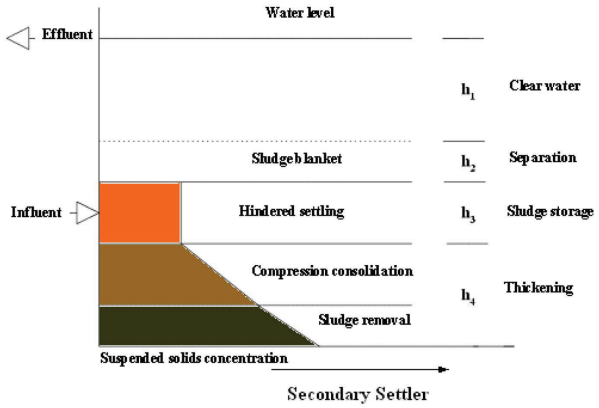


Figure 10.3: Typical solids concentration-depth profile adopted by the ATV design procedure (adapted from [63]).

A way of turning around this problem is to allocate a certain depth (h_3 from Figure 10.3) to support the fluctuation of solids during these events. Thus, the sludge storage depth depends on the mass that needs to be stored during a PWWF and is given by

$$h_3 = \Delta X V_a \frac{DSVI}{480 A_s}, \tag{10.2.35}$$

where A_s is the sedimentation area and $DVSI$ is the diluted sludge volume index. When this zone is considered, a reduction in the sedimentation area is allowed.

The transferred sludge causes the biological sludge concentration in the reactor at PWWF to decline, which allows a higher overflow rate and therefore a smaller surface area. However, the greater the decrease in reactor concentration is, the greater is the mass of sludge to be stored in the settler tank, so the deeper the tank needs to be. The ATV procedure allows a trade-off between surface area and depth and one may select the area/depth combination that suites better the particular site under consideration.

The compaction zone, h_4, where the sludge is thickened in order to achieve the convenient concentration to return to the biological reactor, depends only on the characteristics

of the sludge, and is given by

$$h_4 = X_p \frac{DSVI}{1000} \tag{10.2.36}$$

where X_p is the sludge concentration in the biological reactor during a PWWF event.

The clear water zone, h_1, and the separation zone, h_2, are set empirically to 1 m ($h_1 + h_2 = 1$). The depth of the settling tank, h, is the sum of these four zones.

The sedimentation area is still related to the peak flow, Q_p, by the expression

$$\frac{Q_p}{A_s} \leq 2400 \left(X_p DSVI\right)^{-1.34}. \tag{10.2.37}$$

The double exponential model assumes a one dimensional settler, in which the tank is divided into ten layers of equal thickness (Figure 10.4). Some simplifications are considered. No biological reactions take place in this tank, meaning that the dissolved matter concentration is maintained across all the layers. Only vertical flux is considered and the solids are uniformly distributed across the entire cross-sectional area of the feed layer ($j = 7$). This model is based on a traditional solids flux analysis but the flux in a particular layer is limited by what can be handled by the adjacent layer. The settling function, described by Takács et al. in [172], which represents the settling velocity, is given by

$$\nu_{s,j} = \max\left(0, \min\left(\nu_0', w_0\right)\right) \tag{10.2.38}$$

where $\nu_{s,j}$ is the settling velocity in layer j,

$$w_0 = \nu_0 \left(e^{-r_h(TSS_j - f_{ns}TSS_a)} - e^{-r_p(TSS_j - f_{ns}TSS_a)}\right), \tag{10.2.39}$$

TSS_j is the total suspended solids concentration in each of the ten considered layers of the settler, TSS_a is the TSS in the feed layer ($TSS_a = TSS_7$) and ν_0, ν_0', r_h, r_p and f_{ns} are the settling parameters [23].

The solids flux due to the bulk movement of liquid may be up or down, ν_{up} and ν_{dn}

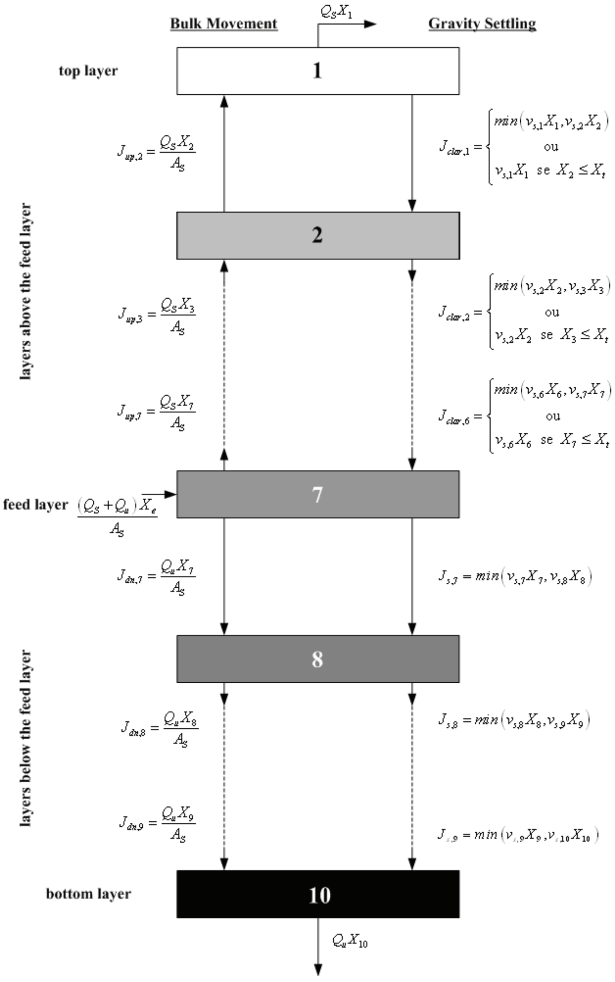


Figure 10.4: Solids balance around the settler layers according to the double exponential model (adapted from [172]).

respectively, depending on its position relative to the feed layer, thus

$$\nu_{up} = \frac{Q_{ef}}{A_s} \quad \text{and} \quad \nu_{dn} = \frac{Q_r + Q_w}{A_s}. \tag{10.2.40}$$

The subscript "r" is concerned with the recycled sludge and "w" refers to the wasted sludge.

The sedimentation flux, J_s, for the layers under the feed layer ($j = 7, \ldots, 10$) is given

by

$$J_{s,j} = \nu_{s,j} TSS_j \tag{10.2.41}$$

and above the feed layer ($j = 1, \ldots, 6$) the clarification flux, J_{clar}, is given by

$$J_{clar,j} = \begin{cases} \nu_{s,j} TSS_j & \text{if } TSS_{j+1} \leq TSS_t \\ \min(\nu_{s,j} TSS_j, \nu_{s,j+1} TSS_{j+1}) & \text{otherwise} \end{cases} \tag{10.2.42}$$

where TSS_t is the threshold concentration of the sludge. The resulting solids balances around each layer, considering steady state, are the following:

- for the layers above the feed layer ($j = 1, \ldots, 6$)

$$\frac{\nu_{up}(TSS_{j+1} - TSS_j) + J_{clar,j-1} - J_{clar,j}}{h/10} = 0, \tag{10.2.43}$$

- for the feed layer ($j = 7$)

$$\frac{\frac{Q\ TSS_a}{A_s} + J_{clar,j-1} - (\nu_{up} + \nu_{dn}) TSS_j - \min(J_{s,j}, J_{s,j+1})}{h/10} = 0, \tag{10.2.44}$$

- for the intermediate layers under the feed layer ($j = 8, \ldots, 10$)

$$\frac{\nu_{dn}(TSS_{j-1} - TSS_j) + \min(J_{s,j}, J_{s,j-1}) - \min(J_{s,j}, J_{s,j+1})}{h/10} = 0. \tag{10.2.45}$$

  By convention, $J_{clar,0} = J_{s,11} = 0$.

The use of the combination of these two models to describe the secondary settler is prepared to turn around the PWWF events without over dimensioning and overcomes the limitations and powers the advantages of each one.

Flow and Mass Balances

The system behavior, in terms of concentration and flows, may be predicted by balances. In order to achieve a consistent system, these balances must be done around the entire system and not only around each unit operation. This is crucial to reinforce the robustness of the model. Furthermore, these balances may not be a sum of the mass balances

of the individual components since the PWWF events are contemplated in the ATV design included in the settler modelling. The balances were done to the suspended matter, dissolved matter and flows.

In the case of the suspended matter, the mass balances concern the organic (X) and inorganic (X_{II}) solids:

$$(1+r)\,Q_{inf}X_{in} = Q_{inf}X_{inf} + (1+r)\,Q_{inf}X - \frac{V_aX}{SRT\,X_r}\,(X_r - X_{ef}) - Q_{inf}X_{ef}, \quad (10.2.46)$$

$$Q_{inf}0.2\,TSS_{inf} = \frac{V_aX_{II}}{SRT\,X_r}\left(X_{II_r} - X_{II_{ef}}\right) + Q_{inf}X_{II_{ef}}, \quad (10.2.47)$$

where r is the recycle rate. The subscripts "inf" and "in" denote the influent and the entry of the aeration tank, respectively.

The balances of the dissolved matter are done for each one of the dissolved components S_S, S_O, S_{NO}, S_{NH}, S_{ND}, S_{alk}, as shown below in the S_S case:

$$(1+r)\,Q_{inf}S_{S_{in}} = Q_{inf}S_{S_{inf}} + rQ_{inf}S_{S_r}. \quad (10.2.48)$$

Besides the mass balances, flow balances are also necessary:

$$Q = Q_{inf} + Q_r \quad \text{and} \quad Q = Q_{ef} + Q_r + Q_w. \quad (10.2.49)$$

System Variables Definition

To complete the model, some definitions are added:

- Sludge retention time

$$SRT = \frac{V_aX}{Q_wX_r}; \quad (10.2.50)$$

- Hydraulic retention time

$$HRT = \frac{V_a}{Q}; \quad (10.2.51)$$

- Recycle rate

$$r = \frac{Q_r}{Q_{inf}}; \tag{10.2.52}$$

$$r = \frac{TSS}{TSS_{r_{max}} - TSS}; \tag{10.2.53}$$

- Recycle rate in a PWWF event

$$r_p = \frac{0.7TSS}{TSS_{max_p} - 0.7TSS}; \tag{10.2.54}$$

- Recycle flow rate during a PWWF event

$$Q_{r_p} = r_p Q_p; \tag{10.2.55}$$

- Maximum overflow rate

$$\frac{Q_p}{A_s} \leq 2. \tag{10.2.56}$$

A fixed value for the relation between volatile and total suspended solids was considered

$$\frac{VSS}{TSS} = 0.7, \tag{10.2.57}$$

where $TSS_{r_{max}}$ is the maximum total suspended solids concentration allowed in the recycle flow and TSS_{max_p} is the maximum total suspended solids concentration allowed in the recycle flow during a PWWF event.

Simple Bounds

All variables must be nonnegative, although more restricted bounds are imposed on some of them due to operational consistencies, namely:

$$\begin{array}{cc} 0 \leq K_L a \leq 300 & 0.05 \leq HRT \leq 2 \\ 800 \leq TSS \leq 6000 & 0.5 \leq r \leq 2 \\ 2500 \leq TSS_r \leq 10000 & 6 \leq S_{alk} \leq 8 \\ 6 \leq S_{alk_{in}} \leq 8 & S_O \geq 2. \end{array} \tag{10.2.58}$$

10.3 Multiobjective Approach

In the following, the WWTP design is optimized in terms of a secondary treatment, in a way that the strict laws on effluent quality are accomplished. In essence, the WWTP design optimization consists in minimizing the total cost, hereupon denoted by TC, which is the sum of investment and operation costs, and maximizing the effluent quality measured by a quality index function, represented by the variable QI.

Total Cost Function

The cost function represents the total cost and includes both investment and operation costs. For the sake of simplicity, no pumps are considered, which means that all the flows in the system move by the effect of gravity. The total cost is given by the sum of the investment (IC) and operation (OC) costs. To obtain a cost function based on portuguese real data, a study was carried out with a WWTP building company. The basic structure of the model is $C = a\,Z^b$ [176], where a and b are parameters that depend on the region where the WWTP is being built, and have to be estimated. Variable Z is the characteristic of the unit operation that is influencing the cost, for example, the volume V_a and the air flow G_S for the case of the aeration tank. Parameters a and b were estimated by a least squares technique, giving the following investment cost function for the aeration tank:

$$IC_a = 148.6V_a^{1.07} + 7737G_S^{0.62}. \tag{10.3.1}$$

The operation cost is usually estimated on an annual basis, so it has to be updated to a present value using the updating term Γ:

$$\Gamma = \sum_{j=1}^{\mathcal{N}} \frac{1}{(1+i)^j} = \frac{1-(1+i)^{-\mathcal{N}}}{i}, \tag{10.3.2}$$

where i represents the discount rate (rate of return), i.e., the rate that is used to valuing a project using the concept of the time value of money, over a certain amount of time, for example, $\mathcal{N}$ years. This is also taken as the average life-expectancy of a WWTP. In this study, $i = 0.05$ and $\mathcal{N} = 20$ years are used. Since the collected data come from a

set of WWTPs in design, operation data are not available. However, from the company experience, the expected life span for the civil engineering construction works is 20 years and the maintenance expenses are around 1% of the investment costs during the first 10 years and around 2% during the remaining ones. Although the replacement costs of the electromechanical components are negligible, they are usually replaced after 10 years. The predominant cost comes from the energy used for pumping the air flow into the aeration tank. The power cost (P_c) in Portugal is 0.08 €/kWh. With this information and with the updating term Γ in (10.3.2), the operation cost of the aeration tank is then

$$OC_a = \left[0.01\Gamma + 0.02\Gamma\,(1+i)^{-10}\right] 148.6V_a^{1.07} + (1+i)^{-10} 7737 G_S^{0.62} + 115.1\Gamma P_c G_S. \quad (10.3.3)$$

The term $(1+i)^{-10}$ is used to bring to present a future value, in this case, 10 years from now.

Similarly, the least squares technique is used to fit the basic model to the available data, and the correspondent investment cost function

$$IC_s = 955.5 A_s^{0.97}, \quad (10.3.4)$$

and the operation cost function, that is concerned only with the maintenance for the civil construction,

$$OC_s = \left[0.01\Gamma + 0.02\Gamma\,(1+i)^{-10}\right] 148.6(A_s\,h)^{1.07} \quad (10.3.5)$$

are obtained for the settling tank. The objective cost function (TC) is then given by the sum of all the previous functions:

$$TC = 174.2V_a^{1.07} + 12487 G_S^{0.62} + 114.8 G_S + 955.5 A_s^{0.97} + 41.3\,(A_s\,h)^{1.07}. \quad (10.3.6)$$

Quality Index Function

To be able to attain effluent quality at a required level, a quality index function may be used to measure the amount of pollution in the effluent.

The quality index (QI) defined by the BSM1 model [2] gives a measure of the amount of daily pollution, in average terms during seven days. It depends on the quality of the

effluent in terms of the total suspended solids (TSS), the chemical oxygen demand (COD), the biochemical oxygen demand (BOD), the total Kjeldahl nitrogen (TKN), the nitrate and nitrite nitrogen (S_{NO}) and the effluent flow (Q_{ef}). The obtained function is:

$$QI = (2\,TSS + COD + 2\,BOD + 20\,TKN + 2\,S_{NO})\frac{Q_{ef}}{1000}. \tag{10.3.7}$$

Experimental Setup

The presented model is coded in the MATLAB® programming language. The problem consists of 2 objective functions, 115 decision variables, 1 inequality constraint, and 99 equality constraints. All the variables are bounded below and above. To solve this problem EMyO-C is used.

In order to handle constraints, the original nondominated sorting used in EMyO-C is modified introducing the constrained-domination principle [46]. Under this principle, solution a is said to constrained-dominate solution b, if any of the following conditions is true: (i) solution a is feasible and solution b is not, (ii) solutions a and b are both infeasible, but solution a has a smaller overall constraint violation, (iii) solutions a and b are feasible and solution a dominates solution b. For a given solution, the overall constraint violation cv is calculated as: $cv = \sum_{i=1}^{p} \max\{0\,, g_i(\boldsymbol{x})\} + \sum_{j=1}^{q} |h_j(\boldsymbol{x})|$.

The major difficulty in solving this real-world problem is related to the presence of a larger number of equality constraints. So to successfully solve this highly constrained problem, the experiments are conducted in several steps. First, 30 independent runs of EMyO-C were performed with a population size of $\mu = 1000$, running for 1000 generations. Further parameter settings were the same as discussed in Chapter 8. In each run, the initial population was randomly generated within the bounds. As a result, no feasible solution was found. Therefore, all the produced outcomes were combined and one nondominated solution was selected from the resulting set based on the concept of constrained-domination. Afterwards, this solution was improved using a single-objective optimizer to find a feasible solution. For this purpose, HGPSAL was used. After obtaining the feasible solution, further experiments are conducted using EMyO-C. In this instance, 30 independent runs

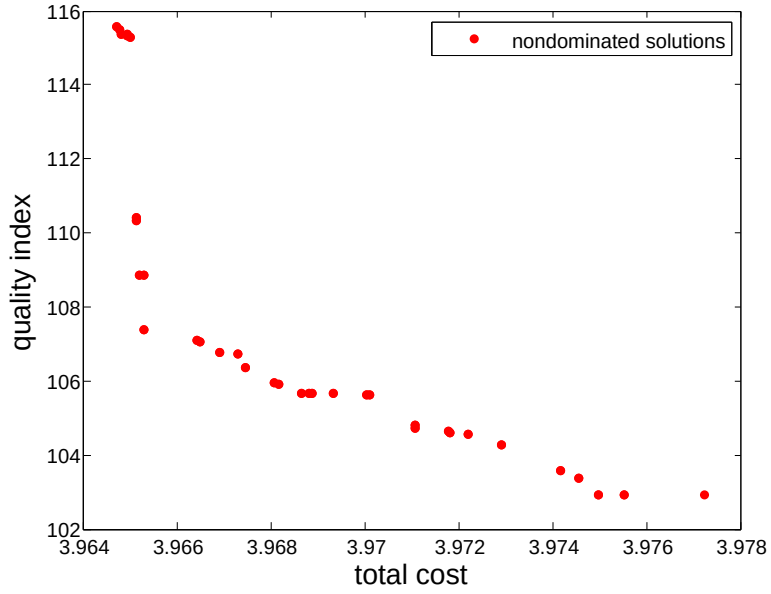


Figure 10.5: Trade-off curve for the total cost and quality index.

of EMyO-C are performed with a population size of $\mu = 300$, running for 300 generations. Further parameter settings are the same as in the previous experiment. In each run, the feasible solution is introduced in the initial population, whereas the remaining individuals are randomly generated.

Experimental Results

Figure 10.5 depicts all nondominated solutions obtained after the performed experiments. The values of the TC are in millions of euros (M €). In this figure, the compromise solutions representing trade-offs between the total cost (TC) and quality index (QI) are plotted. Observing this figure, it can be seen that the proposed approach to the WWTP design produces results with a physical meaning. Specifically, the lower values of the quality index can be achieved through the increase in the total cost, while the smaller values of the total cost result in the larger amount of pollution in the effluent, measured by the quality index.

In Table 10.1, the objective values along with the most important decision variables of the obtained trade-off solutions are presented, namely, the aeration tank volume (V_a),

V_a	G_S	A_s	h	N	COD	TSS	TC(M €)	QI
100	6597.8531	271.7075	1	1.8982	125	10.7124	3.9647	115.5605
100	6598.438	271.7075	1	1.8982	125	10.4436	3.9649	115.3593
100	6598.9164	271.7075	1	1.3069	125	11.7056	3.9651	110.4133
100	6599.3342	271.7075	1	1.0247	125	11.7056	3.9653	108.8466
100	6599.363	271.7075	1.0637	0.90931	125	10.8241	3.9664	107.1068
100	6599.363	271.7075	1.0906	0.77282	125	10.8241	3.9669	106.7584
100	6599.363	271.7075	1.1116	0.83292	125	10.9469	3.9673	106.7376
100	6599.363	271.7075	1.1208	0.77282	125	10.8241	3.9674	106.3607
100	6599.363	271.7075	1.1598	0.71443	125	10.8241	3.9681	105.9032
100	6599.363	271.7075	1.1876	0.64562	125	10.82	3.9686	105.6695
100	6599.363	271.7075	1.1961	0.64562	125	10.82	3.9688	105.6695
100	6599.363	271.7075	1.2245	0.64562	125	10.82	3.9693	105.6695
100	6599.363	271.7075	1.2649	0.64562	125	10.7742	3.97	105.6352
100	6599.363	271.7075	1.3209	0.53704	125	10.8241	3.971	104.7857
100	6599.363	271.7075	1.3631	0.55629	125	10.8241	3.9718	104.6122
100	6599.363	271.7075	1.3835	0.55629	125	10.8241	3.9722	104.5415
100	6599.363	271.7075	1.4222	0.50336	125	10.8241	3.9729	104.2785
100	6599.363	271.7075	1.4924	0.32773	125	10.8241	3.9742	103.5757
100	6599.363	271.7075	1.5123	0.32773	125	10.8241	3.9745	103.3636
100	6599.363	271.7075	1.5359	0.32773	125	10.8241	3.9749	102.9405
100	6599.363	271.7075	1.5661	0.32773	125	10.8241	3.9755	102.9405
100	6599.363	271.7075	1.6586	0.34713	125	10.9389	3.9772	102.9399

Table 10.1: Optimal values for the most important variables obtained using multiobjective approach.

the air flow rate (G_S), the sedimentation area (A_s), the settler tank depth (h), the total nitrogen (N), the chemical oxygen demand (COD), the total suspended solid (TSS). The presented solutions are obtained after applying clustering procedure to group the data in the objective space. This way, each solution represents a distinct cluster, hence a different part of the trade-off curve. This procedure allows to reduce the number of points, thereby facilitating the visualization of the results. Thus, different design perspectives can be easily observed from the obtained solutions.

It can be seen that the aeration tank volume and the sedimentation area maintain the same values in all the presented solutions. Only the slight variation in a few solutions is observed concerning the air flow rate, whereas the settler depth has the larger variation in the values between all the variables composing the cost function. This hints that the lower cost as well as the desirable values of the quality index can be mostly achieved through controlling the settler depth. Furthermore, it can be seen that the chemical oxygen demand is at the highest allowable level (125), whereas the total nitrogen and the total suspended solid are far below the law limits (15 and 35, respectively).

10.4 Many-Objective Approach

In the following, the WWTP design is optimized by simultaneously minimizing the variables that influence the operation and investment costs as well as the quality index function. These variables are referred as to the influential variables. This approach results in a many-objective optimization problem.

Influential Variables

To avoid the use of cost functions that are time and local dependent, four objective functions, each describing a variable that influences the investment and operation costs of a WWTP, in each unit operation, are used.

As far as the aeration tank is concerned, the variables that mostly influence the costs are the volume (V_a) and the air flow (G_S). In terms of investment, the first variable

influences directly the cost of the construction of the tank, and the second influences the required power of the air pumps. In terms of operation, both variables will determine the power needed to aerate the sludge properly, as well as the maintenance, in terms of electromechanical and civil construction material, due to deterioration.

As to the secondary settler, and assuming that the settling process is only due to the gravity, the variables that most influence the costs are the sedimentation area (A_s) and the tank depth (h), for obvious reasons.

Concerning the quality index function, each variable on the right-hand side of (10.3.7) is considered as a distinct function, thereby adding to the optimization problem six different objectives.

Experimental Setup

The problem is coded in the MATLAB® programming language. This time, the problem consists of 10 objective functions, 115 decision variables, 1 inequality constraint, and 99 equality constraints. All the variables are bounded below and above. To solve this problem, 30 independent runs of EMyO-C are performed with a population size of $\mu = 300$, running for 750 generations. Further parameter settings are the same as in the previous experiment. In each run, the feasible solution is introduced in the initial population, whereas the remaining individuals are randomly generated.

Experimental Results

The clustering procedure is used to group all nondominated solutions in the objective space. Table 10.2 presents the most important variables and the objective function values for a representative of each cluster. Additionally, the values of the total cost (TC) and the quality index (QI) are calculated for these solutions. From the table, it can seen that the values of the aeration tank volume, the sedimentation area as well as the nitrate and nitrite nitrogen are constant for all the presented solutions. The values of the chemical oxygen demand are slightly reduced compared with that obtained using the multiobjective

V_a	G_S	A_s	h	N	COD	TSS	BOD	TKN	S_{NO}	Q_{ef}	TC(M €)	QI
100	6594.3023	271.7075	1	2.8161	124.2268	11.8656	61.1615	2.8064	0.0802	374.6088	3.9633	122.8764
100	6594.3023	271.7075	1	2.8171	124.1447	19.2583	61.0899	2.8064	0.0802	374.6306	3.9633	128.3383
100	6594.3023	271.7075	1	2.8453	124.3486	29.1979	57.9506	2.8064	0.0802	373.5855	3.9633	133.1374
100	6594.3023	271.7075	1	2.9506	124.2268	11.3468	61.2184	2.8064	0.0802	373.7916	3.9633	122.2631
100	6599.363	271.7075	1	3.2074	123.4647	34.0113	60.1686	3.2387	0.0802	373.2369	3.9653	141.1593
100	6589.2416	271.7075	1.3647	2.1974	125	19.7987	60.7109	1.9417	0.0802	373.4868	3.9678	121.9279
100	6594.3023	271.7075	1.3849	3.2028	123.374	25.6511	61.0103	3.2387	0.0802	374.5473	3.9702	135.9887
100	6599.363	271.7075	1.3036	2.4526	124.136	15.4315	60.4597	2.374	0.0802	373.1675	3.9707	121.2809
100	6599.363	271.7075	1.4351	3.2256	122.8556	34.5301	61.0103	3.2387	0.0802	374.0567	3.9731	142.2592
100	6594.3023	271.7075	1.5554	2.958	123.6176	15.4219	61.1504	2.8064	0.0802	373.6879	3.9733	124.9963
100	6599.363	271.7075	1.5061	2.106	124.5326	34.5059	60.7304	1.9417	0.0802	373.3003	3.9744	132.6874
100	6594.3023	271.7075	1.6622	2.1492	124.5326	12.1456	59.9099	2.374	0.0802	374.1559	3.9753	118.8802
100	6599.363	271.7075	1.5554	2.958	122.8556	30.1749	61.1504	2.8064	0.0802	374.2245	3.9753	135.9325
100	6599.363	271.7075	1.5558	2.4022	125	16.2075	60.6728	2.374	0.0802	373.0643	3.9753	122.3076
100	6594.3023	271.7075	1.6879	3.0518	123.6599	19.0348	60.8003	2.8064	0.0802	377.0701	3.9757	128.6042
100	6599.363	271.7075	1.6298	2.8089	122.904	32.9028	61.0551	2.8064	0.0802	373.848	3.9767	137.7822
100	6599.363	271.7075	1.6298	2.958	123.374	31.3109	60.5197	2.8064	0.0802	373.3762	3.9767	136.1953
100	6599.363	271.7075	1.6879	2.9973	123.9122	19.0348	60.8003	2.8064	0.0802	373.27	3.9777	127.4023
100	6589.2416	271.7075	1.9295	3.2703	124.6942	19.2583	61.6308	2.8064	0.0802	373.3365	3.9782	128.504
100	6594.3023	271.7075	1.927	2.8689	124.3001	13.7515	60.7982	3.2387	0.0802	373.5766	3.9802	126.933

Table 10.2: Optimal values for the most important variables obtained using many-objective approach.

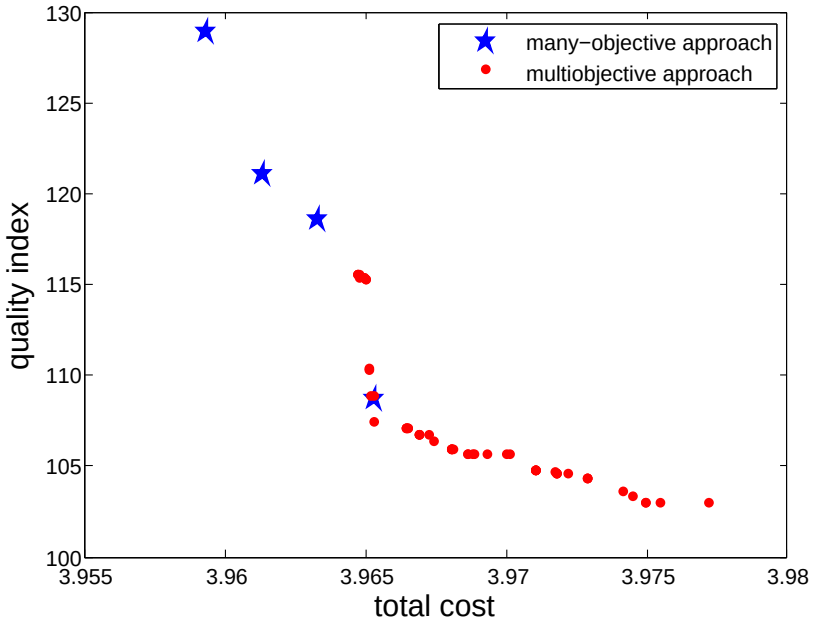


Figure 10.6: Trade-off curves obtained using multiobjective and many-objective approaches.

approach. On the other hand, the values of the total nitrogen and the total suspended solids are higher than in the previous experiments. However, they are still below the law limits. Relatively small variation can be observed regarding the other variables, namely, the biochemical oxygen demand, the total nitrogen of Kjeldahl and the effluent flow.

Furthermore, the values of the total cost and the quality index for all obtained nondominated solutions are calculated. This way, more general approach, which only consists in minimizing the influential variables, is projected to the particular case of a WWTP facility. As a result, only four nondominated solutions were obtained in the space defined by TC and QI. These solutions along with all nondominated solutions obtained using multiobjective approach are shown in Figure 10.6. It is interesting to note that the range of the trade-off curve obtained in the previous experiments was extended. Solution having the smallest cost function value was obtained using the many-objective approach (Figure 10.6). This can be explained by the fact that in a high dimensional objective space the number of nondominated solutions is usually lager than in smaller dimensional spaces. This effect results

in the propagation of the dissimilarities between solutions in the decision space. Typically, this is an undesirable feature that adversely affects the performance of EMO algorithms. However, the advantageous side effect is that different regions of the search space can be explored, which happens to be the case in the herein described experiments. Thus, the obtained results show that the many-objective approach can be useful not only during a draft project when the exact data about design of a particular WWTP is not available, but also for finding optimal values in the later stages of the decision-making process. Defining and simultaneously optimizing different combinations of objectives can enrich the final set of the obtained alternatives, which are eventually presented to the decision maker.

10.5 Summary

This chapter addresses new methodologies to model and solve a WWTP design optimization problem that can be extended to any WWTP unit operation modelling, regardless adjusting to each particularity of the problem under study. Two different approaches are considered. In the first approach, the WWTP design is addressed through a biobjective optimization problem that consists in minimizing the total cost and the quality index functions. To estimate some model parameters, the real portuguese data were used. Since the least squares technique needs to be used for parameter estimation, the formulated problem is time and local dependent. The second approach is more suitable for a draft project, when the exact location and time where the WWTP is going to be built is still unknown. The approach consists in simultaneously minimizing influential variables, which results in a ten-objective optimization problem. After solving this problem, the decision-maker has a set of alternatives from which he(she) can choose from. This information might help to elaborate a first version of the project, allowing to study all the alternatives, even with different unit operations. When the specific location and moment in time are defined, the analysis based on the minimization of the two objective functions – the total cost and the quality index – is to be preferred.

The results obtained in this study clearly show that the multiobjective modelling is

an effective tool to the WWTP design optimization. The achieved optimal solutions are meaningful in physical terms. Investment and operation costs are highly influenced by the optimized variables, meaning that the obtained solutions are economically attractive. Both approaches to WWTP design optimization provide a set of nondominated solutions from which the decision-maker can choose according to his(her) preferences.

Chapter 11

Conclusions

11.1 Conclusions

Optimization problems involving multiple conflicting objectives are common in the real world. These problems, called multiobjective optimization problems, contain multiple conflicting objectives, giving rise to a set of optimal solutions instead of a single optimal solution. This set is generally known as the Pareto optimal set. The existence of two search spaces and multiple optimal solutions constitutes the fundamental difference between a single-objective and multiobjective optimization, making in general the latter more difficult than the former. Recently, due to their population-based nature evolutionary algorithms have become a powerful and increasingly popular technique to approximate the Pareto set.

The present thesis investigates the application of EMO algorithms to solve multiobjective optimization problems from different perspectives. As new approaches based on combinations of existing techniques have proved to be successful in single-objective optimization, the bulk of the thesis is dedicated to develop hybrid multiobjective evolutionary algorithms. Such methods are devised to exploit better features of their single components, increasing the diversity of the existing approaches, thereby representing a significant contribution in the optimization research area. Since the decision maker is prone to put every performance index of the underlying problem as a separate objective, optimization prob-

lems with a large number of objectives arise continuously in diverse fields of science and engineering, leading to an increasing demand for efficient approaches capable to handle such problems. Therefore, a significant effort is spent enforcing the field of many-objective optimization. As a consequence, two highly competitive selection methods for evolutionary many-objective optimization are proposed. Since the demand for solving real-world problems is the main motivation to develop new optimization techniques, the herein developed algorithms are applied to solve two real-world optimization problems arising from epidemiological and environmental mathematical models, thereby making the contributions of the thesis relevant in terms of practical applications.

The review of the existing approaches to multiobjective optimization is presented in Chapter 3. Although different approaches are discussed, the review clearly indicates the popularity of EMO algorithms. Nevertheless, since their emergence in the mid 1980s and a wide variety of proposed algorithmic frameworks, almost all existing EMO algorithms are created by extending single-objective EAs. This is often made by modifying selection operators of EAs, meanwhile maintaining the same variation operators. There is a relatively limited number of hybrid MOEAs, especially, based on combinations of traditional evolutionary algorithms and local search methods.

The first attempt to develop a hybrid algorithm for MO is presented in Chapter 5. The resulting HGPSAL approach uses genetic operators for global search and a pattern search method to improve solutions, within an augmented Lagrangian framework for constrained single-objective optimization. Despite the promising results obtained on SOPs, the extension of this approach to MO adopting frameworks of the NBI and NC methods possesses several disadvantages when compared with EMO due to a relatively large number of function evaluations required for the optimization process.

Further, exploring the direction of hybrid algorithms for MO, two local search-based approaches are proposed in subsequent chapters. Adopting the idea of Timmel's classical method for generating new candidate solutions, DDMOA is introduced in Chapter 6. The potential strengths of this approach are revealed in comparative studies with some state-of-the-art EMO algorithms. The proposed procedure to explore the search space appears

to be highly competitive with different established MOEAs. Employing descent directions found for each objective to generate offspring makes the reproduction operator of DDMOA intrinsically multiobjective. It is devised taking into consideration the fundamental nature of MOPs, i.e., the presence of multiple objectives. On the contrary, most of the existing EMO algorithms just employ variation operators initially designed for single-objective optimization.

Motivated by promising results and some identified potential weaknesses of DDMOA, further research has been conducted in the direction paved by this approach. This results in the successor of DDMOA, termed DDMOA2, which generalizes the original algorithm, and is presented in Chapter 7. Although DDMOA2 mainly inherits the way offspring are generated from its predecessor, significant improvements of the performance are achieved. Moreover, due to the use of scalarizing fitness assignment DDMOA2 is applicable to solve problems with more than three-objectives. Although DDMOA2 represents a viable alternative for solving MOPs, exhibiting a highly competitive and often superior performance with respect to the existing approaches, the deterioration of performance is observed when the number of objectives increases.

Chapter 8 is particularly devoted to many-objective optimization. Adopting the framework of DE, slight modifications are introduced to improve its convergence properties on multimodal problems. The resulting differential evolution with variable-wise mutation restriction is used as the basis for creating efficient many-objective optimization algorithms. Two different mechanisms to provide necessary selection pressure in a high-dimensional objective space are developed. The indicator-based approach uses a set of evenly distributed points on the hyperplane to approximate the Pareto front to calculate distance metrics based on IGD for solutions in the last accepted non-domination level. This approach exhibits a good performance being capable to converge to the Pareto set for all the tested problems. The second selection scheme is created as an attempt to obtain a more flexible and self-adaptive approach for many-objective optimization. Although it shows ability to guide the population toward the Pareto front in a high-dimensional objective space, the results on some problems reveal that more computational effort may be eventually required

to obtain more precise approximations. However, results obtained on DTLZ2 and DTLZ7 clearly highlight the advantages of the suggested clustering-based selection scheme. Since this approach does not require any additional parameters and is polynomial in the number of objectives, it points a promising direction for future research. Similarly to optimization algorithms in general, where it is not possible to obtain one solver that performs the best on all problems, it seems to be difficult, if not impossible, to have the best fitness assignment and selection scheme to guide the population in the objective space.

The third part of the thesis is concerned with solving real-world problems using EMO algorithms. The first problem arises from a mathematical model of the dengue disease transmission, where one seeks to identify the most effective way of applying insecticide minimizing two conflicting objectives: the total cost due to infected humans and the total cost of applying insecticide. Additionally, the performance comparison of different EMO algorithms on this problem is carried out. The obtained results reveal that all of the used state-of-the-art MOEAs perform poorly, failing to provide a wide range of trade-off solutions within the given budget. On the other hand, DDMOA2 takes advantage of its embedded local search procedure that helps to extensively search the high-dimensional decision space. Even the visual comparison of the trade-off solutions obtained by different algorithms clearly shows the superiority of DDMOA2 when solving this problem.

Another considered problem arises from optimizing a wastewater treatment plant design. To solve this problem, different methodologies for finding optimal values of the decision variables are addressed. Although one can define two essential goals in the WWTP design, namely, reducing the total cost and maximizing the WWTP performance, the ways for achieving these goals can be different. Thus, a multiobjective approach, based on a biobjective problem, appears to be an effective way for finding optimal values in the WWTP design. The obtained set of trade-off solutions gives much more possible alternatives than single-objective approach does. The wide range of trade-off solutions provides valuable information about the problem, being highly beneficial to the decision-making process. Aside from the two-objective approach, which is time and local dependent, a more general approach is considered. Thus, each influential variable is minimized sepa-

rately. Although this approach is mainly conceived for a draft project, the obtained results reveal that defining and simultaneously optimizing different objectives can be beneficial at any stage of the decision-making process. In turn, this opens new research opportunities in the WWTP design area.

The performance analysis of different algorithms on real-world problems is particularly important. It can reveal strengths and weaknesses of different methodologies, which in the design phase are often tested only on artificial test functions. Such problems are undoubtedly useful, however, they do not possess all the difficulties encountered in real-world optimization problems. Therefore, careful analysis of algorithms' performance and inherent difficulties of real problems constitute a significant contribution to the research. In turn, the promising results obtained by DDMOA2 on the dengue disease transmission model highlight the relevance of hybrid approaches to multiobjective optimization. Actually, they motivate further research in this direction.

11.2 Future Perspectives

Despite the significant advances achieved in the design of EMO algorithms during the last two decades, and a growing attention to the field of EMO that resulted in a large number of publications, there are a lot of open issues and opportunities for future research. Thus, the popular algorithmic frameworks need to be investigated thoroughly to further define their strengths and weaknesses. Such studies on real-world problems are of particular interest. As a result, some new frameworks or combinations of the existing frameworks may arise in the near future. In turn, these new proposals will also rise other new research questions and opportunities.

Since there are two search spaces in which EAs operate when solving multiobjective optimization problems, new advanced operators for performing the search in each space must be designed. They are essential to successfully deal with increasingly complex real-world problems. Although the current research in EMO very often focuses on either the managing of the external archive or developing an appropriate selection strategy to guide the

search, new fitness assignment and selection procedures capable to maintain the diversity among the population members and to assure the convergence to the Pareto set in difficult problem environments must be investigated. This is especially relevant to problems with a large number of objectives.

Moreover, there is a continuous demand for efficient variation operators, which are crucial to the performance of either single-objective or multiobjective optimization algorithms. A substantial amount of work must still be done to successfully handle multimodal, ill-conditioned, nonseparable, deceptive problems. Although there exists a variety of EAs with different offspring generation strategies that are successful in single-objective optimization, their extensions to the multiobjective case may correspond to a poor performance. However, the current research in evolutionary computation community is mainly based on using the same variation operators in single-objective and multiobjective optimization. Furthermore, hybridization of evolutionary algorithms with local search methods in multiobjective optimization is another promising research direction, which until now seems to receive relatively small attention.

Another important issue that highly affects the performance of EMO algorithms is how to properly choose several control parameters for the algorithms. Usually, the parameters are predefined based on algorithmic or problem knowledge. They also can be determined as a result of an empirical study, where a set of suitable parameters is chosen after performing repeated tests with different settings. However, in real-world applications, especially with computationally expensive function evaluations, this strategy may not be applicable. Thus, new strategies to adaptively tune control parameters need to be investigated. This idea can also be applied to search strategies either in the decision or objective space, where the best suitable among available strategies is chosen in different phases of the search process.

Bibliography

[1] H. A. Abbass, R. Sarker, and C. Newton. A Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'01, pages 971–978, 2001.

[2] J. Alex, L. Benedetti, J. Copp, K. V. Gernaey, U. Jeppsson, I. Nopens, M. N. Pons, C. Rosen, J. P. Steyer, and P. Vanrolleghem. Benchmark simulation model no. 1 (BSM1). Technical report, IWA Taskgroup on Benchmarking of Control Strategies for WWTPs, 2008.

[3] Md. Asafuddoula, T. Ray, and R. Sarker. A decomposition based evolutionary algorithm for many objective optimization with systematic sampling and adaptive epsilon control. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'13, pages 413–427, 2013.

[4] T. Bäck, U. Hammel, and H. P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[5] J. Bader. *Hypervolume-based search for multiobjective optimization: Theory and methods*. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 2010.

[6] J. Bader, K. Deb, and E. Zitzler. Faster hypervolume-based search using Monte Carlo sampling. In *Proceedings of the Conference on Multiple Criteria Decision Making*, MCDM'08, pages 313–326, 2008.

[7] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[8] S. Bandyopadhyay, U. Maulik, and M. K. Pakhira. Clustering using simulated annealing with probabilistic redistribution. *Journal of Pattern Recognition and Artificial Intelligence*, 15(2):269–285, 2001.

[9] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283, 2008.

[10] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. Part I: background and development. *Natural Computing*, 6(4):467–484, 2007.

[11] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1):109–124, 2008.

[12] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, 2 edition, 1999.

[13] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[14] H.-G. Beyer. *The theory of evolution strategies*. Springer-Verlag, New York, USA, 2001.

[15] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[16] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA: A platform and programming language independent interface for search algorithms. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'03, pages 494–508, 2003.

[17] P. A. N. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):174–188, 2003.

[18] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1198–1203, 2011.

[19] D. Brockhoff and E. Zitzler. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Proceedings of the Conference on Parallel Problem Solving from Nature*, PPSN'06, pages 533–542, 2006.

[20] D. Brockhoff and E. Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'07, pages 2086–2093, 2007.

[21] D. Brockhoff and E. Zitzler. Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary Computation*, 17(2):135–166, 2009.

[22] Hydromantis Inc. Canada. GPS-X user's guide, 2001.

[23] Hydromantis Inc. Canada. Gps-x v4.1. http://www.hydromantis.com/software02.html, 2002.

[24] E. G. Carrano, E. F. Wanner, and R. H. C. Takahashi. A multicriteria statistical based comparison methodology for evaluating evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 15(6):848–870, 2011.

[25] P. Cattand, P. Desjeux, M. G. Guzmán, J. Jannin, A. Kroeger, A. Medici, P. Musgrove, M. B. Nathan, A. Shaw, and C. J. Schofield. *Tropical Diseases Lacking Adequate Control Measures: Dengue, Leishmaniasis, and African Trypanosomiasis*. Control Priorities in Developing Countries. DCPP, 2 edition, 2006.

[26] R. Caves, S. Quegan, and R. White. Quantitative comparison of the performance of SAR segmentation algorithms. *Transactions on Image Processing*, 7(11):1534–1546, 1998.

[27] C. Chen, Y. Chen, and Q. Zhang. Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'09, pages 209–216, 2009.

[28] T. Chiang and Y. Lai. MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'11, pages 1473–1480, 2011.

[29] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer, 2 edition, 2007.

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

[31] A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[32] D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'07, pages 773–780, 2007.

[33] L. Costa, I. Espírito Santo, and P. Oliveira. Stochastic algorithms assessment using performance profiles. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'11, pages 933–940, 2011.

[34] L. Costa and P. Oliveira. An adaptive sharing elitist evolution strategy for multiobjective optimization. *Evolutionary Computation*, 11(4):417–438, 2003.

[35] L. Costa, I. A. C. P. Espírito Santo, R. Denysiuk, and E. M. G. P. Fernandes. Hybridization of a genetic algorithm with a pattern search augmented Lagrangian method. In *Proceedings of the Conference on Engineering Optimization*, EngOpt'10, page 1195, 2010.

[36] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140, 2011.

[37] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998.

[38] N. O. Da Cunha and E. Polak. Constrained minimization under vector-valued criteria in finite dimensional spaces. *Journal of Mathematical Analysis and Applications*, 9(1):103–124, 1967.

[39] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[40] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

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

[42] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, 2001.

[43] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[44] K. Deb and H. Jain. Handling many-objective problems using an improved NSGA-II procedure. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'12, pages 1–8, 2012.

[45] K. Deb, K. Miettinen, and S. Chaudhuri. Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation*, 14(6):821–841, 2010.

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

[47] K. Deb and D. K. Saxena. Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the World Congress on Computational Intelligence*, WCCI'06, pages 3352–3360, 2006.

[48] K. Deb, A. Sinha, and S. Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. Technical Report 2006001, Indian Institute of Technology, Kanpur, India, 2006.

[49] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'06, pages 635–642, 2006.

[50] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Swiss Federal Institute of Technology, Zurich, Switzerland, 2001.

[51] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'02, pages 825–830, 2002.

[52] R. Denysiuk, L. Costa, and I. Espírito Santo. DDMOA: Descent directions based multiobjective algorithm. In *Proceedings of the Conference on Computational and Mathematical Methods in Science and Engineering*, CMMSE'12, pages 460–471, 2012.

[53] R. Denysiuk, L. Costa, and I. Espírito Santo. DDMOA2: Improved descent directions-based multiobjective algorithm. In *Proceedings of the Conference on Computational and Mathematical Methods in Science and Engineering*, CMMSE'13, pages 513–524, 2013.

[54] R. Denysiuk, L. Costa, and I. Espírito Santo. Many-objective optimization using differential evolution with variable-wise mutation restriction. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'13, pages 591–598, 2013.

[55] R. Denysiuk, L. Costa, and I. Espírito Santo. A new hybrid evolutionary multiobjective algorithm guided by descent directions. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(3):233–251, 2013.

[56] M. Derouich, A. Boutayeb, and E. Twizell. A model of dengue fever. *Biomedical Engineering Online*, 2(4):1–10, 2003.

[57] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[58] G. J. Devine, E. Z. Perea, G. F. Killen, J. D. Stancil, S. J. Clark, and A. C. Morrison. Using adult mosquitoes to transfer insecticides to aedes aegypti larval habitats. In

Proceeding of the National Academy of Sciences of the United States of America, 2009.

[59] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[60] N. Drechsler, R. Drechsler, and B. Becker. Multi-objective optimisation based on relation favour. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'01, pages 154–166, 2001.

[61] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba. Multi-objective particle swarm optimizers: An experimental comparison. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'09, pages 495–509, 2009.

[62] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[63] G. A. Ekama, J. L. Barnard, F. W. Günthert, P. Krebs, J. A. McCrquodale D. S. Parker, and E. J. Wahlberg. Secondary settling tanks: Theory, modelling, design and operation. Technical Report 6, IAWQ - International Association on Water Quality, 1997.

[64] M. Emmerich, N. Beume, and N. Boris. An EMO algorithm using the hypervolume measure as selection criterion. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 62–76, 2005.

[65] M. T. M. Emmerich and A. H. Deutz. Test problems based on Lamé superspheres. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 922–936, 2007.

[66] L. Esteva and H. M. Yang. Mathematical model to assess the control of aedes aegypti mosquitoes by the sterile insect technique. *Mathematical Biosciences*, 198:132–147, 2005.

[67] M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8(5):425–442, 2004.

[68] M. Fleischer. The measure of Pareto optima applications to multi-objective metaheuristics. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'03, pages 519–533, 2003.

[69] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-objective optimization: An engineering design perspective. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 14–32, 2005.

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

[71] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *Proceedings of the Conference on Parallel Problem Solving from Nature*, PPSN'96, pages 584–593, 1996.

[72] C. M. M. Fonseca. *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. PhD thesis, University of Sheffield, UK, 1995.

[73] X.-S. Qin G.-H. Huang G.-M. Zeng, S.-F. Zhang and J.-B. Li. Application of numerical simulation on optimum design of two-dimensional sedimentation tanks in the wastewater treatment plant. *Journal of Environmental Sciences*, 15(3):346–350, 2003.

[74] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009.

[75] S. Gass and T. Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2(1):39–45, 1955.

[76] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.

[77] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, USA, 1989.

[78] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, 1(3):296–297, 1971.

[79] M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, Copenhagen, Denmark, 1998.

[80] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the IEEE Conference on Evolutionary Computation*, CEC'96, pages 312–317, 1996.

[81] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[82] M. Hapke, A. Jaszkiewicz, and R. Slowinski. Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. *Journal of Heuristics*, 6(3):329–345, 2000.

[83] M. Henze, C. P. L. Grady Jr, W. Gujer, G. V. R. Marais, and T. Matsuo. Activated sludge model no. 1. Technical report, IAWPRC Task Group on Mathematical Modelling for design and operation of biological wastewater treatment, 1986.

[84] Y. C. Ho and D. L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115(3):549–570, 2002.

[85] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Ann Arbor, MI, 1975.

[86] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal on Associated Computation*, 8(2):212–229, 1961.

[87] S. Huband, P. Hingston, L. Barone, and L. While. A scalable multi-objective test problem toolkit. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 280–295, 2005.

[88] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[89] S. Huband, P. Hingston, L. White, and L. Barone. An evolution strategy with probabilistic mutation for multi-objective optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'03, pages 2284–2291, 2003.

[90] E. J. Hughes. Multiple single objective Pareto sampling. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'03, pages 2678–2684, 2003.

[91] E. J. Hughes. Evolutionary many-objective optimisation: Many once or one many? In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'05, pages 222–227, 2005.

[92] E. J. Hughes. MSOPS-II: A general-purpose many-objective optimiser. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'07, pages 3944–3951, 2007.

[93] E. J. Hughes. Many-objective directed evolutionary line search. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'11, pages 761–768, 2011.

[94] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[95] C. Igel, T. Suttorp, and N. Hansen. Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, 2007.

[96] A. W. Iorio and X. Li. Solving rotated multi-objective optimization problems using differential evolution. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pages 861–872, 2004.

[97] H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. In *Proceedings of the Conference on Parallel Problem Solving from Nature*, PPSN'06, pages 493–502, 2006.

[98] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 28(3):392–403, 1998.

[99] H. Ishibuchi and Y. Nojima. Optimization of scalarizing functions through evolutionary multiobjective optimization. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 51–65, 2007.

[100] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima. Simultaneous use of different scalarizing functions in MOEA/D. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'10, pages 519–526, 2010.

[101] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Iterative approach to indicator-based multiobjective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3967–3974, 2007.

[102] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'08, 2008.

[103] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.

[104] H. Jain and K. Deb. An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'13, pages 307–321, 2013.

[105] A. Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[106] V. J. Bowman Jr. On the relationship of the Chebyshev norm and the efficient frontier of multiple criteria objectives. *Lecture Notes in Economics and Mathematical Systems*, 130:76–86, 1976.

[107] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[108] V. R. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'03, pages 376–390, 2003.

[109] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[110] J. Knowles and D. Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[111] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland, 2006.

[112] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, UK, 2002.

[113] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[114] M. Köppen and K. Yoshida. Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 727–741, 2007.

[115] S. Kukkonen and J. Lampinen. An extension of generalized differential evolution for multi-objective optimization with constraints. In *Proceedings of the Conference on Parallel Problem Solving from Nature*, PPSN'04, pages 752–761, 2004.

[116] S. Kukkonen and J. Lampinen. GDE3: the third evolution step of generalized differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'05, pages 443–450, 2005.

[117] S. Kukkonen and J. Lampinen. Ranking-dominance and many-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'07, pages 3983–3990, 2007.

[118] M. Laguna and R. Marti. *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[119] J. Lampinen. DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Lappeenranta, Finland, 2001.

[120] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[121] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[122] R. M. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.

[123] H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

[124] H. Li, Q. Zhang, E. Tsang, and J. A. Ford. Hybrid estimation of distribution algorithm for multiobjective knapsack problem. In *Proceedings of the Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'04, pages 145–154, 2004.

[125] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical report, School of EEE, Nanyang Technological University, Singapore, 2006.

[126] W. L. Loh. On latin hypercube sampling. *Annals of Statistics*, 33(6):2058–2080, 1996.

[127] A. López Jaimes, C. A. Coello Coello, and D. Chakraborty. Objective reduction using a feature selection technique. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'08, pages 673–680, 2008.

[128] I. Loshchilov, M. Schoenauer, and M. Sebag. Not all parents are equal for MO-CMA-ES. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'11, pages 31–45, 2011.

[129] W. K. Mashwani and A. Salhi. A decomposition-based hybrid multiobjective evolutionary algorithm with dynamic resource allocation. *Applied Soft Computing*, 12(9):2765–2780, 2012.

[130] A. Messac and C. Mattson. Normal constraint method with guarantee of even representation of complete Pareto frontier. *AIAA Journal*, 42:2101–2111, 2004.

[131] E. Mezura-Montes, M. Reyes-Sierra, and C. A. Coello Coello. Multi-objective optimization using differential evolution: A survey of the state-of-the-art. *Advances in Differential Evolution*, 143:173–196, 2008.

[132] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1992.

[133] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, 1999.

[134] M. Mitchell. *Introduction to Genetic Algorithms*. MIT Press, Ann Arbor, MI, 1996.

[135] A. J. Nebro, J. J. Durillo, J. García-Nieto, C. A. Coello Coello, F. Luna, and E. Alba. SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In *Proceedings of the Symposium on Computational Intelligence in Multicriteria Decision-Making*, MCDM'09, pages 66–73, 2009.

[136] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 12(4), 2008.

[137] F. Neri and V. Tirronen. Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2):61–106, 2010.

[138] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *Proceedings of the Conference on Parallel Problem Solving From Nature*, PPSN'04, pages 792–802, 2004.

[139] M. Otero, N. Schweigmann, and H. G. Solari. A stochastic spatial dynamical model for aedes aegypti. *Bulletin of Mathematical Biology*, 70(5):1297–1325, 2008.

[140] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1), 2002.

[141] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.

[142] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimisation: An exploratory analysis. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'03, pages 2066–2073, 2003.

[143] S. Zhao P. N. Suganthan W. Liu S. Tiwari Q. Zhang, A. Zhou. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, University of Essex, UK, 2009.

[144] S. Rana, S. Jasola, and R. Kumar. A review on particle swarm optimization algorithms and their applications to data clustering. *Artificial Intelligence Review*, 35(3), 2011.

[145] M. Reyes Sierra and C. A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 505–519, 2005.

[146] M. Reyes Sierra and C. A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[147] H. S. Rodrigues. *Optimal control and numerical optimization applied to epidemiological models*. PhD thesis, University of Aveiro, Aveiro, Portugal, 2012.

[148] C. A. Rodríguez Villalobos and C. A. Coello Coello. A new multi-objective evolutionary algorithm based on a performance assessment indicator. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'12, pages 505–512, 2012.

[149] L. V. Santana-Quintero and C. A. Coello Coello. An algorithm based on differential evolution for multi-objective problems. *International Journal of Computational Intelligence Research*, 1(2):151–169, 2005.

[150] I. A. C. P. Espírito Santo, L. Costa, R. Denysiuk, and E. M. G. P. Fernandes. Hybrid genetic pattern search augmented Lagrangian algorithm: Application to WWTP optimization. In *Proceedings of the Conference on Applied Operational Research*, ICAOR'10, pages 45–56, 2010.

[151] I. A. C. P. Espírito Santo, E. M. G. P. Fernandes, M. M. Araújo, and E. C. Ferreira. On the secondary settler models robustness by simulation. *WSEAS Transactions on Information Science and Applications*, 3:2323–2330, 2006.

[152] H. Sato, H. E. Aguirre, and K. Tanaka. Controlling dominance area of solutions and its impact on the performance of MOEAs. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 5–20, 2007.

[153] D. K. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: employing correntropy and a novel maximum variance unfolding. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 772–787, 2007.

[154] D. K. Saxena, Q. Zhang, J. Duro, and A. Tiwari. Framework for many-objective test problems with both simple and complicated Pareto-set shapes. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'11, pages 197–211, 2011.

[155] D. K. Saxena, Q. Zhang, J. A. Duro, and A. Tiwari. Framework for many-objective test problems with both simple and complicated Pareto-set shapes. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'11, pages 197–211, 2011.

[156] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the Conference on Genetic Algorithms*, pages 93–100, 1985.

[157] J. R. Schott. Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1995.

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

[159] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, USA, 1995.

[160] P. Serafini. Simulated annealing for multiobjective optimization problems. In *Proceedings of the Conference on Multiple Criteria Decision Making*, pages 87–96, 1992.

[161] P. K. Shukla and K. Deb. On finding multiple Pareto-optimal solutions using classical and evolutionary generating methods. *European Journal of Operational Research*, 181(3):1630–1652, 2007.

[162] H. K. Singh, A. Isaacs, and T. Ray. A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 15(4):539–556, 2011.

[163] K. I. Smith, R. M. Everson, and J. E. Fieldsend. Dominance measures for multi-objective simulated annealing. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'04, pages 23–30, 2004.

[164] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[165] R. Storn. Differential evolution research: Trends and open questions. *Advances in Differential Evolution*, 143:1–31, 2008.

[166] R. Storn and K. Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[167] R. Storn and K. V. Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, USA, 1995.

[168] A. Sülflow, N. Drechsler, and R. Drechsler. Robust multi-objective optimization in high dimensional spaces. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'06, pages 715–726, 2006.

[169] B. Suman and P. Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57(18):1143–1160, 2006.

[170] A. Suppapitnarm, K. A. Seffen, G. T. Parks, and P. J. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1):59–85, 2000.

[171] T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 75(2):167–197, 2009.

[172] I. Takács, G. G. Patry, and D. Nolasco. A dynamic model of the clarification-thickening process. *Water Research*, 25(10):1263–1271, 1991.

[173] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436, 2009.

[174] G. Timmel. Ein stochastisches suchverrahren zur bestimmung der optimalen kompromilsungen bei statischen polzkriteriellen optimierungsaufgaben. *Wiss. Z. TH Ilmenau*, 26(5):159–174, 1980.

[175] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7:1–25, 1997.

[176] D. Tyteca. Cost functions for wastewater conveyance systems. *Water Pollution Control Federation*, 48(9):2120–2130, 1976.

[177] K. Deb E. Zitzler P. N. Suganthan J. J. Liang M. Preuss S. Huband V. L. Huang, A. K. Qin. Problem definitions for performance assessment on multi-objective optimization algorithms. Technical report, Nanyang Technological University, Singapore, 2007.

[178] J. A. Vasconcelos, J. H. R. D. Maciel, and R. O. Parreiras. Scatter search techniques applied to electromagnetic problems. *Transactions on Magnetics*, 41(5):1804–1807, 2005.

[179] D. A. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1999.

[180] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1998.

[181] T. Voß, N. Beume, G. Rudolph, and C. Igel. Scalarization versus indicator-based selection in multi-objective CMA evolution strategies. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'08, pages 3036–3043, 2008.

[182] T. Voß, N. Hansen, and C. Igel. Recombination for learning strategy parameters in the MO-CMA-ES. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'09, pages 155–168, 2009.

[183] T. Voß, N. Hansen, and C. Igel. Improved step size adaptation for the MO-CMA-ES. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, GECCO'10, pages 487–494, 2010.

[184] T. Wagner, N. Beume, and B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 742–756, 2007.

[185] S. Wang. Second-order necessary and sufficient conditions in multiobjective programming. *Numerical Functional Analysis and Application*, 12(1–2):237–252, 1991.

[186] WHO. Dengue. http://www.who.int/topics/dengue/en/, January 2012.

[187] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[188] J. Wu and S. Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 2001.

[189] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[190] Q. Zhang, W. Liu, and H. Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'09, pages 203–208, 2009.

[191] Q. Zhang, A. Zhou, and Y. Jin. RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.

[192] Q. Zhang, A. Zhou, S. Z. Zhao, P. N. Suganthan, and S. Tiwari W. Liu. Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-887, University of Essex and Nanyang Technological University, 2008.

[193] A. Zhou, Q. Zhang, and Y. Jin. Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 13(5):1167–1189, 2009.

[194] A. Zhou, Q. Zhang, and G. Zhang. A multiobjective evolutionary algorithm based on decomposition and probability model. In *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC'12, pages 1–8, 2012.

[195] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 1999.

[196] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, pages 862–876, 2007.

[197] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[198] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proceedings of the Conference on Parallel Problem Solving from Nature*, PPSN'04, pages 832–842, 2004.

[199] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for Multiobjective Optimization*, pages 3–38. Springer-Verlag, Berlin, Germany, 2004.

[200] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 2001.

[201] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1998.

[202] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - A comparative case study. In *Proceedings of the Conference on Parallel Problem Solving From Nature*, PPSN'98, pages 292–304, 1998.

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

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

Appendix

The symbols related to the WWTP modelling are the following:

A_s = sedimentation area, m^2

b_A = decay coefficient for autotrophic biomass, day^{-1}

b_H = decay coefficient for heterotrophic biomass, day^{-1}

BOD = biochemical oxygen demand, g O_2/m^3

BOD_U = ultimate BOD, g O_2/m^3

COD = chemical oxygen demand, g COD/m^3

$CSTR$ = completely stirred tank reactor

$DSVI$ = diluted sludge volume index

f_{BOD} = BOD/BOD_U ratio

f_{ns} = non-settleable fraction

f_P = fraction of biomass leading to particulate products

G_S = air flow rate, m^3/day at STP

h = settler tank depth, m

$HenryO_2$ = Henry constant

HRT = hydraulic retention time, day

i = discount rate

icv = X/VSS ratio, g COD/g VSS

i_{X_B} = nitrogen content of active biomass, g N/g COD

i_{X_P} = nitrogen content of endogenous/inert biomass, g N/g COD

IC = investment cost, €

ISS = inorganic suspended solids, g/m^3

J_{clar} = clarification flux above the feed layer, g/(day m^2)

J_s = sedimentation flux under the feed layer, g/(day m^2)

k_a = ammonification rate, m^3/(g COD day)

k_h = maximum specific hydrolysis rate, day^{-1}

K_La = overall mass transfer coefficient, day^{-1}

K_{NH} = ammonia half-saturation coefficient for autotrophic biomass growth, g N/m^3

K_{NO} = nitrate half saturation coefficient for denitrifying heterotrophic biomass, g N/m^3

K_{OA} = oxygen half-saturation coefficient for autotrophs growth, g O_2/m^3

K_{OH} = oxygen half-saturation coefficient for heterotrophs growth, g O_2/m^3

K_S = readily biodegradable substrate half-saturation coefficient for heterotrophic biomass, g COD/m^3

K_X = half-saturation coefficient for hydrolysis of slowly biodegradable substrate, g COD/g COD

$\mathcal{N}$ = life span of the treatment plant, years

N = total nitrogen, g N/m^3

OC = operation cost, €

P_c = power cost, €/kWh

P_{O_2} = partial pressure of oxygen uncorrected

PWWF = peak wet weather flow

Q = flow, m^3/day

QI = quality index, kg of pollution/day

r = recycle rate

r_h = hindered zone floating parameter, m^3/g TSS

r_p = flocculant zone settling parameter, m^3/g TSS

S = soluble COD, g COD/m^3

S_{alk} = alkalinity, molar units

S_I = soluble inert organic matter, g COD/m^3

S_{ND} = soluble biodegradable organic nitrogen, g N/m^3

S_{NH} = free and ionized ammonia, g N/m^3

S_{NO} = nitrate and nitrite nitrogen, g N/m^3

S_O = dissolved oxygen, g $(-COD)/\mathrm{m}^3$

$S_{O_{sat}}$ = saturated oxygen concentration, $\mathrm{g/m}^3$

S_S = readily biodegradable soluble substrate, g COD/m^3

SRT = sludge retention time, day

STP = standard temperature and pressure

TC = total cost, €

V_a = aeration tank volume, m^3

VSS = volatile suspended solids, $\mathrm{g/m}^3$

T = temperature, °C

TKN = total nitrogen of Kjeldahl, g N/m^3

TSS = total suspended solids, $\mathrm{g/m}^3$

TSS_t = threshold concentration of the sludge, $\mathrm{g/m}^3$

X = particulate COD, g COD/m^3

X_{BA} = active autotrophic biomass, g COD/m^3

X_{BH} = active heterotrophic biomass, g COD/m^3

X_I = particulate inert organic matter, g COD/m^3

X_{II} = inert inorganic suspended solids, $\mathrm{g/m}^3$

X_{ND} = particulate biodegradable organic nitrogen, g N/m^3

X_P = particulate products arising from biomass decay, g COD/m^3

X_S = slowly biodegradable substrate, g COD/m^3

Y_A = yield for autotrophic biomass, g COD/g N

Y_H = yield for heterotrophic biomass, g COD/g COD

Greek symbols

α = wastewater/clean water coefficient

β = salts and ions correction factor

η = standard oxygen transfer efficiency

η_g = correction factor for μ_H under anoxic conditions

η_h = correction factor for hydrolysis under anoxic conditions

μ_A = maximum specific growth rate for autotrophic biomass, day^{-1}

μ_H = maximum specific growth rate for heterotrophic biomass, day^{-1}

$\nu_{s,j}$ = settling velocity in layer j, m/day

ν_0 = maximum Vesilind settling velocity, m/day

ν_0' = security barrier to the maximum Vesilind settling velocity, m/day

ν_{up} = solids flux due to the descending bulk movement of the liquid, m/day

ν_{dn} = solids flux due to the ascending bulk movement of the liquid, m/day

ρ = density of water, kg/m^3

θ = temperature correction factor

ΔX = change in sludge concentration within the aeration tank

Subscripts

a = aeration tank

ef = effluent

in = entry of the aeration tank

inf = influent

p = during a PWWF event

r = recycle

s = settling tank

w = sludge waste

no index = inside the aeration tank=exit of the aeration tank