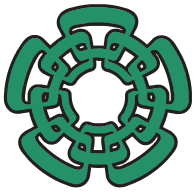


Center for Research and Advanced Studies
of the National Polytechnic Institute of Mexico

Zacatenco Campus
Computer Science Department

Use of Gradient-Free Mathematical Programming Techniques to Improve the Performance of Multi-Objective Evolutionary Algorithms

by

Saúl Zapotecas Martínez

as the fulfillment of the requirement for the degree of

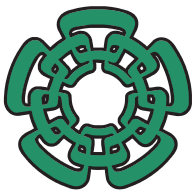
Ph.D. in Computer Science

Advisor:
Dr. Carlos A. Coello Coello

Mexico City June, 2013

Saúl Zapotecas Martínez: *Use of Gradient-Free Mathematical Programming Techniques to Improve the Performance of Multi-Objective Evolutionary Algorithms*. Ph.D. in Computer Science. © June, 2013.

ADVISOR: Dr. Carlos A. Coello Coello
LOCATION: Mexico City
DATE: June, 2013



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Computación

Uso de Técnicas de Programación Matemática que no requieren gradientes para Mejorar el Desempeño de Algoritmos Evolutivos Multi-Objetivo

Tesis que presenta

Saúl Zapotecas Martínez

Para Obtener el Grado de

Doctor en Ciencias en Computación

Director de Tesis:
Dr. Carlos A. Coello Coello

México, D.F. Junio, 2013

Saúl Zapotecas Martínez: *Uso de Técnicas de Programación Matemática que no requieren gradientes para Mejorar el Desempeño de Algoritmos Evolutivos Multi-Objetivo*. Doctor en Ciencias en Computación. © Junio, 2013.

DIRECTOR DE TESIS: Dr. Carlos A. Coello Coello
LUGAR: México, D.F.
FECHA: Junio, 2013

This thesis is dedicated to my parents and my sister.

Because family means nobody gets left behind, or forgotten.

To the memory of Pedro Zapotecas.

Abstract

In spite of the current widespread use of *Multi-Objective Evolutionary Algorithms (MOEAs)* for solving *Multi-objective Optimization Problems (MOPs)*, their computational cost (measured in terms of fitness function evaluations performed) remains as one of their main limitations when applied to real-world applications. In order to address this issue, a variety of hybrid approaches combining mathematical programming techniques with a MOEA have been proposed in the last few years. In this way, while the MOEA explores the whole search space, mathematical programming techniques exploit the promising regions given by the same MOEA. Most of these hybrid approaches rely on mathematical programming techniques based on gradients. Therefore, when the functions are not differentiable, these mathematical programming techniques become impractical, and then, other alternatives need to be explored, such as the direct search methods, i. e., mathematical programming methods that do not require gradient information.

In this thesis, we present different strategies to hybridize a popular direct search method (the *Nonlinear Simplex Search (NSS)* algorithm) with a MOEA. First, we present an extension of the NSS (which was originally introduced for single-objective optimization) for dealing with MOPs. The main goal of this study is to analyze and exploit the properties of the NSS algorithm when it is used to approximate solutions to the *Pareto optimal set* while maintaining a reasonably good representation of the *Pareto front*. Based on experimental evidence, we conclude that the NSS is a good alternative to be used as a local search engine into a MOEA. Then, we take the ideas proposed in the extension of the NSS for multi-objective optimization to be coupled as a local search engine into different MOEAs. This gave rise to different hybrid approaches which were validated using standard test problems and performance measures taken from the specialized literature.

Resumen

A pesar del actual uso extendido de los *Algoritmos Evolutivos Multi-objetivo (AEMOs)* para la resolución de *Problemas de Optimización Multi-objetivo (POMs)*, su costo computacional (medido en términos del número de evaluaciones de la función de aptitud) continúa siendo una de sus principales limitaciones cuando son utilizados en aplicaciones del mundo real. A fin de abordar este problema, una variedad de enfoques híbridos combinando técnicas de programación matemática con un AEMO han sido propuestos recientes años. De esta manera, mientras que el AEMO explora todo el espacio de búsqueda, las técnicas de programación matemática explotan las regiones prometedoras dadas por el mismo AEMO. La mayoría de estos enfoques híbridos dependen de técnicas de programación matemática basadas en gradientes. Por lo tanto, cuando las funciones no son diferenciables, estas técnicas llegan a ser poco prácticas y por tanto, deben explorarse otras alternativas, tales como los métodos de búsqueda directa, es decir, métodos de programación matemática que no requieren información del gradiente.

En esta tesis, se presentan diferentes estrategias para hibridizar un popular método de búsqueda directa (el algoritmo de la *búsqueda del simplex no lineal (NSS)*) con un AEMO. En primer lugar, presentamos una extensión del algoritmo NSS (que fue originalmente propuesto para optimización mono-objetivo) para lidiar con POMs. El objetivo principal de este estudio es analizar y explotar las propiedades del algoritmo NSS cuando es utilizado para aproximar soluciones al *conjunto de óptimos de Pareto* mientras se mantiene una buena representación del *frente de Pareto*. Con base en evidencia experimental, concluimos que el algoritmo NSS es una buena alternativa para ser utilizado como un motor de búsqueda local en un AEMO. Después, tomamos las ideas propuestas en la extensión del algoritmo NSS para optimización multi-objetivo para ser acoplado como un motor de búsqueda local en diferentes AEMOs. Esto dio pie a diferentes enfoques híbridos, los cuales fueron validados usando problemas de

prueba y medidas de desempeño estándar tomados de la literatura especializada.

Acknowledgments

First and foremost, I would like to thank sincerely my supervisor, Prof. Carlos A. Coello Coello, for his guidance and support throughout this thesis. Thanks for his patience and dedication in reviewing the papers that I developed throughout this research work. Thanks for showing me the way to do research.

I would also like to thank Dr. Luis Gerardo de la Fraga, Dr. Gregorio Toscano Pulido, Dr. Carlos Eduardo Mariano Romero and Dr. Edgar Emmanuel Vallejo Clemente, for serving as members on my thesis committee. Their comments were very beneficial to the completion of this manuscript.

I would like to thank Prof. Qingfu Zhang for his good advice during my stay at the University of Essex, UK. Thanks also to Chixin Xiao and Christina Anastasiou for their hospitality in Essex.

In my research stays in India and Chile, I would like to thank Dra. Cristina Riff, Dra. Sanghamitra Bandyopadhyay and all my friends that I met in those stays, thanks for their hospitality.

I would like to thank my friends of the EVOCINV group, Antonio López, Alfredo Arias, Adriana Lara, Eduardo Vazquez. Thank you for sharing your good ideas and knowledge during my research work.

I will always remember my friends with whom I lived pleasant moments at the CINVESTAV: Cuauhtemoc Mancillas, William de la Cruz, Edgar Ventura, Alejandro García, Arturo Yee, Lil María, Sandra Díaz, and all students at Computer Science Department of CINVESTAV. Without forgetting all the professors and the administrative staff, thanks for your support in these four years of research.

I want to thank my family, José Lauro, María Inés and Verónica for their endless love, support and encouragement throughout my life. Without them, my successes would not have taste of victory. I would also like to thank Victor and Maru, for their unconditional support during my stay in Mexico City.

My special gratitude to Adriana Menchaca, with whom I lived beautiful moments in my stay at CINVESTAV, and who showed me that life can be easy going, and that one can be truly happy.

Finally, I acknowledge CONACyT scholarship support along these four years.

The research work presented in this thesis was derived and partially supported with funds from the CONACyT project entitled "Escalabilidad y nuevos esquemas híbridos en optimización evolutiva multiobjetivo" (Ref. 103570), whose Principal Investigator is Dr. Carlos A. Coello Coello.

Contributions

The different contributions that have been obtained during the development of this thesis, are presented below.

Book Chapter

[1] A. López Jaimes, ***S. Zapotecas Martínez***, and C. A. Coello Coello, *An Introduction to Multiobjective Optimization Techniques*, in *Optimization in Polymer Processing* (A. Gaspar-Cunha and J. A. Covas, eds.), ch. 3, pp. 29–57, New York: Nova Science Publishers, 2011. ISBN 978-1-61122-818-2.

International Conference Papers

[2] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *MOEA/D assisted by RBF Networks for Expensive Multi-Objective Optimization Problems*, in *Proceedings of the 15th annual conference on Genetic and Evolutionary Computation (GECCO'2013)*, (Amsterdam, The Neatherlands), ACM Press, July 2013, (*To appear*).

[3] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *Combining Surrogate Models and Local Search for Dealing with Expensive Multi-objective Optimization Problems*, in *2013 IEEE Congress on Evolutionary Computation (CEC'2013)*, (Cacún, México), pp. 2572–2579, IEEE Press, June 2013.

[4] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Hybridization of MOEA/D with the Nonlinear Simplex Search Algorithm*, in *2013 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM'2013)*, (Singapore), pp. 48–55, IEEE Press, April 2013.

[5] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Direct Local Search Mechanism for Decomposition-based Multi-Objective Evolutionary Algorithms*, in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, (Brisbane, Australia), pp. 3431–3438, IEEE Press, June 2012.

[6] S. Roy, ***S. Zapotecas Martínez***, C. A. Coello Coello, and S. Sengupta, *A Multi-Objective Evolutionary Approach for Linear Antenna Array Design and Synthesis*, in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, (Brisbane, Australia), pp. 3423–3430, IEEE Press, June 2012.

[7] S. Roy, ***S. Zapotecas Martínez***, C. A. Coello Coello, and S. Sengupta, *Adaptive IIR System Identification using JADE*, in *Proceedings of 2012 World Automation Congress (WAC 2012)*, (Puerto Vallarta, México), pp. 1–6, TSI Enterprises, Inc., June 2012.

[8] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Multi-objective Particle Swarm Optimizer Based on Decomposition*, in *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO'2011)*, (Dublin, Ireland), pp. 69–76, ACM Press, July 2011.

[9] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *Swarm Intelligence Guided by Multi-objective Mathematical Programming Techniques*, in *GECCO (Companion)*, (Dublin, Ireland), pp. 771–774, ACM Press, July 2011.

[10] ***S. Zapotecas Martínez***, A. Arias Montaño, and C. A. Coello Coello, *A Nonlinear Simplex Search Approach for Multi-Objective Optimization*, in *2011 IEEE Congress on Evolutionary Computation (CEC'2011)*, (New Orleans, USA), pp. 2367–2374, IEEE Press, June 2011.

[11] ***S. Zapotecas Martínez***, E. G. Yáñez Oropeza, and C. A. Coello Coello, *Self-Adaptation Techniques Applied to Multi-Objective Evolutionary Algorithms*, in *Learning and Intelligent Optimization, 5th International Conference, LION 5* (C. A. Coello Coello, ed.), vol. 6683, (Rome, Italy), pp. 567–581, Springer. Lecture Notes in Computer Science, January 2011.

[12] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model*, in *Parallel Problem Solving from Nature–PPSN XI* (R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, eds.), vol. 6238, (Kraków, Poland), pp. 576–585, Springer, Lecture Notes in Computer Science, September 2010.

[13] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Multi-Objective Meta-Model Assisted Memetic Algorithm with Non Gradient-Based Local Search*, in *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation (GECCO'2010)*, (Portland, Oregon, USA), pp. 537–538, ACM Press, July 2010. ISBN 978-1-4503-0072-8.

[14] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *A Novel Diversification Strategy for Multi-Objective Evolutionary Algorithms*, in *GECCO (Companion)*, (Portland, Oregon, USA), pp. 2031–2034, ACM Press, July 2010. ISBN 978-1-4503-0073-5.

[15] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *An Archiving Strategy Based on the Convex Hull of Individual Minima for MOEAs*, in *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, (Barcelona, Spain), pp. 912–919, IEEE Press, July 2010.

Technical Reports

[16] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *MONSS: A Multi-Objective Nonlinear Simplex Search Algorithm*, Tech. Rep. EVOCINV-01-2013, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, México, February 2013.

[17] ***S. Zapotecas Martínez*** and C. A. Coello Coello, *MOEA/D assisted by RBF Networks for Expensive Multi-Objective Optimization Problems*, Tech. Rep. EVOCINV-02-2013, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, México, February 2013.

Other Talks at Conferences

[18] ***S. Zapotecas Martínez*** and C. A. Coello Coello. *Cooperative Surrogate Models Improving Multi-objective Evolutionary Algorithms*. in *INFORMS Annual Meeting 2012*. (Phoenix, Arizona, USA), October 2012.

[19] ***S. Zapotecas Martínez*** and C. A. Coello Coello. *A Multi-Objective Nonlinear Simplex Search*. In *International Conference on Multiple Criteria Decision Making 2011 (MCDM'2011)*. (Jyväskylä, Finland), June 2011.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ABC	Artificial Bee Colony
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AEMO	Algoritmo Evolutivo Multi-objetivo
AIS	Artificial Immune System
ANOVA	Analysis of variance
CDMOMA	Cross Dominant Multi-Objective Memetic Algorithm
CFD	Computational Fluid Dynamics
CHIM	Convex Hull of Individual Minima
CMODE	Coevolutionary Multi-Objective Differential Evolution
CPU	Central Processing Unit
DE	Differential Evolution
DM	Decision Maker
DTLZ	Deb-Thiele-Laumanns-Zitzler
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolution Strategy
GA	Genetic Algorithm

I_H	Hypervolume
I_C	Two Set Coverage
KKT	Karush-Kuhn-Tucker
M-PAES	Memetic Pareto Archived Evolution Strategy
MA	Memetic Algorithm
MADA	Multi-Attribute Decision Analysis
MCDM	Multi-Criteria Decision Making
MOEA/D+LS-II	Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II
MOEA/D+LS	Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOEA	Multi-Objective Evolutionary Algorithm
MOEA/D-EGO	Multi-Objective Evolutionary Algorithm based on Decomposition with Gaussian Process Model
MOEA/D-RBF	Multi-Objective Evolutionary Algorithm based on Decomposition assisted by Radial Basis Functions
MOEA/D-RBF+LS	MOEA/D-RBF with Local Search
MOGA	Multi-Objective Genetic Algorithm
MOGLS	Multi-Objective Genetic Local Search
MOMA	Multi-Objective Memetic Algorithm
MONSS	Multi-objective Nonlinear Simplex Search
MOP	Multi-objective Optimization Problem
NBI	Normal Boundary Intersection

NSDE	Non-dominated Sorting Differential Evolution
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NSGA	Non-dominated Sorting Genetic Algorithm
NSS-GA	Nonlinear Simplex Search Genetic Algorithm
NSS	Nonlinear Simplex Search
OR	Operations Research
PARSEC	PARametric SECtion
PBI	Penalty Boundary Intersection
PBM	Polynomial-Based Mutation
PDMOSA	Pareto Domination Multi-Objective Simulated Annealing
$\mathcal{PF}$	Pareto front
PMA	Pareto Memetic Algorithm
POM	Problema de Optimización Multi-objetivo
$\mathcal{PS}$	Pareto optimal set
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
SBX	Simulated Binary Crossover
SMS-EMOA	S-Metric Selection Evolutionary Multi-objective Optimization Algorithm
SPEA	Strength Pareto Evolutionary Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2
SS	Scatter Search
SVR	Support Vector Regresion
VEGA	Vector Evaluated Genetic Algorithm

WFG	Walking-Fish-Group
ZDT	Zitzler-Deb-Thiele

1

Introduction

In engineering and scientific applications, there exist problems that involve the simultaneous optimization of several objectives. Usually, such objectives are conflicting such that no single solution is simultaneously optimal with respect to all objectives. These types of problems are known as *Multi-objective Optimization Problems (MOPs)*. In contrast to single-objective optimization (where a single optimal solution is aimed for), in multi-objective optimization, a set of solutions with different trade-offs among the objectives is usually achieved. The method most commonly adopted in multi-objective optimization to compare solutions is the well-known *Pareto dominance relation* [106]. Therefore, optimal solutions in multi-objective optimization, are called Pareto optimal solutions and all of them constitute the so-called *Pareto optimal set (PS)*. The evaluation of solutions in PS using the objective functions is collectively known as *Pareto front (PF)*.

Since their origins, *Multi-Criteria Decision Making (MCDM)* techniques have shown to be an effective tool for solving MOPs, at a reasonably low computational cost. However, in real-world applications, there exist several MOPs for which MCDM techniques cannot guarantee that the solution obtained is optimum. Furthermore, these methods can be inefficient and sometimes even inapplicable for a particular problem. For these more complex optimization problems, the use of meta-heuristics is fully justified. *Multi-Objective Evolutionary Algorithms (MOEAs)* are meta-heuristics which, in recent years, have become very popular because of their conceptual simplicity and efficiency in these types of problems. For their nature (based on a population), MOEAs allow to generate multiple elements of the PS in a single run. Therefore, nowadays, MOEAs constitute one of the most successful approaches for solving MOPs.

1.1 Problem Statement

Traditional mathematical programming methods for solving both single- and multi-objective optimization problems have shown to be an effective tool in many science and engineering problems. However, this type of methods cannot guarantee that the solution obtained is optimum in the most general optimization problem. In the specialized literature, there exist several mathematical programming techniques available to solve MOPs, see for example [35, 58, 96, 138]. However, some researchers have identified several limitations of these traditional mathematical programming approaches [13, 23, 42, 95], including the fact that many of them generate a single nondominated solution per run, and that many others cannot properly handle non-convex, or disconnected Pareto fronts.

The nature of MOEAs (based on a population) and their flexible selection mechanisms have proved to be extremely useful and successful for dealing with MOPs. MOEAs possess the advantage of not requiring previous information of the problem as most traditional mathematical programming methods. Therefore, they do not need either an initial search point or the gradient information of a function to approximate solutions to the $\mathcal{PS}$. Instead of this, they have been designed with two main goals in mind:

1. *maximize the number of elements of the $\mathcal{PS}$ obtained*, and
2. *distribute such solutions as uniformly as possible along the $\mathcal{PF}$*.

However, MOEAs normally require a relatively high number of objective function evaluations in order to produce a reasonably good approximation to the $\mathcal{PF}$ of a MOP. This remains as one of their main limitations when applied to real-world applications, particularly when dealing with objective functions that are computationally expensive to evaluate.

In order to address this issue, a variety of hybrid approaches combining mathematical programming techniques with a MOEA have been proposed. In this way, while the MOEA explores the whole search space, mathematical programming techniques exploit the promising regions given by the same MOEA. However, when the gradient information of the functions is not available, mathematical programming techniques become impractical, and then, we look for

alternative search strategies to address this issue, such as the direct search methods—i. e., mathematical programming methods that do not require gradient information of the functions.

1.2 Our Proposal

In this thesis, we investigate different strategies to hybridize MOEAs with a popular direct search method (the *Nonlinear Simplex Search (NSS)* algorithm). The contributions presented here, follow the two main goals mentioned in the previous section. The first contribution presented in this thesis, consists of a *Multi-objective Nonlinear Simplex Search (MONSS)* approach. This proposal turns out to be effective and competitive when dealing with MOPs having moderate and low dimensionality. Based on experimental evidence, we concluded that the NSS is a good alternative to be used as a local search engine into a MOEA. The design of local search mechanisms coupled to MOEAs by using direct search methods and having a low computational cost (in terms of the number of fitness function evaluations performed) is an open research problem. Some attempts for the hybridization between these two types of algorithms are presented in Chapter 4. In order to investigate efficient manners of using direct search methods to approximate solutions to the $\mathcal{PS}$, in Chapter 6, we propose a *Multi-Objective Memetic Algorithm (MOMA)* based on the NSS. Preliminary results show that the proposed approach is, in general, a competitive tool to deal with MOPs having moderate and high dimensionality in decision variable space. Some weaknesses of this MOMA are noted and addressed in Chapter 7, giving rise to an enhanced version of the hybrid approach presented in Chapter 6. The use of a low number of fitness function evaluations in MOEAs is an important issue in multi-objective optimization, because there are several real-world applications that are computationally expensive to solve. In order to build a more efficient MOMA, in Chapter 8, we present a hybridization between the NSS and a MOEA assisted by surrogate models. Preliminary results show that the proposed approach is a viable choice to deal with MOPs having different features, and the applicability to real-world applications could speed up convergence to the $\mathcal{PF}$ in comparison to conventional MOEAs.

1.3 General and Specific Goals of the Thesis

1.3.1 Main goal

The main goal of this research is to advance the state-of-the-art with respect to the design of hybrid algorithms, which combine MOEAs with non-gradient mathematical programing techniques.

1.3.2 Specific goals

- To study different direct search methods proposed in the mathematical programming literature, analyzing their main advantages and disadvantages in terms of their possible coupling with a MOEA.
- To study the state of the art regarding MOEAs, including their foundations, main mechanisms, performance measures, operators, density estimators and their advantages and disadvantages.
- To design strategies that combine the properties of traditional non-gradient mathematical programming methods with the exploratory power of a MOEA.
- To validate the proposed strategies with respect to state-of-the-art MOEAs using standard test problems and performance measures reported in the specialized literature.
- Perform a detailed statistical study of the proposed strategies in order to determine the parameters, to which they are most sensitive.

1.4 Structure of the Document

This document is organized in nine chapters and two appendices. The first three chapters (including this one) describe basic concepts required to understand the contributions of this thesis work. The last five chapters present the current contributions and their corresponding conclusion. The document is organized as follows.

Chapter 2 presents the basic notions related to optimization and evolutionary computation. The main goal of this chapter is to get acquainted with the concepts, definitions and notations used in the remainder of this document. In Chapter 3, we present a brief introduction to MOPs. This chapter describes some mathematical and evolutionary approaches for solving MOPs; additionally, some performance measures and test problems to evaluate MOEAs are also introduced. Chapter 4 presents the state of the art regarding hybrid algorithms that combine direct search methods with MOEAs. The first contribution of the thesis is presented Chapter 5. In this chapter we preset the design and results of a novel *Multi-objective Nonlinear Simplex Search (MONSS)* which is an extension of the NSS for multi-objective optimization. In Chapter 6, we present a MOMA based on the NSS. Preliminary results indicate that the proposed approach is, in general, a competitive tool to deal with the MOPs adopted. In Chapter 7, some weaknesses of the MOMA presented in Chapter 6 are reported and addressed, giving rise to an enhanced version of this hybrid approach. In order to build a more efficient MOMA, in Chapter 8, we present a hybridization between the NSS and a MOEA assisted by surrogate models. Preliminary results show that the proposed approach is a viable choice to deal with MOPs having different features. Such results lead us to believe that its applicability to real-world applications could speed up convergence to the $\mathcal{PF}$ in comparison to conventional MOEAs. In Chapter 9, we present the conclusions obtained regarding our current contributions. Also, we describe some possible paths for future research. In Appendix A, we describe in detail, the standard test problems adopted to validate the proposed algorithms presented in this thesis. Finally, Appendix B describes an airfoil shape problem, which has been adopted to assess the performance of the MOMA assisted by surrogate models which is introduced in Chapter 8.

2

Background

This chapter presents some basic concepts related to *optimization* and *Evolutionary Computation (EC)*. The most important aim of this chapter is that the reader familiarizes with the basic concepts, definitions and notations used in the remainder of this thesis. Section 2.1 provides the conceptual and theoretical basis for global optimization. A classification of different mathematical programming methods for solving nonlinear optimization problems is presented in Section 2.2. Section 2.3 provides a brief description of evolutionary approaches for solving optimization problems. Section 2.4 introduces the most important paradigms available within EC. Section 2.5 presents a brief description of memetic algorithms which is of interest in this work. Finally, Section 2.6 describes the advantages and disadvantages of using these bio-inspired approaches in the optimization field.

2.1 Notions of Optimality

In mathematics, optimization refers to the process of determining the minimum or maximum point of a function by choosing systematically the values of its corresponding decision variables within a certain search space. To be more precise, a generic optimization problem [1] can be formally stated as follows.

Definition 2.1 (Optimization Problem)
Find the vector $\mathbf{x}$ which minimizes the function $f(\mathbf{x})$ subject to $\mathbf{x} \in \Omega$, where $\Omega \subseteq \mathbb{R}^n$ is the feasible region which satisfies the p inequality constraints:

$$g_i(\mathbf{x}) \leqslant 0; \qquad i = 1\ldots, p$$

and the q equality constraints:

$$h_j(\mathbf{x}) = 0; \qquad j = 1\ldots, q$$

1 Without loss of generality, in this thesis we will assume minimization problems

where Ω defines the subspace of feasible solutions and f is commonly called *objective function*. The feasible solution $\mathbf{x}^\star \in \Omega$ that corresponds to the minimum value of the objective function in all the search space is called *global optimum*.

The hardness of an optimization problem is determined by the different types of mathematical relationships among the objective function, the constraints and the range of the decision variables. To understand the complexity involved in solving an optimization problem, the following definitions are introduced.

Definition 2.2 (Global Minimum)
Given a function $f : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$, $\Omega \neq \phi$, for $\mathbf{x}^\star \in \Omega$ the value $f^\star = f(\mathbf{x}^\star) > -\infty$ is called *global minimum*, if and only if:

$$\forall \mathbf{x} \in \Omega : f(\mathbf{x}^\star) \leqslant f(\mathbf{x}) \tag{2.1}$$

where vector $\mathbf{x}^\star$ is a *global minimum point*

Definition 2.3 (Local Minimum)
Given a function $f : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$, a solution $\mathbf{x}^o \in \Omega$ is called *local minimum point*, if and only if:

$$\forall \mathbf{x} \in \Omega : f(\mathbf{x}^o) \leqslant f(\mathbf{x}), \qquad \text{such as: } \|\mathbf{x} - \mathbf{x}^o\| < \epsilon \tag{2.2}$$

where $\epsilon > 0$ and the value $f(\mathbf{x}^o)$ is called *local minimum*.

Definition 2.4 (Convex Function)
A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is called *convex*, if for any two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$:

$$f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leqslant \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2) \tag{2.3}$$

where $\lambda \in [0, 1]$.

In this way, if the objective function and all the constraints are convex, it is possible to find in an exact manner the globally optimal solution, and solve the problem up to a very large number of decision variables. On the other hand, if the function is non-convex, the problem is much harder to solve and it becomes much more difficult to locate the feasible region and, therefore to find the global optimum.

2.1.1 Optimality Criterion

In the early 1950s, and in an independent way, Karush [70] as well as Kuhn and Tucker [81] derived the optimality conditions for an optimization problem. This laid the foundations for the development of the optimization field. These conditions provide the necessary and sufficient requirements that an optimal solution must satisfy. Formally, the *Karush-Kuhn-Tucker (KKT) conditions* can be stated as follows.

Definition 2.5 (KKT Necessary Conditions)
Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be the objective function. Let $g_i : \mathbb{R}^n \mapsto \mathbb{R}$ and $h_j : \mathbb{R}^n \mapsto \mathbb{R}$ be the inequality and the equality constraint functions, respectively. The KKT conditions or KKT problem is defined as finding the vector $\mathbf{x}^\star$, and the constants u_i $(i = 1, \ldots, p)$ and v_j $(j = 1, \ldots, q)$ that satisfy:

$$\begin{aligned} \nabla f(\mathbf{x}^\star) - \textstyle\sum_{i=1}^{p} u_i \nabla g_i(\mathbf{x}^\star) - \sum_{j=1}^{q} v_j \nabla h_j(\mathbf{x}^\star) = 0, \quad & \text{subject to:} \\ g_i(\mathbf{x}^\star) \geqslant 0, \quad & \text{for all } i = 1, \ldots, p \\ h_j(\mathbf{x}^\star) = 0, \quad & \text{for all } j = 1, \ldots, q \\ u_i g_i(\mathbf{x}^\star) = 0, \quad & \text{for all } i = 1, \ldots, p \\ u_i \geqslant 0, \quad & \text{for all } i = 1, \ldots, p \end{aligned} \tag{2.4}$$

In particular, if $p = 0$, i.e., without inequality constraints, these KKT *conditions* turn into *Lagrange conditions*, and the KKT *multipliers* are called *Lagrange multipliers*.

Although in some cases the necessary conditions are also sufficient for optimality, in general, additional information is necessary. Therefore, certain additional convexity assumptions are needed to guarantee that the solution $\mathbf{x}^\star$ is optimal. These conditions are called *sufficient conditions* and they are presented in the following theorem.

Theorem 1 (KKT Necessity Theorem)
Consider the nonlinear optimization problem described by definition 2.1. Let f, g_i and h_j be differentiable functions and $\mathbf{x}^\star$ be a feasible solution to the optimization problem. Let $I = \{i | g_i(\mathbf{x}^\star) = 0\}$. Furthermore, $\nabla g_i(\mathbf{x}^\star)$ for $i \in I$ and $\nabla h_j(\mathbf{x}^\star)$ for $j = 1, \ldots, q$ are linearly independent. If $\mathbf{x}^\star$ is an optimal solution to the optimization problem, then there exist vectors $\mathbf{u}^\star$ and $\mathbf{v}^\star$ such that $\mathbf{x}^\star, \mathbf{u}^\star$ and $\mathbf{v}^\star$ solve the KKT conditions given by equation (2.4).

2.2 Optimization Techniques

Over the years, a large number of mathematical programming techniques for solving nonlinear optimization problems have been proposed. However, it was after KKT's work that a large number of nonlinear programming methods were developed for solving nonlinear optimization problems. Comprehensive surveys of these mathematical programming methods can be found in [3, 17, 25, 104, 111, 112].

The development of these optimization methods has been motivated by different problems in the real world. Over the years, different taxonomies for classifying optimization methods have been proposed (see, for example, those presented in [25, 111]). For the purposes of this thesis, we classify these methods in two different categories: *mathematical programming techniques* and *stochastic techniques*, see Figure 2.1.

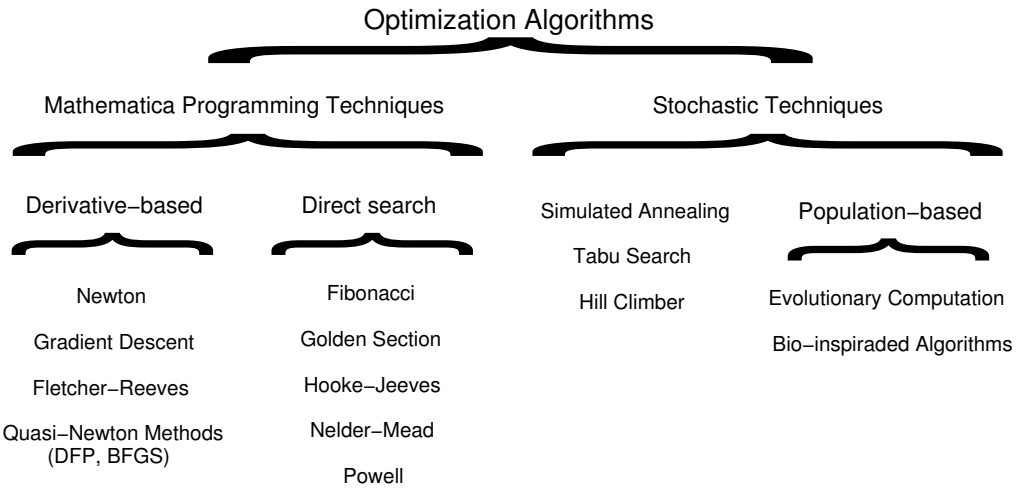


Figure 2.1.: A taxonomy of optimization techniques

2.2.1 Mathematical Programming Techniques

The classical or *mathematical programming* methods are deterministic algorithms characterized by having specific rules to move from one solution to another. These methods have been used for some time and have been successfully applied in many problems in engineering. Currently, there are different variations of these meth-

ods [3, 17, 25, 104, 111, 112]. Based on their conceptual foundations, these methods can be divided in two large groups: *gradient* and *non-gradient* mathematical programing methods.

GRADIENT TECHNIQUES. These methods use information from the derivatives of the function as their strategy to move from one solution to another. In the *Operations Research (OR)* literature, there exist several of these algorithms, which have been applied both to one-dimensional problems and to multi-dimensional optimization problems (e.g. Cauchy's method [10] (or steepest descent), Newton's method [103], Fletcher and Reeves's method [38] (or conjugate gradients), etc.).

NON-GRADIENT TECHNIQUES. Non-gradient methods or direct search methods, are techniques that do not require any information of the derivatives of the function and constitute a good alternative when the function to be optimized is not differentiable. Similar to previous methods, there exist algorithms for solving one-dimensional and multi-dimensional optimization problems (e.g. Hooke and Jeeves's method [60], Nelder and Mead's method [102], Zangwill's method [144], etc).

Unfortunately, none of these mathematical methods guarantees convergence to the global optimum when dealing with the general nonlinear optimization problem. In most cases, these methods rely on an initial search point and, when dealing with multi-modal problems (i.e., problems that have several local optima) most of these methods get easily trapped in local optima and are unable to reach the global optimum.

2.2.2 Stochastic Techniques

The *non-classical* or *stochastic* methods, are algorithms that usually employ probabilistic transition rules. These methods include evolutionary computation, which in recent years, has become very popular especially when dealing with hard optimization problems. This type of approaches, in comparison, are new and quite useful, because they have certain properties that deterministic algorithms do not have. These stochastic techniques possess certain advantages that traditional methods do not have, and they do not require previous information

of the problem. Indeed, they do not need neither an initial search point nor any gradient information as most traditional mathematical programming methods. But not only evolutionary computation has been used to deal with optimization problems; other stochastic algorithms (see for example [31, 49, 71, 72, 122]) have also been found to be useful to deal with complex optimization problems.

2.3 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are methods inspired on natural selection (particularly, the "survival of the fittest" principle). EAs are techniques based on the use of a population, i.e., they operate on a set of solutions instead of operating on one solution at a time, as traditional optimization methods. At each iteration of an EA, a competitive selection mechanism that tends to preserve the fittest solutions is applied. The solutions with the highest fitness values have the highest probability of being recombined with other solutions to mix information and form new solutions, which are expected to be better than their predecessors. This process is repeated until a termination condition is reached. The pseudo code of an evolutionary algorithm is shown in Algorithm 1. The main components, procedures and operators that must be specified in order to define a particular EA are described below.

Algorithm 1: General scheme of an *Evolutionary Algorithm (EA)*

```
1  begin
2      t = 0;
3      INITIALIZATION: P(t) ∈ I^μ;
4      EVALUATION: Φ(P(t));
5      while (ι(P(t)) ≠ TRUE) do
6          RECOMBINATION: P'(t) = r(P(t));
7          MUTATION: P''(t) = m(P'(t));
8          EVALUATION: Φ(P''(t));
9          SELECTION: P(t+1) = s(P(t) ∪ P''(t));
10         t = t + 1;
11     end
12 end
```

Representation (encoding of individuals). To link a real world problem with an EA, we need to use a particular representation of the decision variables. There are two levels of representation used in an EA: genotypic and phenotypic. The genotype is the encoding adopted in the chromosome and its corresponding genes. The phenotype is the result of decoding the values of the chromosome into the decision variable space of the problem.

Fitness Function (objective function). The fitness of an individual is related to the objective function value and represents the task to be solved by the evolutionary algorithm. The evaluation function assigns a quality measure to each individual that allows to compare it with respect to the other solutions in the population.

Population. The population is the set of solutions adopted by the EA to perform the search. The population is responsible for maintaining diversity, so that the EA does not get stuck in local optima. Thus, it is important that the initial population contains solutions that are spread over all the search space.

Parent Selection Mechanism. This mechanism allows the best individuals within the population to become parents of the next generation and guides the search towards solutions with a higher fitness. An individual is selected as a Parent if it survives the selection process. Different types of selection schemes are possible. For example: *proportional*, *stochastic*, *deterministic* or based on *tournaments*.

Variation Operators (recombination and mutation). Their function is to modify the way in which the parents are combined to form the offspring. The crossover (or recombination operator) uses two or more parents to generate one or two offspring. The main principle behind recombination is to produce an offspring that combines the selected parents to form better individuals into the search space. The mutation operator is applied to only one solution and slightly modifies the genetic information of the offspring. The general idea of the mutation operator is to allow jumps (or abruptly move) from one region to another in the search space.

Survivor Selection Mechanism (replacement). This mechanism helps the EA to distinguish among individuals based on their fitness or quality, favoring those with the highest quality.

As the EAs are stochastic methods, there are no guarantees that the best final solutions have reached the global optimum by the time the stopping condition has been reached. Thus, the termination condition is an important issue in EAs. If the optimization problem has a known optimal solution, the EA should stop when the objective function reaches the desired level of accuracy. If not, then the user should determine the number of generations allowed, the maximum allowable *Central Processing Unit (CPU)* time, the maximum number of fitness evaluations, or some other similar criterion.

2.4 Evolutionary Computation Paradigms

In EC, there exist three main paradigms: *i) Evolution Strategies (ESs)*, *ii) Evolutionary Programming (EP)* and *iii) Genetic Algorithms (GAs)*. In the following, a brief description of the most important paradigms is presented.

2.4.1 Evolution Strategies

ESs were proposed in 1965 by Rechenberg [113] and Schwefel [120] in Germany. These techniques were originally developed to solve hydrodynamic optimization problems having a high degree of complexity. ESs not only evolve the decision variables of the problem but also the parameters of the techniques (such mechanism is called self-adaptation). This technique simulates the evolutionary process to an individual level, and, therefore, recombination is possible although, it is normally a secondary operator (i.e., less important than mutation). The original ES proposal was not based on a population but only on the use of one individual. However, population-based ESs were introduced by Schwefel [121] a few years later. ESs normally adopt one of two possible selection schemes:

i Plus selection $(\mu + \lambda)$: in this case, the next population is generated from the union of parents and children, and

ii Comma selection (μ, λ): in this case, the next population is generated only from the children.

Algorithm 2 shows an outline of a simple ES using an initial population μ.

Algorithm 2: Evolution Strategy

Input: A number μ of individuals
Output: An evolved population $P(t)$

begin
  $t = 0$;
  INITIALIZATION: $P(t) \in I^{\mu}$;
  EVALUATION: $\Phi(P(t))$;
  while $(\iota(P(t)) \neq \text{TRUE})$ **do**
    RECOMBINATION: $P'(t) = r(P(t))$;
    MUTATION: $P''(t) = m(P'(t))$;
    EVALUATION: $\Phi(P''(t))$;
    SELECTION:

$$P(t+1) = \begin{cases} s(P(t) \cup P''(t)) & (\mu + \lambda)\text{-SELECTION} \\ P''(t) & (\mu, \lambda)\text{-SELECTION} \end{cases};$$

    $t = t + 1$;
  end
end

2.4.2 Evolutionary Programming

EP was developed by Fogel in the mid-1960s. Fogel simulated the natural evolution as a learning process, aiming to generate artificial intelligence [41, 42].

The parent selection in EP is deterministic and every member of the population creates exactly one offspring via mutation. The crossover in EP is not used, as the members of the population are viewed as part of a specific species rather than as members of the same species, and different species are not allowed to recombine (as happens in nature). After having created the offspring, each solution is evaluated and a $(\mu + \lambda)$-selection is normally adopted. Therefore, each solution participates with other solutions in a binary tournament assigning

a "win" if one solution is better than its opponent. Finally, the μ solutions with the greatest number of wins are retained to be the parents of the next generation, see Algorithm 3.

Algorithm 3: *Evolutionary Programming (EP)*

Input: A number μ of individuals
Output: An evolved population $P(t)$

```
1  begin
2      t = 0;
3      Initialization: P(t) ∈ I^μ;
4      Evaluation: Φ(P(t));
5      while (ι(P(t)) ≠ TRUE) do
6          Mutation: P'(t) = m(P'(t));
7          Evaluation: Φ(P'(t));
8          Selection: P(t + 1) = s(P(t) ∪ P'(t));
9          t = t + 1;
10     end
11 end
```

2.4.3 Genetic Algorithms

GAs were originally called genetic "reproductive plans" and were developed in the early 1960s by Holland [59], aiming to solve machine learning problems. This type of evolutionary algorithm is characterized mainly by coding individuals (traditionally using a binary string), and for having a probabilistic selection mechanism. Crossover plays a major role in GAs, but a mutation operator is also adopted to maintain good exploratory capabilities, see Algorithm 4. GAs work at the genotypic level and normally do not adopt a self-adaptation mechanism as ESs, although some proposals in that regard have been studied in the specialized literature [20, 119]. Additionally, there is another operator called elitism which plays a crucial role in GAs. This operator retains the best individual produced at each generation, and passes it intact (i.e. without being recombined or mutated) to the following generation. Rudolph [116] showed that a GA requires elitism to converge to the optimum. For this reason, elitism is a mechanism that has become standard in EAs.

Algorithm 4: Simple *Genetic Algorithm (GA)*

Input: A number μ of individuals
Output: A evolved population $P(t)$

```
1  begin
2      t = 0;
3      Initialization: P(t) ∈ I^μ;
4      while (ι(P(t)) ≠ TRUE) do
5          Evaluation: Φ(P(t));
6          Selection: P_p(t) = s(P(t));
7          Recombination: P_r(t) = r(P_p(t));
8          Mutation: P_m(t) = m(P_r(t));
9          P(t + 1) = P_m(t);
10         t = t + 1;
11     end
12 end
```

Currently, there exist many variants of GAs, with different solution encodings, as well as a variety of selection, crossover and mutation operators [5]. Nevertheless, the characteristics of the so-called simple GA are the use of binary representation, fitness proportional selection, bit-flip mutation (an operator that is applied with a low probability), and 1-point crossover (which is applied with a high probability) [52].

2.4.4 Other Evolutionary Approaches

In spite of the success of EAs, there are other bio-inspired approaches which are not included in the three main paradigms but that, in the last few years, have been widely used to solve optimization problems. They are: *Artificial Immune Systems (AISs)* [22], *Ant Colony Optimization (ACO)* [30], *Scatter Search (SS)* [49], *Artificial Bee Colony (ABC)* [69], *Particle Swarm Optimization (PSO)* [71] and *Differential Evolution (DE)* [129], among others.

2.5 Memetic Algorithms

The term *Memetic Algorithm (MA)* was first introduced in 1989 by Moscato [97]. The term "memetic" has its roots in the word "meme" introduced by Dawkins in 1976 [21] to denote the unit of imitation in cultural transmission. The essential idea behind MAs is the combination of local search refinement techniques with a population-based strategy, such as evolutionary algorithms. In fact, for the purposes of this work, we will assume that the population-based strategy adopted by the MA is an evolutionary algorithm. The main difference between genetic and memetic algorithms is the approach and view of the information's transmission techniques. In GAs, the genetic information carried by genes is usually transmitted intact to the offspring, whereas in MAs, the base units are the so-called "memes" and they are typically adapted by the individual transmitting information. While GAs are good at exploring the solution space from a set of candidate solutions, MAs explore from a single point, allowing to exploit solutions that are close to the optimal solutions. Some important decisions that should be taken when designing MAs are the following:

a) In which moment of the evolutionary process should the local search be performed?

b) How often should the local search be applied along the entire evolutionary process?, and

c) From which solutions should the local search be started?

The combination of local improvement operators among the evolutionary steps of an EA is essential to improve solutions that are close from becoming optimal. This has been shown in several application domains to bring improvements to the standard results achieved by standalone GAs in terms of quality of the results and speed of convergence. In general, there is no specific method to design a MA. However, Algorithm 5 shows a general framework of what a MA should contain.

Algorithm 5: General scheme of a *Memetic Algorithm (MA)*

Input: A number μ of individuals
Output: An evolved population $P(t)$

begin
  $t = 0$;
  INITIALIZATION: $P(t) \in I^{\mu}$;
  while $(\iota(P(t)) \neq \text{TRUE})$ **do**
    EVALUATION: $\Phi(P(t))$;
    EVOLVE: $Q(t) = evo(P(t))$ `// using stochastic operators;`
    SELECTION: $R(t) \subseteq Q(t)$ `// select a set of solutions` (R);
    forall the $r^t \in R(t)$ **do**
      IMPROVE: $r^t = i(r^t)$ `// using a improvement mechanism` (i);
    end
    SELECTION: $P(t+1) = s(P(t) \cup Q(t) \cup R(t))$ `// next generation;`
    $t = t + 1$;
  end
end

2.6 Advantages and Disadvantages of Evolutionary Algorithms

EAs have as their primary advantage, that they are conceptually simple. We have provided description of different types of EAs. Each algorithm consists of an initialization process, which may be a purely random sampling of possible solutions, followed by the use of variation operators and selection in light of a performance index. EAs can be applied to virtually any problem that can be formulated as an optimization task. EAs require a data structure to represent solutions, a performance index to evaluate solutions, and variation operators to generate new solutions from old solutions (selection is also required but is less dependent on human preferences). Real-world optimization problems often impose nonlinear constraints, involve nonstationary conditions, incorporate noisy observations or random processing, or include other components that do not conform well to the prerrequisites of classic optimization techniques. Moreover, real-world problems are often multi-modal, and gradient-based methods rapidly converge to local optima (or perhaps saddle points) which may yield insufficient performance. For these types of problems, EAs have shown to be a good choice. An important aspect to consider is that EAs offer a framework such that it is comparably easy to incorporate specific domain knowledge. For example, specific variation operators may be known to be useful when applied to particular representations. Actually, the search space can be exploited by using either mathematical or stochastic methods (including memetic algorithms). From a computational point of view, the evolutionary process can be highly parallel. As distributed processing computers become more readily available, there will be a corresponding increased potential for applying EAs to more complex problems. A solution can be handled in parallel, whereas the selection mechanism (which requires at least pairwise comparisons) requires serial processing. Traditional methods of optimization are not robust to dynamic changes in the environment and often require a complete restart in order to provide a solution. In contrast, EAs can be used to adapt solutions to changing circumstances. The population provides a basis for further improvement and in most cases it is not necessary, nor desirable, to reinitialize

the population at random. Indeed, these adaption capabilities can be advantageous when dealing with dynamic environments.

On the other hand, since EAs are general search algorithms, it can be difficult to "fine tune" their parameters to work well on any specific problem. This is, indeed, one of the main drawbacks of EAs, since this fine tuning is normally done by hand. As already mentioned before, the choice of the representation and the construction of the fitness function depend directly on the problem at hand. A bad choice for any of these can make the algorithm to perform poorly. Unfortunately, it remains unknown how to make the best parameter choices, or how to construct the best fitness function for a given a problem. Most of this relies on trial and error, and often much thinking and testing needs to be done before the algorithm performs reasonably well. Moreover, in the specialized literature there are many different operators to choose from (selection, crossover, and mutation methods, etc.), and several parameters to set (population size, crossover and mutation rates, etc.). Furthermore, premature convergence to a local optimum may result from a wrong adverse configuration and not yield (a point nearby) the global optimum. Finally, EAs do not guarantee convergence towards an optimal solution in a finite amount of time. In addition, the stochastic nature of EAs makes it hard to know if they have reached the global optimum and convergence cannot, in general, be guaranteed. Thus, it is advisable to run the EA several times, each time starting with a different (random) initial population, and perhaps using different parameter settings. The best solution over all these runs can then be taken as the best approximation to the optimum.

3

Multi-Objective Optimization

Multi-objective optimization (or multi-objective programming), is the process of simultaneously optimizing a vector function whose elements represent the objective functions subject to certain domain constraints. These functions form a mathematical description of performance criteria which are usually in conflict with each other. In order to understand the type of problems that we are interested on, the following definition is introduced.

Definition 3.1 (Multi-objective Optimization Problem)
Formally, a *Multi-objective Optimization Problem (MOP)* is defined as:

Without loss of generality, we assume minimization problems.

$$\text{Minimize:} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T \tag{3.1}$$

subject to:

$$h_i(\mathbf{x}) = 0, \qquad i = 1, \ldots, p \tag{3.2}$$

$$g_j(\mathbf{x}) \leqslant 0, \qquad j = 1, \ldots, q \tag{3.3}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is the vector of decision variables, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, k$ are the *objective functions* and $h_i, g_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$, $j = 1, \ldots, q$ are the constraint functions of the problem. Equations (3.2) and (3.3) determine the *feasible region* $\Omega \subseteq \mathbb{R}^n$ and any decision vector $\mathbf{x} \in \Omega$ defines a feasible solution of the MOP. $\mathbf{F} : \Omega \to \mathcal{F}$ is a function that maps the *decision variable space* $\Omega \subseteq \mathbb{R}^n$ into the *objective space* $\mathcal{F} \subseteq \mathbb{R}^k$, which contains all the possible values of the functions, see Figure 3.1.

Note however that, the decision variables x_i $(i = 1, \ldots, n)$ can be continuous or discrete—in this work we are only interested in continuous domains which are contained on $\mathbb{R}^n$. When the functions g_i and h_i are not present, the above problem is called unconstrained MOP. If all the objective functions and the constraint functions are linear, the problem 3.1 is called a linear MOP. If at least one of the functions is nonlinear, the problem is then called a nonlinear MOP. If

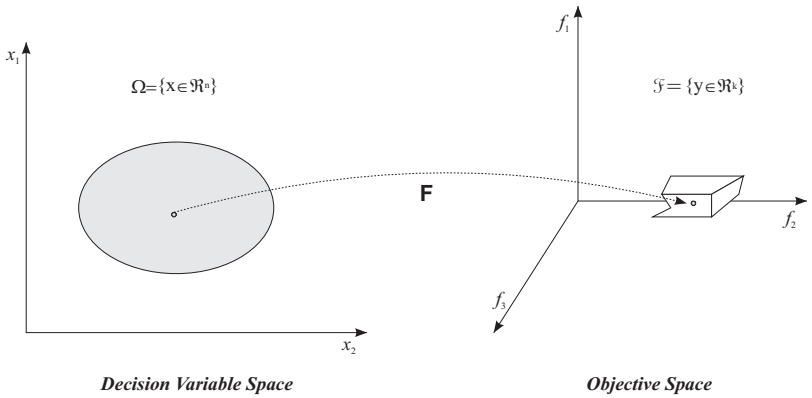


Figure 3.1.: Mapping the decision variable space Ω to the objective space $\mathcal{F}$.

all the objective functions are convex, and the feasible region is also convex, the problem is known as a convex MOP. In this study we are interested in solving nonlinear unconstrained MOPs.

Solving a MOP is very different that solving a single-objective optimization problem. In single-objective optimization, it is possible to determine between any given pair of solutions if one is better than the other by comparing the function values. As a result, we usually obtain a single optimal solution (i. e., the global optimum). On the other hand, in multi-objective optimization there does not exist a straightforward method to determine if a solution is better than another one. The method most commonly adopted in multi-objective optimization to compare solutions is called Pareto dominance relation, which was originally proposed by Edgeworth in 1881 [34], and later generalized by Pareto in 1896 [106]. This relation establishes that the aim when solving a MOP is to find the best possible trade-offs among all the objectives. This leads to the generation of a set of solutions, instead of only one (as happens in single-objective optimization).

Therefore, in multi-objective optimization, we aim to produce a set of trade-off solutions representing the best possible compromises among the objectives (i.e., solutions such that no objective can be improved without worsening another). In order to describe the concept of optimality in which we are interested on, the following concepts are introduced [96].

3.1 Optimality in Multi-Objective Optimization

Definition 3.2 (Pareto dominance)
Let $\mathbf{x}, \mathbf{y} \in \Omega$, we say that $\mathbf{x}$ *dominates* $\mathbf{y}$ (denoted by $\mathbf{x} \prec \mathbf{y}$) if and only if, $f_i(\mathbf{x}) \leqslant f_i(\mathbf{y})$ and $f_i(\mathbf{x}) < f_i(\mathbf{y})$ in at least one f_i for all $i = 1, \ldots, k$, see Figure 3.2.

Definition 3.3 (Pareto optimal)
Let $\mathbf{x}^\star \in \Omega$, we say that $\mathbf{x}^\star$ is a *Pareto optimal* solution, if there is no other solution $\mathbf{y} \in \Omega$ such that: $\mathbf{y} \prec \mathbf{x}^\star$.

Definition 3.4 (Pareto optimal set)
The *Pareto optimal set (*$\mathcal{PS}$*)* is defined by:

$$\mathcal{PS} = \{\mathbf{x} \in \Omega | \mathbf{x} \text{ is a } \textit{Pareto optimal} \text{ solution}\}$$

Definition 3.5 (Pareto optimal front)
The *Pareto front (*$\mathcal{PF}$*)* is defined by:

$$\mathcal{PF} = \{\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) | \mathbf{x} \in \mathcal{PS}\}$$

In general, it is not possible to find an analytical expression that defines the $\mathcal{PF}$ of a MOP. Thus, the most common way to get the $\mathcal{PF}$ is to compute a sufficient number of points in the feasible region, and then filter out the nondominated vectors from them. The desirable aim in multi-objective optimization, is to determine the $\mathcal{PS}$ from the feasible region Ω, i. e., to find all the decision variables that satisfy definition 3.3. Note however that in practice, not all the $\mathcal{PS}$ is normally desirable (e.g., it may not be desirable to have different solutions that map to the same values in objective function space) or achievable. Therefore, we are interested in maximizing the number of elements of the $\mathcal{PS}$ and maintaining a well-distributed set of solutions along the $\mathcal{PF}$.

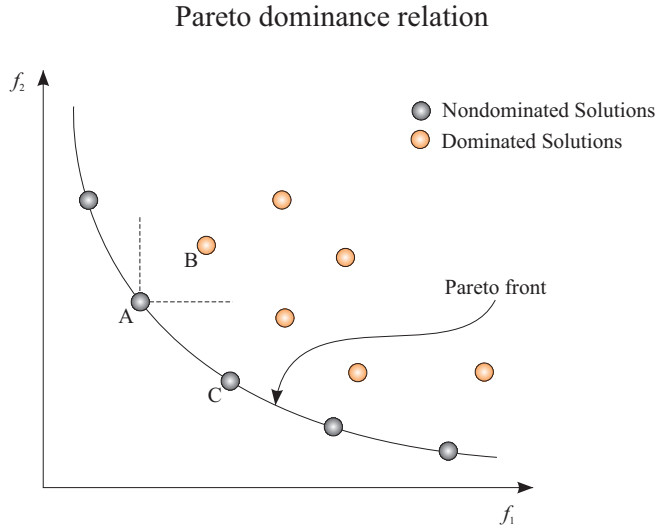


Figure 3.2.: Solution A dominates solution B, however, solution A does not dominate solution C.

3.2 Multi-Objective Mathematical Programming Techniques

Multi-objective mathematical programming techniques as well as *Multi-Criteria Decision Making (MCDM)* techniques, are commonly classified based on how and when they incorporate preferences from the *Decision Maker (DM)* into the search process. A very important issue is the moment at which the DM is required to provide preference information. Cohon and Marks [16] propose a classification, which has been the most popular in the *Operations Research (OR)* community for many years. This taxonomy is presented below.

A Priori Approaches. The DM defines the importance of the objectives before starting the search.

A Posteriori Approaches. The optimizer produces nondominated solutions and then the DM chooses the most preferred one(s) according to his/her preferences.

Interactive approaches. The optimizer produces solutions and the DM progressively provides preference information so that the most preferred solutions can be found.

However, other classifications are also possible, e. g., the one presented by Duckstein [33]. For the purposes of this thesis we shall adopt the proposal made by Cohon and Marks, because their classification is focused on the problems of search and decision making. In the following, we present a brief description of the most popular MCDM techniques according to the above classification. Some of these methods are referred to in this work.

3.2.1 A Priori Preference Articulation

GOAL PROGRAMMING. Charnes and Cooper [11] are credited with the development of the goal programming method for a linear model. In this method, the DM has to assign targets or goals that wishes to achieve for each objective. These values are incorporated into the problem as additional constraints. The objective function then tries to minimize the absolute deviations from the targets to the objectives. The simplest form of this method may be formulated as follows:

$$\begin{aligned} &\text{Minimize:} \quad g(\mathbf{x}) = \sum_{i=1}^{k} |f_i(\mathbf{x}) - z_i^\star| \\ &\text{subject to:} \quad \mathbf{x} \in \Omega, \end{aligned} \tag{3.4}$$

where $z_i^\star$ denotes the target or goal set by the decision maker for the i^{th} objective function $f_i(\mathbf{x})$, and Ω represents the feasible region. The criterion, then is to minimize the sum of the absolute values of the differences between the target values and the achieved ones.

LEXICOGRAPHIC METHOD. In this method, the objectives are ranked in order of importance by the decision maker (from best to worst). The optimum solution $\mathbf{x}^\star$ is then obtained by minimizing the objective functions separately, starting with the most important one and proceeding according to the order of importance of the objectives. Additionally, the optimal value found of each objective is added as a constraint for subsequent optimizations. This way, it is preserved the optimal value of the most important objectives. Only in the case of several optimal solutions in the current objective, the rest of the objectives are considered. Let

the subscripts of the objectives indicate not only the objective function number, but also the priority of the objective. Thus, $f_1(\mathbf{x})$ and $f_k(\mathbf{x})$ denote the most and least important objective functions, respectively. Then, the first problem is formulated as:

$$\begin{aligned} &\text{Minimize:} && f_1(\mathbf{x}) \\ &\text{subject to:} && \mathbf{x} \in \Omega, \end{aligned} \tag{3.5}$$

and its solution $\mathbf{x}_1^\star$ and $f_1^\star = f(\mathbf{x}_1^\star)$ is obtained.

This procedure is repeated until all k objectives have been considered, or a single optimal solution is obtained for the current objective. In the latter case, the solution found is the solution of the original problem. In the former case, we have to continue the optimization process with the problem given by

$$\begin{aligned} &\text{Minimize:} && f_i(\mathbf{x}) \\ &\text{subject to:} && \mathbf{x} \in \Omega, \\ & && f_l(\mathbf{x}) = f_l^\star; \qquad l = 1, \ldots, i-1. \end{aligned} \tag{3.6}$$

If several optimal solutions were obtained in each optimization subproblem, then the solution obtained, i.e., $\mathbf{x}_k^\star$, is taken as the desired solution of the original problem. More details of this method can be found in [96, 35].

3.2.2 A Posteriori Preference Articulation

TCHEBYCHEFF APPROACH. This approach transforms the vector of function values into a scalar optimization problem which is in the form:

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^\star) = \min_{1 \leqslant i \leqslant k} \{w_i |f_i(\mathbf{x}) - z_i|\} \tag{3.7}$$

where $\mathbf{x} \in \Omega$, $\mathbf{z}^\star = (z_1, \ldots, z_k)^T$, such that: $z_i = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}$ and $\mathbf{w}$ is a weight vector, i. e., $\sum_{i=1}^{k} w_i = 1$ and $w_i \geqslant 0$.

For each Pareto optimal point $\mathbf{x}^\star$ there exists a weighting vector $\mathbf{w}$ such that $\mathbf{x}^\star$ is the optimal solution of (3.7) and each optimal solution of (3.7) is a Pareto optimal solution of (3.1). Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector $\mathbf{w}$. One weakness of this approach is that

its aggregation function is not smooth for a continuous MOP. It is worth noticing that, in general, there exist many scalarization methods that transform the MOP into a single-objective optimization problem, which can lead to a reasonably good approximation of the entire Pareto front.

NORMAL BOUNDARY INTERSECTION. Das and Dennis [19] proposed this novel method for generating evenly distributed Pareto optimal solutions. The main idea in the *Normal Boundary Intersection (NBI)* method, is to intersect the feasible objective region with a normal to the convex combinations of the columns of the *pay-off* matrix. For understanding this method let's consider the next definition.

Definition 3.6 (Convex Hull of Individual Minima)
Let $x_i^\star$ be the respective global minimizers of $f_i(\mathbf{x})$, $i = 1, \ldots, k$ over $\mathbf{x} \in \Omega$. Let $\mathbf{F}_i^\star = \mathbf{F}(x_i^\star)$, $i = 1, \ldots, k$. Let Φ be the $k \times k$ matrix whose i^{th} column is $\mathbf{F}_i^* - \mathbf{F}^*$, which is sometimes known as the pay-off matrix. Then the set of points in $\mathbb{R}^k$ that are convex combinations of $\mathbf{F}_i^\star - \mathbf{F}^\star$, i.e. $\{\Phi\beta : \beta \in \mathbb{R}^k, \sum_{i=1}^k \beta_i = 1, \beta_i \geqslant 0\}$, is referred to as the *Convex Hull of Individual Minima (CHIM)*.

The set the attainable objective vectors, $\{\mathbf{F}(\mathbf{x}) : \mathbf{x} \in \Omega\}$ is denoted by $\mathcal{F}$, thus Ω is mapped onto $\mathcal{F}$ by $\mathbf{F}$. The space $\mathbb{R}^k$ which contains $\mathcal{F}$ is referred to as the *objective space*. The boundary of $\mathcal{F}$ is denoted by $\partial\mathcal{F}$. The NBI method can be mathematically formulated as follows.

Given a weighted vector β, $\Phi\beta$ represents a point in the CHIM. Let $\hat{n}$ denote the unit normal to the CHIM simplex towards the origin; then $\Phi\beta + t\hat{n}$ represents the set of points on that normal. The point of intersection of the normal and the boundary of $\mathcal{F}$ closest to the origin is the global solution of the following problem:

$$\begin{aligned} &\text{Maximize:} && t \\ &\text{subject to:} && \Phi\beta + t\hat{n} = \mathbf{F}(\mathbf{x}), \\ & && \mathbf{x} \in \Omega \end{aligned} \tag{3.8}$$

The vector $\Phi\beta + t\hat{n} = \mathbf{F}(\mathbf{x})$ ensures that the point $\mathbf{x}$ is actually mapped by $\mathbf{F}$ to a point on the normal, while the remaining

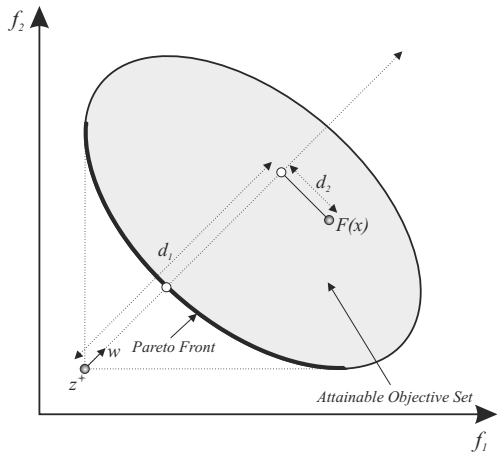


Figure 3.3.: Illustration of the *Penalty Boundary Intersection* (PBI) approach

constraints ensure feasibility of $\mathbf{x}$ in Ω. This approach considers that the shadow minimum $\mathbf{F}^\star$ is in the origin. Otherwise, the first set of constraints should be $\Phi\beta + t\hat{n} = \mathbf{F}(\mathbf{x}) - \mathbf{F}^\star$.

As many scalarization methods, for various β, a number of points on the boundary of $\mathcal{F}$ are obtained thus, effectively, constructing the Pareto surface.

The PBI approach is based on the Das and Dennis method (the NBI method)

Penalty Boundary Intersection Approach. The *Penalty Boundary Intersection (PBI)* approach was proposed by Zhang and Li [155]. This approach uses a weight vector $\mathbf{w}$ and a penalty value θ for minimizing both the distance to the utopian vector d_1 and the direction error to the weight vector d_2 from the solution $\mathbf{F}(\mathbf{x})$, see Fig. 3.3. The optimization problem is formulated as:

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^\star) = d_1 + \theta d_2 \tag{3.9}$$

where

$$d_1 = \frac{||(\mathbf{F}(\mathbf{x}) - z^\star)^T \mathbf{w}||}{||\mathbf{w}||}$$
$$\text{and} \quad d_2 = \left|\left|(\mathbf{F}(\mathbf{x}) - \mathbf{z}^\star) - d_1 \frac{\mathbf{w}}{||\mathbf{w}||}\right|\right|$$

as the Tchebycheff approach, $\mathbf{x} \in \Omega$, $\mathbf{z}^\star = (z_1, \ldots, z_k)^T$, such that: $z_i = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}$ and $\mathbf{w} = (w_1, \ldots, w_k)^T$ is a weight vector, i. e., $\sum_{i=1}^{k} w_i = 1$ and $w_i \geqslant 0$.

3.2.3 Interactive Preference Articulation

METHOD OF GEOFFRION-DYER-FEINBERG. The Geoffrion et al. method [46] is an interactive algorithm based on the maximization of a value function (utility function) using a gradient-based mathematical programming method. The value function is only implicitly known, but is assumed to be differentiable and concave. The gradient-based method employed is the Frank and Wolfe method [45], however, as pointed out by the authors, another approach could be also used in the interactive method. The Frank and Wolfe method assumes that the feasible set, $\Omega \subseteq \mathbb{R}^n$, is compact and convex. The direction-finding problem of the Frank and Wolfe method is the following:

$$\begin{aligned} &\text{Maximize:} && \nabla_x U(\mathbf{F}(\mathbf{x}^h)) \cdot \mathbf{y} \\ &\text{subject to:} && \mathbf{y} \in \Omega, \end{aligned} \tag{3.10}$$

where $U : \mathbb{R}^k \to \mathbb{R}$ is the value function, $\mathbf{x}^h$ is the current point, and $\mathbf{y}$ is the new variable of the problem. Using the chain rule, we obtain

$$\nabla_x U(\mathbf{F}(\mathbf{x}^h)) = \sum_{i=1}^{k} \left(\frac{\partial U}{\partial f_i} \right) \nabla_x f_i(\mathbf{x}^h). \tag{3.11}$$

Dividing this equation by $\frac{\partial U}{\partial f_1}$ we obtain the following reformulation of the Frank and Wolfe problem:

$$\begin{aligned} &\text{Maximize:} && \left(\sum_{i=1}^{k} -m_i^h \nabla_x f_i(\mathbf{x}^h) \right) \cdot \mathbf{y} \\ &\text{subject to:} && \mathbf{y} \in \Omega, \end{aligned} \tag{3.12}$$

where $m_i^h = (\partial U/\partial f_i)/(\partial U/\partial f_1)$ for all $i = 1, \ldots, k$, $i \neq 1$ are the marginal rates of substitution (or indifference trade-off) at $\mathbf{x}^h$ between objectives f_1 and f_i. The marginal rate of substitution is the amount of loss on objective f_i that the decision maker is willing to tolerate in exchange of one unit of gain in objective f_1, while the values of the other objectives remain unchanged.

LIGHT BEAM SEARCH. The Light Beam Search method was proposed by Jaszkiewicz and Slowinski [68], and is an iterative method which combines the reference point idea and tools of *Multi-Attribute Decision Analysis* (MADA). At each iteration, a finite sample of nondominated points is generated. The sample is composed of a current point called *middle point*, which is obtained in the previous iteration, and J nondominated points from its neighborhood. A local preference model in the form of an *outranking relation* S is used to define the neighborhood of the middle point. It is said that a outranks b (aSb), if a is considered to be at least as good as b. The outranking relations is defined by the DM, who specifies three preference thresholds for each objective. They are 1) *indifference threshold*, 2) *preference threshold* and 3) *veto threshold*. The DM has the possibility to scan the inner area of the neighborhood along the objective function trajectories between any two characteristic neighbors or between a characteristic neighbor and the middle point.

3.3 Multi-Objective Evolutionary Algorithms

Traditional multi-objective programming methods are a small subset of a large variety of methods—see for example [35, 58, 96, 138]—available to solve MOPs. However, some researchers [13, 23, 42, 95] have identified several limitations of traditional mathematical programming approaches to solve MOPs. Some of them are the following:

1. It is necessary to run many times those algorithms to find several elements of the Pareto optimal set.
2. Many of them require domain knowledge about the problem to be solved.
3. Some of those algorithms are sensitive to the shape or continuity of the Pareto front.

These complexities call for alternative approaches to deal with certain types of MOPs. Among these alternative approaches, we can find *Evolutionary Algorithms (EAs)*, which are stochastic search and optimization methods that simulate the natural evolution process.

In the late 1960s, Rosenberg [114] proposed the use of genetic algorithms to solve MOPs. However, it was until 1984, when Schaffer [117] introduced the first actual implementation of what it is now called a *Multi-Objective Evolutionary Algorithm (MOEA)*. Since then, many researchers [15, 159, 27, 43, 61, 76, 155] have developed a wide variety of MOEAs. MOEAs are particularly well-suited to solve MOPs because they operate over a set of potential solutions (they are based on a population). This feature allows them to generate several elements of the Pareto optimal set (or a good approximation of them) in a single run. Furthermore, MOEAs are less susceptible to the shape or continuity of the Pareto front than traditional mathematical programming techniques, require little domain information and are relatively easy to implement and use. As pointed out by different authors [161, 14], finding an approximation to the $\mathcal{PF}$ is, by itself, a bi-objective problem whose objectives are:

1. to minimize the distance of the generated solutions to the $\mathcal{PF}$, and
2. to maximize the diversity among the solutions in the Pareto front approximation as much as possible.

Therefore, the fitness assignment scheme that we adopt in a MOEA must consider these two objectives. Based on their selection mechanism, MOEAs can be classified in different ways. In the following, we present the most popular MOEAs developed over the years, some of which are referred to in this thesis.

3.3.1 MOEAs based on a population

The nature of MOEAs (based on a population) and their flexible selection mechanisms have proved to be extremely useful and successful for solving MOPs [14]. The two factors that make the use of a population in MOEAs very practical are: 1) the simultaneous operation on multiple solutions transforms the search for optimal solutions into a cooperative process and hence increases the convergence speed. 2) the Pareto dominance scheme used by most MOEAs makes it possible to tackle the problems as well as to assess candidate solutions to such problems without requiring the aggregation of noncommensurable objectives.

VECTOR EVALUATED GENETIC ALGORITHM. The *Vector Evaluated Genetic Algorithm (VEGA)* is the first actual implementation of a multi-objective evolutionary optimizer which was introduced by Schaffer in 1985 [118]. VEGA is an extension of the well-known GENESIS [53] program. This evolutionary approach operates by dividing the population of solutions into N equally sized subpopulations at every generation. Each subpopulation is designed to separately optimize only one of the N objectives. In other words, the selection of the fittest individuals in each subpopulation is based on a single objective of the problem. After performing selection, individuals are shuffled and then they are recombined and mutated. The main problem in VEGA is the selection procedure, which can hardly produce compromise solutions among all the objectives. Most of the solutions found by VEGA are in the extreme parts of the Pareto front.

3.3.2 MOEAs based on Pareto

MULTI-OBJECTIVE GENETIC ALGORITHM. Fonseca and Fleming [43] proposed the *Multi-Objective Genetic Algorithm (MOGA)* which is based on the ranking scheme proposed by Goldberg [52]. This algorithm ranks the population based on nondominance. Thus, the rank of an individual $\mathbf{x}^i$ at generation t is equal to the number of solutions, $p(\mathbf{x}^i, t)$, by which it is dominated, namely $\text{rank}(\mathbf{x}^i, t) = 1 + p(\mathbf{x}^i, t)$. Then, fitness is computed using, for example: $\text{fitness} = \frac{1}{\text{rank}(\mathbf{x}^i, t)}$, so that all nondominated solutions get the same fitness and all dominated individuals get a fitness value that decreases proportionally to the number of solutions that dominate it.

NON-DOMINATED SORTING GENETIC ALGORITHM. Goldberg's ranking scheme was implemented by Srinivas and Deb [128] in a more straightforward way. The *Non-dominated Sorting Genetic Algorithm (NSGA)* ranks the population in different nondominated layers or fronts with respect to nondominance. The first front (the best ranked) is composed by the nondominated individuals of the current population. The second front is the set composed of the nondominated individuals excluding individuals in the first rank. In general, each front is computed

only using the unranked individuals in the population. An improved version of the NSGA algorithm, called *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* was proposed by Deb et al. [27]. The NSGA-II builds a population of competing individuals, ranks and sorts each individual according to its nondomination level, it applies evolutionary operators to create a new offspring pool, and then combines the parents and offspring before partitioning the new combined pool into fronts. For each ranking level, a crowding distance is estimated by calculating the sum of the Euclidean distances between the two neighboring solutions from either side of the solution along each of the objectives.

Once the nodomination rank and the crowding distance is calculated, the next population is stated by using the crowded-comparison operator ($\prec_n$). The crowded-comparison operator guides the selection process at the various stages of the algorithm toward a uniformly spread-out $\mathcal{PS}$. Assuming that every individual in the population has two attributes: 1) nondomination rank (i_{rank}) and 2) crowding distance ($i_{distance}$), the partial order $\prec_n$ is defined as:

$$i \prec_n j : \begin{array}{l} \text{if } (i_{rank} < j_{rank}) \text{or} \\ ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})) \end{array} \tag{3.13}$$

That is, between two solutions with differing nondomination ranks, it is preferred the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then the solution that is located in a less crowded region is preferred. Algorithm 6 presents the outline of the NSGA-II, which (in the last decade) has been the most popular MOEA, and it is frequently adopted to compare the performance of newly introduced MOEAs.

Strength Pareto Evolutionary Algorithm. The *Strength Pareto Evolutionary Algorithm (SPEA)* was introduced by Zitzler and Thiele [161]. This evolutionary approach integrates some successful mechanisms from other MOEAs, namely, a secondary population (external archive) and the use of Pareto dominance

Algorithm 6: General Framework of NSGA-II

Input:
N: the population size;
T_{max}: the maximum number of generations;
Output:
A: the final approximation to the Pareto optimal front;

```
1  begin
2      t = 0;
3      Generate a random population P_t of size N;
4      Evaluate the population P_t;
5      while t < T_max do
6          Generate the offspring population Q_t by using binary
           tournament and genetic operators (crossover and
           mutation);
7          Evaluate the offspring population Q_t;
8          R_t = P_t ∪ Q_t;
9          Rank R_t by using nondominated sorting to define ℱ;
10         // ℱ = (ℱ_1, ℱ_2, ...), all nondominated fronts of R_t
11         P_{t+1} = ∅ and i = 1;
12         while (|P_{t+1}| + |ℱ_i| ⩽ N) do
13             Assign crowding distance to each front ℱ_i;
14             P_{t+1} = P_{t+1} ∪ ℱ_i;
15             i = i + 1;
16         end
17         Sort ℱ_i by using the crowded-comparison operator;
18         P_{t+1} = P_{t+1} ∪ ℱ_i[1 : (N − |P_{t+1}|)];
19         t = t + 1;
20     end
21     A = P_t;
22 end
```

ranking. SPEA uses an external archive containing nondominated solutions previously found. At each generation, nondominated individuals are copied to the external nondominated set. In SPEA, the fitness of each individual in the primary population is computed using the individuals of the external archive. First, for each individual in this external set, a strength value

is computed. The strength, $S(i)$, of individual i is determined by $S(i) = \frac{n}{\overline{N}+1}$, where n is the number of solutions dominated by i, and $\overline{N}$ is the size of the archive. Finally, the fitness of each individual in the primary population is equal to the sum of the strengths of all the external members that dominate it. Zitzler et al. introduced a revised version of SPEA called *Strength Pareto Evolutionary Algorithm 2 (SPEA2)* [159]. SPEA2 has three main differences with respect to its predecessor: 1) it incorporates a fine-grained fitness assignment strategy which takes into account, for each individual, the number of individuals that dominate it and the number of individuals to which it dominates; 2) it uses a nearest neighbor density estimation technique which guides the search more efficiently, and it has an enhanced archive truncation method that guarantees the preservation of boundary solutions. The outline of the SPEA2 is shown in Algorithm 7.

Let's consider $\overline{P}_t$ and P_t as the external archive and the population at generation t, respectively. Each individual i in the external archive $\overline{P}_t$ and the population P_t is assigned a strength value $S(i)$, representing the number of solutions it dominates: $S(i) = |j|j \in P_t + \overline{P}_t \wedge i \prec j|$, where $|\cdot|$ denotes the cardinality of a set, $+$ stands for multi-set union and the symbol $\prec$ corresponds to the Pareto dominance relation. On the basis the S values, the raw fitness $R(i)$ of an individual i is calculated by:

$$R(i) = \sum_{j \in P \cup \overline{P}, j \prec i} S(j)$$

SPEA2 incorporates additional density information to discriminate between individuals having identical raw fitness values. The density estimation technique used in SPEA2 is an adaptation of the k-th nearest neighbor method, and it is calculated by:

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

where, $k = \sqrt{|N| + |\overline{N}|}$, and σ_i^k denotes the distance from i to its k-th nearest neighbor in $P_t + \overline{P}_t$. Finally, the fitness value $F(i)$ of an individual i is calculated by:

$$F(i) = R(i) + D(i) \tag{3.14}$$

During environmental selection, the first step is to copy all nondominated individuals. If the nondominated front fits exactly into the archive ($|\overline{P}_{t+1}| = N$) the environmental selection step is completed. Otherwise, there can be two situations: Either the archive is too small ($|\overline{P}_{t+1}| < N$) or too large ($|\overline{P}_{t+1}| > N$). In the first case, the best $\overline{N} - |\overline{P}_{t+1}|$ dominated individuals in the previous archive and population are copied to the new archive. In the second case, when the size of the current nondominated (multi) set exceeds $\overline{N}$, an archive truncation procedure is invoked which iteratively removes individuals from $\overline{P}_{t+1}$ until $|\overline{P}_{t+1}| = \overline{N}$.

3.3.3 MOEAs based on Decomposition

MOEA BASED ON DECOMPOSITION. The *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)* was introduced by Zhang and Li [155]. MOEA/D explicitly decomposes the MOP into scalar optimization subproblems. It is well-known that a Pareto optimal solution to a MOP, under certain conditions, could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the functions f_i's. Therefore, an approximation of the Pareto optimal front can be decomposed into a number of scalar objective optimization subproblems. This is a basic idea behind many traditional mathematical programming methods for approximating the Pareto optimal front. Several methods for constructing aggregation functions can be found in [35, 96, 138]. This basic idea of decomposition is used by MOEA/D, and it solves these subproblems simultaneously by evolving a population of solutions. At each generation, the population is composed of the best solution found so far (i.e. since the start of the run of the algorithm) for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions to two neighboring subproblems should be very similar. Each subproblem (i.e., each scalar aggregation function) is optimized in MOEA/D by using information only from its neighboring subproblems. To obtain a good representation of the Pareto

Algorithm 7: General Framework of SPEA2

Input:
N: the population size;
$\overline{N}$: the archive size;
T_{max}: the maximum number of generations;
Output:
A: the final approximation to the Pareto optimal front.

begin
  $t = 0$;
  Generate a random population P_t of size N;
  $\overline{P}_t = \emptyset$; `// the external archive`
  while $(t < T_{max})$ **do**
    Calculate the fitness values of individuals in P_t and $\overline{P}_t$;
    Copy all nondominated individuals in P_t and $\overline{P}_t$ to P_{t+1}. If size of P_{t+1} exceeds $\overline{N}$ then reduce P_{t+1} by means of the truncation operator, otherwise if size of P_{t+1} is less than $\overline{N}$ then fill P_{t+1} with dominated individuals in P_t and $\overline{P}_t$;
    if $(t + 1 < T_{max})$ **then**
      Perform binary tournament selection with replacement on $\overline{P}_{t+1}$ in order to fill the mating pool;
      Apply recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population.;
    end
    $t = t + 1$;
  end
  Set A as the set of decision vectors represented by the nondominated individuals in $\overline{P}_t$;
end

optimal front, a set of evenly spread weighting vectors needs to be previously generated.

Considering N as the number of scalar optimization subproblems and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ as the set of weighting vectors which defines such subproblems, MOEA/D finds the best solution to each subproblem along the evolutionary process. Assuming the Tchebycheff approach (3.7), the fitness function of

the i^{th} subproblem is stated by $g(\mathbf{x}|\mathbf{w}_i, \mathbf{z})$. MOEA/D defines a neighborhood of each weighting vector $\mathbf{w}_i$ as a set of its closest weighting vectors in W. Therefore, the neighborhood of the i^{th} subproblem consists of all the subproblems with the weighting vectors from the neighborhood of $\mathbf{w}_i$ and it is denoted by $B(\mathbf{w}_i)$. The genetic operators (mutation and crossover) in MOEA/D are performed between pair of individuals in each neighborhood $B(\mathbf{w}_i)$. At each generation, MOEA/D maintains:

1. A population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N points, where $\mathbf{x}_i \in \Omega$ is the current solution to the i^{th} subproblem;
2. $FV^1, \ldots, FV^N$, where FV^i is the $\mathbf{F}$-value of $\mathbf{x}_i$, i.e., $FV^i = \mathbf{F}(\mathbf{x}_i)$ for each $i = 1, \ldots, N$;
3. an external archive EP, which is used to store the nondominated solutions found during the search.

In contrast to NSGA-II and SPEA2 which use density estimators (crowding distance and neighboring solutions, respectively), MOEA/D uses the well-distributed set of weight vectors W for guiding the search, and therefore, multiple solutions along the $\mathcal{PF}$ are maintained. With that, the diversity in the population of MOEA/D is implicitly maintained. For an easy interpretation of MOEA/D, it is outlined in Algorithm 8, in page 44. Nowadays, several authors have taken the idea behind MOEA/D for developing current state-of-the-art MOEAs based on decomposition, see for example those presented in [107, 98, 149].

3.4 Performance Assessment

As pointed before, since their origins, MOEAs have attempted to satisfy the two main goals of multi-objective optimization: 1) minimize the distance of the generated solutions to the $\mathcal{PF}$, and 2) maximize the diversity among the solutions in the Pareto front approximation as much as possible. Therefore, to assess the performance of a MOEA, the two above issues should be considered. In order to allow a quantitative comparison of results among the different algorithms presented here, we adopted the performance measures that are briefly described next.

HYPERVOLUME. The *Hypervolume* (I_H) performance measure was proposed by Zitzler [160]. This performance measure is Pareto compliant [162] and quantifies both convergence and spread of nondominated solutions along the $\mathcal{PF}$. The hypervolume corresponds to the non-overlapped volume of all the hypercubes formed by a reference point $\mathbf{r}$ (given by the user) and each solution p in the Pareto front approximation. The I_H performance measure (also known as $\mathcal{S}$-metric) can be defined as follows.

Definition 3.7 (Hypervolume)
Let P be a Pareto front approximation given by an algorithm. The I_H performance measure of P is calculated as:

$$I_H(P) = \Lambda\left(\bigcup_{\mathbf{p}\in P}\{\mathbf{x}|\mathbf{p} \prec \mathbf{x} \prec \mathbf{r}\}\right) \tag{3.15}$$

where Λ denotes the Lebesgue measure and $\mathbf{r} \in \mathbb{R}^k$ denotes a reference vector being dominated by all valid candidate solutions.

A high I_H value, indicates that the approximation P is close to $\mathcal{PF}$ and has a good spread towards the extreme portions of the $\mathcal{PF}$.

TWO SET COVERAGE. The *Two Set Coverage* (I_C) metric was proposed by Zitzler et al. [158]. This performance measure compares a set of non-dominated solutions A with respect to another set B, using Pareto dominance. The I_C metric is mathematically defined as follows:

Definition 3.8 (Two Set Coverage)
Let A and B be two sets of decision variables. The function I_C maps the ordered pair (A, B) to the interval $[0, 1]$, according to:

$$I_C(A, B) = \frac{|\{\mathbf{b} \in B|\exists \mathbf{a} \in A : \mathbf{a} \preceq \mathbf{b}\}|}{|B|} \tag{3.16}$$

where $\preceq$ defines the Pareto dominance relation.

The value $I_C(A, B) = 1$ means that all solutions in B are dominated by or are equal to the solutions in A. The opposite,

$I_C(A, B) = 0$, represents the situation when none of the solutions in B are covered by the set A. Note that both $I_C(A, B)$ and $I_C(B, A)$ have to be considered, since $I_C(A, B)$ is not necessarily equal to $1 - I_C(B, A)$. Nonetheless, we can say that A is better than B, if and only if, $I_C(A, B) = 1$ and $I_C(B, A) = 0$.

3.5 Test functions

In order to challenge the search capabilities of the MOEAs proposed in this thesis, a set of multi-objective test functions were adopted. In the specialized literature, there are several artificial test problems to evaluate the abilities of a MOEA—see for example [158, 28, 27, 63, 154]. These test functions encompass specific features, such as: multimodality, non-convexity and discontinuity. Some test problems have a disconnected and/or asymmetric $\mathcal{PF}$ in two and three objective functions. Such features are known to generally cause several difficulties to most MOEAs to reach all the regions in the $\mathcal{PF}$. In the following, we briefly describe the test suite that we have adopted for the purposes of this thesis.

Classic test problems. We have adopted nine test problems from different authors such as: Deb [24], Fonseca and Fleming [44], Laumanns [84], Lis and Eiben [87], Murata and Ishibuchi [101], Valenzuela and Uresti [135], Viennet et al. [137]. These problems are characterized by using a low number of decision variables and they are considered here as classic MOPs, since they were proposed before 2000. The mathematical description of these test problems is presented in Appendix A.1.

Zitzler-Deb-Thiele test suite. Zitzler et al. [158] proposed a set of test functions which are known as the *Zitzler-Deb-Thiele (ZDT)* test suite. In our study, we adopt five bi-objective MOPs (ZDT5 is not adopted because it is a discrete problem) from this test suite. The description of the ZDT test problems is presented in Appendix A.2.

Deb-Thiele-Laumanns-Zitzler test suite. Deb et al. [28, 29] proposed a set of test functions for testing and comparing MOEAs. This set of problems, known as the *Deb-Thiele-Laumanns-Zitzler*

(DTLZ) test suite, attempts to define generic multi-objective test problems that are scalable to a user-defined number of objectives. An increase in dimensionality of the objective space also causes a large portion of a randomly generated initial population to be nondominated to each other, thereby reducing the effect of the selection operator in a MOEA. In our study, we have adopted the seven unconstrained MOPs (DTLZ1–DTLZ7) from this test suite. The mathematical formulation of these problems is shown in Appendix A.3.

WALKING-FISH-GROUP TEST SUITE. Huband et al. [63] proposed a set of test functions which are known as the *Walking-Fish-Group (WFG)* test suite. These MOPs are generalized to be scaled in the number of objective functions. In each problem, a set of sequential transformations to the vector of decision variables is applied. This strategy is used to increase the difficulty of the problem. Therefore, the WFG test suite constitutes a set of difficult test problems to solve, in comparison with both the ZDT and DTLZ test suites. The description of the nine WFG test problems is presented in Appendix A.4.

Algorithm 8: General Framework of MOEA/D

Input:
a stopping criterion;
N: the number of the subproblems considered in MOEA/D;
W: a well-distributed set of weighting vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$;
T: the number of weight vectors in the neighborhood of each weighting vector.
Output:
EP: the nondominated solutions found during the search;
P: the final population found by MOEA/D.

begin
  Step 1. Initialization:
  $EP = \emptyset$;
  Generate an initial population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ randomly;
  $FV^i = \mathbf{F}(\mathbf{x}_i)$;
  $B(\mathbf{w}_i) = \{\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}\}$ where $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}$ are the T closest weighting vectors to $\mathbf{w}_i$, for each $i = 1, \ldots, N$;
  $\mathbf{z} = (+\infty, \ldots, +\infty)^T$;
  while *stopping criterion is not satisfied* **do**
    Step 2. Update: (the next population)
    for $\mathbf{x}_i \in P$ **do**
      Reproduction: Randomly select two indexes k, l from $B(\mathbf{w}_i)$, and then generate a new solution $\mathbf{y}$ from $\mathbf{x}_k$ and $\mathbf{x}_l$ by using genetic operators.
      Mutation: Apply a mutation operator on $\mathbf{y}$ to produce $\mathbf{y}'$.
      Update of $\mathbf{z}$: For each $j = 1, \ldots, k$, if $z_j < f_j(\mathbf{x})$, then set $z_j = f_j(\mathbf{y}')$.
      Update of Neighboring Solutions: For each index $j \in B(\mathbf{w}_i)$, if $g(\mathbf{y}'|\mathbf{w}_j, \mathbf{z}) \leqslant g(\mathbf{x}_i|\mathbf{w}_j, \mathbf{z})$, then set $\mathbf{x}_j = \mathbf{y}'$ and $FV^j = \mathbf{F}(\mathbf{y}')$.
      Update of EP: Remove from EP all the vectors dominated by $\mathbf{F}(\mathbf{y}')$. Add $\mathbf{F}(\mathbf{y}')$ to EP if no vectors in EP dominate $\mathbf{F}(\mathbf{y}')$.
    end
  end
end

4

Multi-Objective Memetic Algorithms Based on Direct Search Methods

Mathematical programming techniques for solving optimization problems have shown to be an effective tool in many domains, at a reasonably low computational cost. However, as we discussed in Chapters 2 and 3 (Sections 2.6 in page 20 and 3.3 in page 32), they have several limitations and therefore, their use is limited to certain types of problems. Over the years, *Evolutionary Algorithms (EAs)* have been found to offer several advantages in comparison with traditional programming methods, including generality (they require little domain information to work) and ease of use. However, they are normally computationally expensive (in terms of the number of objective function evaluations required to obtain optimal solutions), which limits their use in some real-world applications. The characteristics of these two types of approaches have motivated the idea of combining them. Algorithms that combine *Multi-Objective Evolutionary Algorithms (MOEAs)* with an improvement mechanism (normally, a local search engine) are called *Multi-Objective Memetic Algorithms (MOMAs)* [100]. Here, we are interested in developing new MOMAs that combine direct search methods—i. e., mathematical programming methods that do not require gradient information of the functions—with MOEAs.

In the following, we review state-of-the-art MOMAs based on mathematical programming techniques. Particularly, we place special emphasis on the hybridization of MOEAs with direct search methods which is the main focus of this thesis.

4.1 Multi-Objective Memetic Algorithms

Hybridization of MOEAs with local search algorithms has been investigated for more than one decade [75]. Some of the first MOMAs for dealing on discrete domains were presented in [65, 66], including the *Multi-Objective Genetic Local Search (MOGLS)* approach, and the *Pareto Memetic Algorithm (PMA)* presented in [67]. These two approaches use scalarization functions to approximate solutions to the *Pareto front ($\mathcal{PF}$)*. Another proposal employing Pareto ranking selection was called *Memetic Pareto Archived Evolution Strategy (M-PAES)* [73]. Also, in [100], the authors proposed a local search procedure with a generalized replacement rule based on the dominance relation. In [9], the *Cross Dominant Multi-Objective Memetic Algorithm (CDMOMA)* was proposed as an adaptation of the *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*, and two local search engines: a multi-objective implementation of Rosenbrock's algorithm [115], which performs very small movements, and *Pareto Domination Multi-Objective Simulated Annealing (PDMOSA)* [130], which performs a more global exploration. A memetic version of a *Coevolutionary Multi-Objective Differential Evolution (CMODE)* was proposed in [125] and was named CMODE-MEM. Most of this mentioned work has been proposed for combinatorial problems. For the continuous case—i. e., continuous objectives defined on a continuous domain—the first attempts started, to the author's best knowledge, with the hybrid algorithm presented in [50], where a neighborhood search was applied to the NSGA-II. This is a very simple scheme and the authors found that the added computational work had a severe impact on the efficiency of the algorithm. Since then, a significant amount of work related to the development of hybrid algorithms to deal with continuous problems has been explored by several researchers, see for example [12, 47, 51, 62, 78, 79, 83, 88]. The above mentioned hybrid algorithms use mathematical programming techniques as their local search procedures and such procedures are applied during the evolutionary process of a MOEA. However, several authors have used some traditional mathematical programming techniques in different ways to guide the search of MOEAs, see for example [98, 107, 147, 149, 155]. The main goal of such hybridizations is to improve the performance of MOEAs, by accelerating the convergence to the *Pareto optimal set ($\mathcal{PS}$)* and maintaining a good representation of the $\mathcal{PF}$. In our study, we

are interested in developing MOMAs that incorporate direct search methods as a local search mechanism in the evolutionary process of a MOEA.

In the last few years, the development of MOEAs hybridized with direct search methods has attracted the attention of several researchers. In the following, we present some of these hybrid approaches that have reported improvements with respect to the original MOEA adopted.

4.2 MOMAs Based on Direct Search Methods

4.2.1 A Multi-objective GA-Simplex Hybrid Algorithm

Koduru et al. [78] introduced a hybrid *Genetic Algorithm (GA)* using fuzzy dominance and the *Nonlinear Simplex Search (NSS)* algorithm [102]. The simplex search algorithm is used for improving solutions in the population of a GA. The proposed memetic approach is employed to estimate the parameters of a gene regulatory network for flowering time control in rice. In order to understand the fuzzy dominance relation, the following definitions are introduced. Assuming minimization problems with n decision variables and considering $\Omega \subset \mathbb{R}^n$ as the feasible solution space, fuzzy i-dominance is defined as follows.

Definition 4.1
Given a monotonically nondecreasing function $\mu_i^{dom} : \Omega \to [0, 1], i = \{1, \ldots, n\}$ such that $\mu_i^{dom}(0) = 0$, solution $\mathbf{u} \in \Omega$ is said to **i-dominate** solution $\mathbf{v} \in \Omega$, if and only if $f_i(\mathbf{u}) < f_i(\mathbf{v})$. This relationship will be denoted as $\mathbf{u} \prec_i^F \mathbf{v}$. If $\mathbf{u} \prec_i^F \mathbf{v}$, the degree of fuzzy i-dominance is equal to $\mu_i^{dom}(f_i(\mathbf{v}) - f_i(\mathbf{u})) \equiv \mu_i^{dom}(\mathbf{u} \prec_i^F \mathbf{v})$. Fuzzy dominance can be regarded as a fuzzy relationship $\mathbf{u} \prec_i^F \mathbf{v}$ between $\mathbf{u}$ and $\mathbf{v}$ [94].

Definition 4.2
Solution $\mathbf{u} \in \Omega$ is said to **fuzzy dominate** solution $\mathbf{v} \in \Omega$ if and only if $\forall i \in \{1, \ldots, k\}, \mathbf{u} \prec_i^F \mathbf{v}$. This relationship will be denoted as $\mathbf{u} \prec^F \mathbf{v}$. The degree of fuzzy dominance can be defined by invoking the concept of fuzzy intersection [94]. If $\mathbf{u} \prec^F \mathbf{v}$, the degree of fuzzy dominance $\mu^{dom}(\mathbf{u} \prec^F \mathbf{v})$ is obtained by computing the intersection of the fuzzy relationships $\mathbf{u} \prec_i^F \mathbf{v}$ for each i. The fuzzy intersection

operation is carried out using a family of functions called t-norms, denoted by $\bigcap$. Hence,

$$\mu^{dom}(\mathbf{u} \prec^F \mathbf{v}) = \bigcap_{i=1}^{k} \mu_i^{dom}(\mathbf{u} \prec \mathbf{v}) \tag{4.1}$$

where k is the number of objective functions.

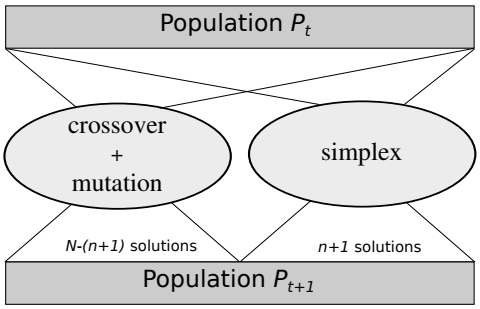


Figure 4.1.: The offspring population generated by the multi-objective GA-Simplex Hybrid Algorithm

Definition 4.3
Given a population of solutions $P \subset \Omega$, a solution $\mathbf{v} \in P$ is said to be fuzzy dominated in P if and only if it is fuzzy dominated by any other solution $\mathbf{u} \in P$. In this case, the degree of fuzzy dominance can be computed by performing a union operation over every possible $\mu^{dom}(\mathbf{u} \prec^F \mathbf{v})$, carried out using t-co norms, that are denoted by $\bigcup$. Hence the degree of fuzzy dominance of a solution $\mathbf{v} \in P$ in the set P is given by,

$$\mu^{dom}(P \prec^F \mathbf{v}) = \bigcup_{\mathbf{u} \in P} \mu^{dom}(\mathbf{u} \prec^F \mathbf{v}) \tag{4.2}$$

In order to calculate the fuzzy dominance relationship between two solution vectors, trapezoidal membership functions are used. Therefore,

$$\mu_i^{dom}(\mathbf{u} \prec_i^F \mathbf{v}) = \begin{cases} 0 & \text{if } f_i(\mathbf{v}) - f_i(\mathbf{u}) < 0, \\ \frac{f_i(\mathbf{v}) - f_i(\mathbf{u})}{p_i} & \text{if } 0 \leqslant f_i(\mathbf{v}) - f_i(\mathbf{u}) < p_i, \\ 1 & \text{otherwise.} \end{cases} \tag{4.3}$$

where p_i determines the length of the linear region of the trapezoid for the objective function f_i. The t-norm and t-co norms are defined as $x \cap y = xy$ and $x \cup y = x + y - xy$. Both are standard forms of operators [94].

At each generation of the MOMA, the fuzzy dominances of all solutions in the current population P_t are calculated according to the equation (4.3). Then, the fuzzy dominances of the population are stored as a two dimensional array, where each entry is a fuzzy dominance relationship between two solution vectors. The hybrid algorithm obtains a part of the offspring population P_{t+1} by using the genetic operators of the GA and the rest is stated by performing the NSS, see Figure 4.1. Thus, the first part of the population is obtained by evolving a set B of $N - (n + 1)$ solutions chosen randomly from the population P_t, where N denotes the population size and n the number of decision variables of the MOP. The subpopulation B is evolved performing genetic operators (crossover and mutation) and the fuzzy dominance relation is used as a measure of fitness during the selection into the GA. The resulting offspring population Q_1 is inserted as part of the next population P_{t+1}. The second part of the population is generated by performing the NSS algorithm. The simplex is built by selecting a sample set S of $n + 1$ solutions from the current population P_t and then, the centroid $\mathbf{c}$ of the sample S is calculated. Any solution $\mathbf{u} \in S$ at a distance $\|\mathbf{c} - \mathbf{u}\| > \rho_{simplex}$ is rejected and replaced with another one taken in a random way from the population P_t, where $\rho_{simplex}$ represents the radius parameter of the simplex and $\|\cdot\|$ denotes the Euclidean norm. This process is repeated until either all the sample solutions fit within the radius $\rho_{simplex}$, or the total replacements exceed r_{max}. After selecting the initial vertices of the simplex, the NSS algorithm is performed during α times. To each solution in the simplex, the fuzzy dominance is calculated considering the solutions of the simplex and the vertices are sorted according to fuzzy dominance relation. From the solutions obtained by the NSS algorithm, a set Q_2 of the best $n + 1$ solutions (according to the fuzzy dominance relation) are selected and they are inserted into the next population P_{t+1}. The evolutionary process of the MOMA is carried out by T_{max} generations. Algorithm 9 shows the general framework of the multi-objective GA-Simplex hybrid algorithm. The authors suggested the use of $\alpha = 10$ as the maximum number of iterations for the NSS algorithm.

The coefficients for the reflection, expansion and contraction movements of the NSS were defined as: $\rho = 1, \chi = 1.5$ and $\gamma = 0.5$, respectively. The NSS algorithm was performed without using the shrinkage step. The hybrid approach was tested using a population size of $N = 100$ and it was was compared against a well-known state-of-the art MOEA, the *Strength Pareto Evolutionary Algorithm (SPEA)*. A more detailed description of this algorithm can be found in [78].

Algorithm 9: The Multi-objective GA-Simplex Hybrid Algorithm

Input:
N: the population size;
T_{max}: the maximum number of generations;
Output:
P: the final approximation to the *Pareto front* ($\mathcal{PF}$);

begin
  $t = 0$;
  Generate a random population P_t of size N;
  Evaluate the population P_t;
  while $t < T_{max}$ **do**
    `// Selecting the solutions for performing the evolutionary process;`
    $B = x_i \in P_t$ such that: $\mathbf{x}_i$ is randomly chosen from P_t and $|P_t| = N - (n + 1)$;
    $Q_1 = \text{MUTATION}(\text{CROSSOVER}(B))$; `// Apply genetic operators`
    $S = \mathbf{x}_i \in P_t$ such that: $|S| = n + 1$; `// Defining the simplex`
    for $j = 0$ **to** $j < \alpha$ **do**
      Perform NSS using the initial simplex S;
    end
    Define Q_2 as the best $n + 1$ solutions (according to fuzzy dominance) found by the simplex search;
    $P_{t+1} = Q_1 \cup Q_2$;
    $t = t + 1$;
  end
  return P_t
end

4.2.2 A Multi-objective Hybrid Particle Swarm Optimization Algorithm

Koduru et al. [79] hybridized a *Particle Swarm Optimization (PSO)* algorithm [71] with the NSS method for dealing with *Multi-objective Optimization Problems (MOPs)*. In this approach, the simplex search is used as a local search engine in order to find nondominated solutions in the neighborhood defined by the solution to be improved. The bio-inspired technique evolves a set of solutions P (called swarm) to approximate solutions to the $\mathcal{PF}$. Each particle $\mathbf{x}_i$ in the swarm possesses a flight velocity which is initially set in zero. The swarm is evolved by updating both the velocity $\mathbf{v}_i^{t+1}$ and the position of each particle $\mathbf{x}_i^{t+1}$ according to the following equations:

$$\mathbf{v}_i^{t+1} = w(\mathbf{v}_i^t + c_1 r_1(\mathbf{x}_{pb,i} - \mathbf{x}_t^t) + c_2 r_2(\mathbf{x}_{gb,i} - \mathbf{x}_i^t)) \tag{4.4}$$

and the new particle position is updated according to the equation [71]:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{4.5}$$

where $w \geqslant 0$ represents the constriction coefficient, $c_1, c_2 \geqslant 0$ are the constraints on the velocity, r_1, r_2 are two random variables having uniform distribution in the range $(0, 1)$. $\mathbf{v}_i, \mathbf{x}_{pb,i}$ and $\mathbf{x}_{gb,i}$ represent the velocity, the personal best and the global best position for the i^{th} particle, respectively.

Since at the beginning, a particle does not have a previous position, the best personal position is initialized with the same position as the particle, i. e., $\mathbf{x}_{pb,i} = \mathbf{x}_i$. To avoid getting stuck in a local minimum a turbulence factor is implemented into the velocity update (Equation (4.4)), which is similar to a mutation operator in GAs. The modified update equation is given by:

$$\mathbf{v}_i^{t+1} = w(\mathbf{v}_i^t + c_1 r_1(\mathbf{x}_{pb,i} - \mathbf{x}_t^t) + c_2 r_2(\mathbf{x}_{gb,i} - \mathbf{x}_i^t)) + \exp(-\delta t) \cdot u \tag{4.6}$$

where δ is the turbulence coefficient and u is a uniformly distributed random number in $[-1, 1]$. The negative exponential term assures that the turbulence in the velocities is higher at the initial generations which promotes more exploration. Later on, the behavior will be more exploitative.

The nondominated solutions found along the evolutionary process are stored in an external archive denoted as A. This set of nondominated solutions is updated along the evolutionary process by selecting the best N solutions from the union between the current population P and the external archive A, according to the fuzzy dominance relation. In a previous implementation of the fuzzy dominance [78], the membership functions $\mu_i^{dom}(\cdot)$ employed to compute the fuzzy i-dominances were defined to be zero for negative arguments. Therefore, whenever $f_i(\mathbf{u}) > f_i(\mathbf{v})$, the degree of fuzzy dominance $\mathbf{u} \prec_i^F \mathbf{v}$ is necessarily zero. In this memetic approach, nonzero values are allowed. The membership functions used are trapezoidal, yielding nonzero values whenever their arguments are to the right of a threshold ε. Mathematically, the memberships $\mu_i^{dom}(\mathbf{u} \prec^F \mathbf{v})$ are defined as:

$$\mu_i^{dom}(\delta f_i) = \begin{cases} 0, & \delta f_i \leqslant -\varepsilon \\ \frac{\delta f_i}{\delta_i}, & -\varepsilon < \delta f_i < \delta_i - \varepsilon \\ 1, & \delta f_i \geqslant \delta_i - \varepsilon \end{cases} \tag{4.7}$$

where $\delta f_i = f_i(\mathbf{v}) - f_i(\mathbf{u})$. Given a population of solutions $P \subset \Omega$, a solution $\mathbf{v} \in P$ is said to be fuzzy dominated in P if and only if it is fuzzy dominated by any other solution $\mathbf{u} \in S$. In this way, each solution can be assigned a single measure to reflect the amount it dominates others in a population. Better solutions within a set will have a lower fuzzy dominance value, although, unlike in [78] nondominated solution may not necessarily be assigned zero values. In order to compare multiple solutions having similar fuzzy dominance values, the crowding distance of NSGA-II is used [27].

Considering a MOP with n decision variables, the set of solutions P is divided into separate clusters, where each cluster consists of proximally located solutions and it is generated by using a variant of the k-means algorithm [90]. The clusters are disjoint, with $n+1$ solutions each. Each cluster defines the simplex from which, NSS is performed. At each iteration of the local search procedure, NSS performs l movements (reflection, expansion or contraction) into the simplex before finishing. The solutions found by NSS are employed to update both the swarm P and the external archive A by using the fuzzy dominance relation. Algorithm 10 shows the general framework of the multi-objective hybrid PSO algorithm.

The authors suggested the use of $k = 9$ as the number of centers for the k-means algorithm, $l = 2$ for the number of movements (reflection, expansion or contraction) into the simplex. The simplex search was tested using $\rho = 1, \chi = 1.5$ and $\gamma = 0.5$ for the reflection, expansion and contraction, respectively. NSS was executed omitting the use of the shrinkage transformation. The population size was set to $N = 100$ and the external archive was limited to 100 as the maximum number of particles. The proposed hybrid algorithm was tested by solving artificial test functions and a molecular genetic model plant problem having between 3 and 10 decision variables and two objective functions. For a more detailed description of this algorithm see [79].

4.2.3 A Nonlinear Simplex Search Genetic Algorithm

Zapotecas and Coello [146] presented a hybridization between the well-known NSGA-II and the NSS algorithm. The proposed *Nonlinear Simplex Search Genetic Algorithm (NSS-GA)* combines the explorative power of NSGA-II with the exploitative power of the NSS algorithm, which acts as a local search engine. The general framework of the proposed MOMA is shown in Algorithm 11. NSS-GA evolves a population P_t by using the genetic operators of the NSGA-II (*Simulated Binary Crossover (SBX)* and *Polynomial-Based Mutation (PBM)*) and then, the local search mechanism is performed. The general idea of the local search procedure is to intensify the search towards better solutions for each objective function and the maximum bungle (sometimes called knee) of the $\mathcal{PF}$. The main goal of the NSS is to obtain the set Λ, which is defined as:

$$\Lambda = \lambda_1 \cup \lambda_2 \cup \cdots \lambda_k \cup \Upsilon$$

where λ_i is a set of the best solutions found for the i-th objective function of the MOP. Υ is a set of the best solutions found by minimizing an aggregating function which approximates solutions to the knee of the $\mathcal{PF}$.

The local search mechanism is performed each $\frac{n}{2}$ generations, where n denotes the number of decision variables of the MOP. Initially, the local search focuses on minimizing separately the objective functions f_i's of the MOP. Once the separate functions are minimized, an

Algorithm 10: The Multi-objective Hybrid PSO Algorithm

Input:
T_{max}: the maximum number of generations;
Output:
A: the final approximation to the *Pareto front* ($\mathcal{PF}$);

```
1  begin
2      t = 0;
3      Generate a set of particles P_t of size N // using an uniform
       distribution;
4      Initialize all velocities v_i^t, to zero;
5      while t < t_max do
6          Evaluate the set of particles P_t;
7          Evaluate the fuzzy dominance in the population P_t
           according to Equation (4.7);
8          Update the archive A;
9          Update each particle x_i ∈ P_t including its personal best
           and global best;
10         Randomly initialize k cluster centers;
11         Assign each particle x_i to a cluster using the k-means
           algorithm;
12         For each cluster apply the simplex search algorithm.;
13         Update the velocities v_i^{t+1} according to Equation (4.6);
14         Update the positions x_i ∈ P_t according to Equation (4.5);
15         t = t + 1;
16     end
17     return P_t
18 end
```

aggregating function is used for approximating solutions to the knee of the $\mathcal{PF}$. The initial search point from which the local search starts is defined according to the next rules:

- Minimizing separate functions. In the population P, the individual $\mathbf{x}_\Delta \in P^\star$ is chosen such that:

$$\mathbf{x}_\Delta = \mathbf{x}_l | \mathbf{x}_l = \min_{\forall \mathbf{x}_l \in P^\star} \{f_i(\mathbf{x}_l)\}$$

where $P^\star$ is a set of nondominated solutions within the population P. In other words, the selected individual is the best nondominated solution for the objective f_i.

- Minimizing the aggregating function. The individual $\mathbf{x}_\Delta \in P^\star$ is chosen such that it minimizes:

$$G(\mathbf{x}_\Delta) = \sum_{i=1}^{k} \frac{|z_i - f_i(\mathbf{x}_\Delta)|}{|z_i|} \tag{4.8}$$

where $\mathbf{z}^\star = (z_1^\star, \dots z_k^\star)^T$ is the utopian vector defined by the minimum values f_i^* of the k objective functions until the current generation. In this way, the local search minimizes the aggregating function defined by:

$$g(\mathbf{x}) = ED(\mathbf{F}(\mathbf{x}), \mathbf{z}^\star) \tag{4.9}$$

where $ED(\cdot)$ is the Euclidean distance between the vector of objective functions $\mathbf{F}(\mathbf{x})$ and the utopian vector $\mathbf{z}^\star$.

The selected solution $\mathbf{x}_\Delta$ is called "simplex-head", which is the first vertex of the n-simplex. The remaining n vertices are created in two phases:

Reducing the Search Domain. A sample of s solutions which minimize the objective function to be optimized is identified, and then, the average ($\mathbf{m}$) and standard deviation (σ) of these decision variables is computed. Based on that information, the new search space is defined as:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma \end{aligned}$$

where $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively. In this work, the authors propose to use $s = 0.20 \times N$, where N is the population size of the evolutionary algorithm—i.e. 20% of the population size.

Building the Vertices. Once the new search domain has been defined, the remaining vertices are determined by using either the Halton [54] or the Hammersley [55] sequence (each has a 50% probability of being selected) in the new bounds $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$, previously defined.

Once the simplex is defined, the NSS algorithm is executed during a determined number of iterations, and it is stopped according to the following stopping criteria. The local search is stopped if: 1) it does not generate a better solution after $n+1$ iterations, or 2) if after performing $2(n+1)$ iterations, the convergence is less than ϵ. Considering Λ as the set of solutions found by the local search mechanism, the gained knowledge is introduced to the population of the NSGA-II by using the crowding comparison operator [27] over the union of the population P and Λ.

The simplex was controlled using $\rho = 1, \chi = 2$ and $\gamma = 0.5$ for the reflection, expansion and contraction coefficients, respectively. The shrinkage step is not employed in this approach. The threshold for the convergence in the simplex search was set to: $\epsilon = 1 \times 10^{-3}$. The hybrid algorithm was tested over artificial test functions having between 10 and 30 decision variables, and two and three objective functions. A more detailed description of this hybrid algorithm can be found in [146].

4.2.4 A Hybrid Non-dominated Sorting Differential Evolutionary Algorithm

Zhong et al. [157] hybridized the NSS algorithm with *Differential Evolution (DE)* [129]. The proposal of Zhong et al. adopts NSS as its local search engine in order to obtain nondominated solutions during the evolutionary process according to the Pareto dominance relation. The sorting strategy adopted in this approach, involves the evaluation of the fitness function of each solution, and the dominance relation among the individuals in the population is defined according to their fitness cost. Thought the search, the nondominated solutions are stored in a separate set A which, at the end of the search, will constitute an approximation of the $\mathcal{PS}$.

At each iteration t, DE generates an offspring population Q_t by evolving each solution $\mathbf{x}_i$ of the current population P_t. The DE/best/2 strategy is employed in order to generate the trial vector $\mathbf{v}_i$:

$$\mathbf{v}_i = \mathbf{x}_i^{best} + F \cdot (\mathbf{x}_{r_0} - \mathbf{x}_{r_1}) + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{4.10}$$

where $\mathbf{x}_{r_0}, \mathbf{x}_{r_1}, \mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$ are different solutions taken of P_t. $\mathbf{x}_i^{best}$ is a solution randomly chosen from the set of nondominated solutions A.

Algorithm 11: The Nonlinear Simplex Search Genetic Algorithm

Input:
t_{max}: the maximum number of generations;
Output:
P: the final approximation to the Pareto front;

begin
  $t = 0$;
  Randomly initialize a population P_t of size N;
  Evaluate the fitness of each individual in P_t;
  while $t < t_{max}$ **do**
    $Q_t = \text{MUTATION}(\text{CROSSOVER}(B))$; `// Apply genetic operators of NSGA-II`
    $R_t = P_t \cup Q_t$;
    Assign to P^* the N better individuals from R_t `// According to crowding comparison operation;`
    if $(t \mod \frac{n}{2} = 0)$ **then**
      Get Λ set by minimizing each function of the MOP and the aggregating function (equation (4.9)) by using the NSS algorithm.
      $R_t^* = P_t^* \cup \Lambda$;
      Assign to P_{t+1} the N better individuals from R_t^* `// According to crowding comparison operation;`
    else
      $P_{t+1} = P^*$;
    end
    $t = t + 1$;
  end
  return P_t
end

The trial vector $\mathbf{v}_i$ is then used for generating the new solution $\mathbf{x}_i'$ by using the binary crossover:

$$\mathbf{x}_i'(j) = \begin{cases} \mathbf{v}_i(j) & \text{if } r < CR \\ \mathbf{x}_i(j) & \text{otherwise} \end{cases} \tag{4.11}$$

where r is a random number having uniform distribution, $j = 1, \ldots, n$ is the j^{th} parameter of each vector and CR represents the crossover ratio.

After the whole offspring population Q_t is generated, the nondominated sorting of $P_t \cup Q_t$ is used for obtaining the set of N solutions (N is the number of solutions in P_t) for the next population P_{t+1}.

In the local search procedure, the simplex S is built by selecting randomly a nondominated solution from A, the other n vertices of the simplex (where n is the number of decision variables) are randomly chosen from the current population P_t. If the population P_t cannot provide enough points to compose the simplex, other points are selected from A. After the simplex is built, the vertices of the simplex are stored by using nondominated sorting, in analogous way as in the *Non-dominated Sorting Differential Evolution (NSDE)* algorithm [2]. The movements in the simplex are performed according to the NSS algorithm. However, for the comparison among the solutions, the dominance relation is used instead of a function cost. The shrinkage step is performed if either inside or outside contractions fail; in this case, all the vertices into the transformed simplex S are sorted to obtain the solutions which are nondominated. Considering m as the number of the nondominated solutions in the simplex. The shrinkage step is performed according to next the description.

If $m > 1$, there exist different converging directions, which could help to maintain the diversity of the solutions. Then, new simplexes $S_1, S_2, \ldots, S_m$, which adopt a nondominated solution each, as respective guiding point, are generated. The new simplexes are stored within a bounded array. If the total number simplexes exceeds the storing space of the array, no more new simplexes are accepted. Then, these simplexes iterate to shrink to the $\mathcal{PF}$. If $m \leqslant 1$ or $S \in S_1, \ldots, S_m$, the nondominated point m must be set correspondingly, in the simplex S_m as the guiding point $\mathbf{x}_1$. The vertices in the simplex are relocated according to:

$$\mathbf{v}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1), i = 2, \ldots, n+1,$$

where σ is the shrinkage coefficient. The new simplex uses $\mathbf{x}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n+1}$ as vertices to form the new simplex.

The Euclidean distance among the centroid and the vertices of the simplex is used for assessing the convergence at each simplex. After the convergence has taken place in all simplexes of the array, the

nondominated solutions found by NSS are introduced into the population of the evolutionary algorithm according to a nondominated sorting strategy, in an analogous way as in NSDE.

The authors suggested a population size of $N = 20 \times k \times n$ where n and k represent the number of decision variables and the number of objective functions of the MOP, respectively. The DE/best/2/bin strategy was used with a crossover ratio $CR = 0.8$ and a weighting factor $F = 0.5$. The NSS algorithm was performed using $\rho = 1, \chi = 2$, $\gamma = 0.5$ and $\sigma = 0.5$ for the reflection, expansion, contraction and shrinkage movements, respectively. The Euclidean distance criterion to assess the convergence was set as 1×10^{-12}. For a more detailed description of this MOMA, the interested reader is referred to [157].

4.2.5 A Hybrid Multi-objective Evolutionary Algorithm based on the S Metric

Koch et al. [77] introduced a hybrid algorithm which combines the exploratory properties of the *S-Metric Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA)* [6] with the exploitative power of the Hooke and Jeeves algorithm [60] which is used as a local search engine. SMS-EMOA optimizes a set of solutions according to the S-metric or Hypervolume indicator [160], which measures the size of the space dominated by the population. This performance measure is integrated into the selection operator of SMS-EMOA which aims for maximization of the S-metric and thereby guides the population to the $\mathcal{PF}$. A $(\mu + 1)$ (or steady-state) selection scheme is applied. At each generation, SMS-EMOA discards the individual that contributes least to the S-metric value of the population. The invoked variation operators are not specific for the SMS-EMOA but taken from the literature, namely PBM and SBX with the same parametrization as NSGA-II [27]. At each iteration the Hooke and Jeeves algorithm performs an exploratory move along the coordinate axes. Afterwards, the vectors of the last exploratory moves are combined to a projected direction that can accelerate the descent of the search vector. When the exploratory moves lead to no improvement in any coordinate direction, step sizes are reduced by a factor η. The search terminates after a number of predefined function evaluations or, alternatively, when the step size falls below a constant value $\varepsilon > 0$.

The Hooke and Jeeves was conceived for minimizing single-objective optimization problems, therefore, its use to deal with MOPs is not possible without modifications. Koch et al. adopted a scalar function by using the weighting sum approach developed in [26]. Besides, the proposed MOMA introduces a probability function $p_{ls}(t)$ for extending the idea presented by Sindhya et al. [123] who linearly oscillate the probability for starting local search. The probability function adopted in this work is given by:

$$p_{ls}(t) = \frac{p_{max} \cdot \Phi(t \mod (\alpha\mu))}{\Phi(\alpha\mu - 1)} \tag{4.12}$$

where parameter μ is the population size of the MOEA and $\alpha \in (0, 1]$ is a small constant value—in the experiments the authors suggested to use $\alpha = 0.05$. The probability function oscillates with period $\alpha \cdot \mu$ and is linear decreasing in each period. The auxiliary function Φ determines the type of reduction, i.e. linear, quadratic or logarithmic, and has to be defined by the user. Algorithm 12 shows the general framework of the proposed hybrid SMS-EMOA.

Koch et al. hybridized also the SMS-EMOA with other mathematical programming techniques. The multi-objective Newton method [39] and the step descent method [40] are hybridized with the SMS-EMOA. Koch et al. emphasize the importance of the used probability function p_{ls} that controls the frequency of local search during the optimization process. Three different functions using equation (4.12) and a constant probability p_{ls} were adopted. The hybrid approaches use equation (4.12) to obtain a value of $\alpha = 0.5$ as it was proposed by Sindhya et al. [123] and the next functions were used.

1. $p_{ls}(t)$ with $\Phi(x) = x$ (in equation (4.12))
2. $p_{ls}(t)$ with $\Phi(x) = x^2$ (in equation (4.12))
3. $p_{ls}(t)$ with $\Phi(x) = \log(x)$ (in equation (4.12))
4. $p_{ls}(t)$ with $\Phi(x) = 0.01$

Each hybridization with the above probability functions was tested on the *Zitzler-Deb-Thiele (ZDT)* test suite. The hybrid SMS-EMOA is started with a population size of $N = 100$. The SBX recombination and the PBM mutation operator, were employed. The authors reported that the hybrid algorithm using the multi-objective Newton

Algorithm 12: The hybrid SMS-EMOA

Input:
T_{max}: The maximum number of generations;
Output:
A: The final approximation to the *Pareto front* ($\mathcal{PF}$);

```
1  begin
2      t = 0;
3      Generate a population P_t of size N; // using uniform
       distribution
4      Evaluate the population P_t;
5      while t < T_max do
6          Select μ parents of P_t;
7          Create population Q_t with λ offspring;
8          for i=1 to λ do
9              Choose random variable r ∈ [0, 1];
10             if r ⩽ p_ls(t) then
11                 Local search for Q_t[i];
12             end
13         end
14         Evaluate λ offspring;
15         Create population P_{t+1} out of P_t and Q_t;
16         t = t + 1;
17     end
18 end
```

method achieved better results than those obtained by both the hybrid SMS-EMOA using Hooke and Jeeves algorithm, and the one using the step descent method. More details of this hybridization can be found in [77].

5

A Nonlinear Simplex Search for Multi-Objective Optimization

The development of multi-objective mathematical programming techniques has been a very active area of research for many years, giving rise to a wide variety of approaches [35, 96, 99, 138]. Recently, several powerful approaches that rely on gradient information, have been proposed. For example, Fliege et al. [39] proposed an extension of Newton's method [103] for unconstrained multi-objective optimization. Fischer and Shukla [37] introduced an algorithm based on the Levenberg and Marquardt method [85, 91] to solve nonlinear unconstrained *Multi-objective Optimization Problems (MOPs)*. These and other gradient-based methods (see e. g. [7, 89]) have taken advantage of using gradient information of the functions to generate Pareto optimal solutions of a MOP. However, when the gradient of the functions is not available, the use of these methods becomes impractical, and it is necessary to look for alternative approaches.

An alternative is to use methods that do not require the gradient information of the functions—the well-known direct search methods. However, the use of such methods in a multi-objective optimization context has been scarce. Nevertheless, as we have seen in Chapter 4, some researchers have used them as local search operators in *Multi-Objective Evolutionary Algorithms (MOEAs)*. To the authors' best knowledge, no method is currently available to approximate multiple solutions to the *Pareto optimal set (PS)* (maintaining a good distribution of the *Pareto front (PF)*) using direct search methods that are not based on metaheuristics. The main reason for the shortage in such strategies, is that it is not efficient to approximate different solutions the the PS maintaining a good representation of the PF, by using such mathematical programming techniques.

As a preliminary study, in this thesis, we present an extension of a popular direct search method—the well-known Nelder and Mead algorithm [102] (which was originally proposed for single-objective

optimization)—for dealing with MOPs. The proposed approach is based on the decomposition of a MOP into several single-objective scalarization functions, in which each function consists of the aggregation of all the (original) objective functions f_i's. With that, multiple solutions of the $\mathcal{PS}$ are achieved by solving each optimization problem. Each optimization problem is solved by deforming a geometric shape called *simplex* according to the movements described by Nelder and Mead's algorithm.

The main goal of this study is to analyze and exploit the properties of Nelder and Mead's algorithm when it is used to approximate solutions to the $\mathcal{PS}$ while maintaining a reasonably good representation of the $\mathcal{PF}$. In the following section, we describe in detail, the Nelder and Mead method in which, our proposed multi-objective direct search method is based.

5.1 The Nonlinear Simplex Search

Nelder and Mead's method [102] (also known as the *Nonlinear Simplex Search (NSS)*), is an algorithm based on the simplex algorithm of Spendley et al. [126], which was introduced for minimizing nonlinear and multi-dimensional unconstrained functions. While Spendley et al.'s algorithm uses regular simplexes, Nelder and Mead's method generalizes the procedure to change the shape and size of the simplex. Therefore, the convergence towards a minimum value at each iteration of the NSS method is conducted by four main movements in a geometric shape called *simplex*. The following definitions are of relevance to the remainder of this algorithm.

Definition 5.1 (n-simplex)
A *simplex* or n-*simplex* is a convex hull of a set of $n+1$ affinely independent points Δ_i $(i = 1, \ldots, n+1)$, in some Euclidean space of dimension n.

If the vertices of the simplex are all mutually equidistant, then the simplex is said to be regular. Thus, in two dimensions, a regular simplex is an equilateral triangle, while in three dimensions a regular simplex is a regular tetrahedron.

Definition 5.2 (Degenerated simplex)
A simplex is called *nondegenerated*, if and only if, the vectors in the simplex denote a linearly independent set. Otherwise, the simplex is called *degenerated*, and then, the simplex will be defined in a lower dimension than n.

The NSS expands or focuses the search adaptively on the basis of the topography of the fitness landscape. The full algorithm is defined stating four scalar parameters to control the movements performed in the simplex: **reflection** (ρ), **expansion** (χ), **contraction** (γ) and **shrinkage** (σ). According to Nelder and Mead, these parameters should satisfy:

$$\rho > 0, \qquad \chi > 1, \qquad \chi > \rho, \qquad 0 < \gamma < 1 \quad \text{and} \quad 0 < \sigma < 1$$

Actually, there is no method that can be used to establish this set of parameters. However, the nearly universal choices used in Nelder and Mead's method are [102]:

$$\rho = 1, \qquad \chi = 2, \qquad \gamma = \frac{1}{2}, \qquad \text{and} \qquad \sigma = \frac{1}{2}$$

At each iteration of the NSS algorithm, the $n+1$ vertices Δ_i's of the simplex represent solutions which are evaluated and sorted according to: $f(\Delta_1) \leqslant f(\Delta_2) \leqslant \ldots \leqslant f(\Delta_{n+1})$. Let's consider $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_{n+1}\}$ as the simplex with the vertices sorted according to the function value. Then, the transformations performed by the NSS into the simplex are defined as:

1. Reflection: $\mathbf{x}_r = (1+\rho)\mathbf{x}_c - \rho\Delta_{n+1}$ (see Figure 5.2).
2. Expansion: $\mathbf{x}_e = (1+\rho\gamma)\mathbf{x}_c - \rho\chi\Delta_{n+1}$ (see Figure 5.3).
3. Contraction:
   a) *Outside*: $\mathbf{x}_{oc} = (1+\rho\gamma)\mathbf{x}_c - \rho\gamma\Delta_{n+1}$.
   b) *Inside*: $\mathbf{x}_{ic} = (1-\gamma)\mathbf{x}_c + \gamma\Delta_{n+1}$ (see Figure 5.4).
4. Shrinkage: Each vertex of the simplex is transformed by the geometric shrinkage defined by: $\Delta_i = \Delta_1 + \sigma(\Delta_i - \Delta_1)$, $i = 2, \ldots, n+1$, and the new vertices are evaluated (see Figure 5.5).

where $\mathbf{x}_c = \frac{1}{n}\sum_{i=1}^{n}\Delta_i$ is the centroid of the n best points (all vertices except for Δ_{n+1}), Δ_1 and Δ_{n+1} are the best and the worst solutions identified within the simplex, respectively.

At each iteration, the simplex is modified by one of the above movements, according to the following rules:

1. If $f(\Delta_1) \leqslant f(\mathbf{x}_r) \leqslant f(\Delta_n)$, then $\Delta_{n+1} = \mathbf{x}_r$.
2. If $f(\mathbf{x}_e) < f(\mathbf{x}_r) < f(\Delta_1)$, then $\Delta_{n+1} = \mathbf{x}_e$, otherwise $\Delta_{n+1} = \mathbf{x}_r$.
3. If $f(\Delta_n) \leqslant f(\mathbf{x}_r) < f(\Delta_{n+1})$ and $f(\mathbf{x}_{oc}) \leqslant f(\mathbf{x}_r)$, then $\Delta_{n+1} = \mathbf{x}_{oc}$, otherwise perform a shrinkage.
4. If $f(\mathbf{x}_r) \geqslant f(\Delta_{n+1})$ and $f(\mathbf{x}_{ic}) < f(\Delta_{n+1})$, then $\Delta_{n+1} = \mathbf{x}_{ic}$, otherwise perform a shrinkage.

Until now, we have presented in detail the description of Nelder and Mead's algorithm. The next section is dedicated to explain in detail, the proposed *Nonlinear Simplex Search* (NSS) for multi-objective optimization.

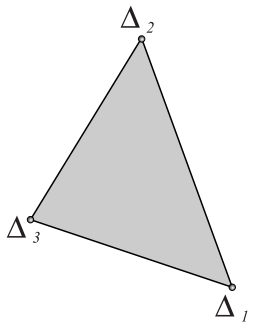


Figure 5.1.: A 2-simplex

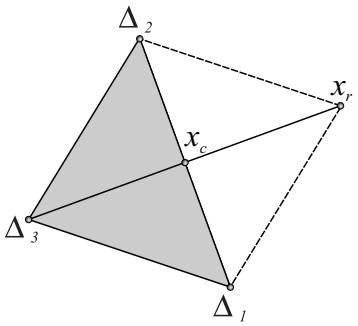


Figure 5.2.: Reflection

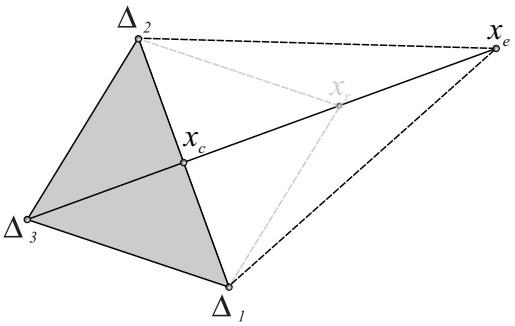


Figure 5.3.: Expansion

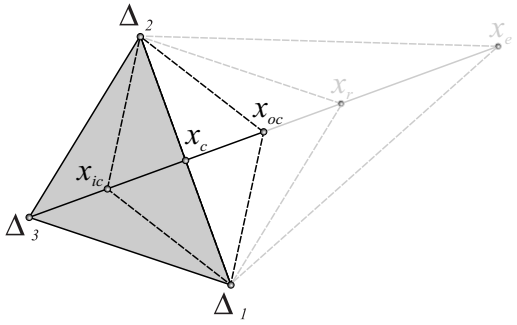


Figure 5.4.: Inside and outside contraction

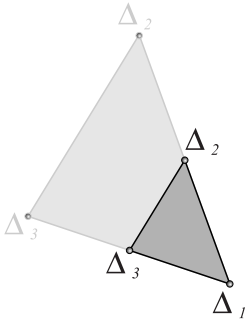


Figure 5.5.: Shrinkage

5.2 The Nonlinear Simplex Search for Multi-Objective Optimization

5.2.1 Decomposing MOPs

There are several approaches for transforming a MOP into a single-objective optimization problem. Most of these approaches use a weight vector for defining their search directions. In this way, and under certain assumptions (e.g., the minimum is unique, the weighting coefficients are positive, etc.), a Pareto optimal solution is achieved by solving such optimization problem. In Chapter 3 (Section 3.2.2), we presented some methods of the wide variety of approaches that transform a MOP in a single-objective optimization problem (an extensive review of such methods can be found in [35, 96, 99, 138]). Among these methods, probably the two most widely used are the *Tchebycheff* and the *Weighted Sum* approaches. However, as previously discussed in [19, 155], the approaches based on boundary intersection possess certain advantages over those based on either Tchebycheff or the Weighted Sum. In this thesis, we shall adopt the *Penalty Boundary Intersection (PBI)* approach as our method to transform a MOP into a single-objective optimization problem. The description of the PBI approach was presented in Chapter 3 (Section 3.2.2 in page 28). However, in order to remember the optimization problem to which the PBI approach refers, we present the mathematical formulation which is stated as:

PBI method belongs to the approaches based on boundary intersection.

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w},\mathbf{z}^\star) = d_1 + \theta d_2 \tag{5.1}$$

such that:

$$d_1 = \frac{\|(F(\mathbf{x}) - \mathbf{z}^\star)^T \mathbf{w}\|}{\|\mathbf{w}\|}$$
$$\text{and} \quad d_2 = \left\| (F(\mathbf{x}) - \mathbf{z}^\star) - d_1 \frac{\mathbf{w}}{\|\mathbf{w}\|} \right\|$$

where $\mathbf{x} \in \mathbb{R}^n$, θ is the penalty value, $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ is the utopian vector, i.e., $z_i^\star = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}, \forall i = 1, \ldots, k$, and $\mathbf{w}$ is a weight vector, such that $w_i \geqslant 0$ for all $i = 1, \ldots, k$ and $\sum_{i=1}^{k} w_i = 1$.

Therefore, if the weight vectors are well distributed, an appropriate representation of the $\mathcal{PF}$ could be reached by solving the different

scalarization problems [155]. Such weight vectors define then, the search direction in the optimization process. This strategy of decomposition is employed by the multi-objective direct search algorithm presented in this Chapter.

5.2.2 About the Nonlinear Simplex Search and MOPs

Mathematical programming techniques are known to have several limitations compared to *Evolutionary Algorithms (EAs)*. As mentioned before, many mathematical programming methods were designed to deal with convex functions and a number of them require gradient information. Being a direct search method, Nelder and Mead's method has the advantage of not requiring gradient information. Instead, the NSS algorithm aims at obtaining a better solution by deforming a simplex shape along the search process. Nonetheless, Nelder and Mead's method has an important disadvantage: convergence towards an optimal value can fail when the simplexes elongate indefinitely and their shape goes to infinity in the space of simplex shapes (as, e.g, in McKinnon's functions [93]). For this family of functions and others having similar features, a more appropriate strategy needs to be adopted (e.g., adjusting the control parameters, constructing the simplex in a different way, modifying the movements into the simplex, etc.). In recent years, several attempts to improve the NSS method have been reported in the specialized literature, see for example [4, 8, 110, 133]. However, due to its inherent nature (based on heuristic movements), several of these variants of the NSS algorithm normally produce additional problems, and in some cases, they fail in more cases than the original algorithm.

In addition to any changes to the NSS algorithm itself, it is also possible to propose different strategies for constructing the simplex, and several researchers have reported work in that direction, see e. g. [12, 146]. The construction of the simplex plays an important role in the performance of the NSS algorithm. For example, to use a degenerated simplex (i. e., a simplex defined in lower dimensionality than the number of decision variables) in the minimization process, is inappropriate. The reason is that in such case, the search is restricted to find an optimal solution in lower dimensionality, which avoids achieving this optimal solution if it is not allocated in the same

dimensionality of the simplex [82]. However, the use of a degenerated simplex could, at least, obtain local minima, in the dimensionality defined by the simplex.

In most real-world MOPs, the features of the $\mathcal{PS}$ are unknown. When a Pareto optimal solution is found, the property that exists when using a degenerated simplex in the search could be exploited. With that, multiple efficient solutions will be found in the same dimension if they exist. Since in our case the search is directed by a well-distributed set of weight vectors, each of which defines a scalarization problem, and we assume that each subproblem is solved throughout the search, then, the simplex could be constructed using such solutions. In this way, multiple trade-off solutions are achieved while the search eventually converges to the region in which the $\mathcal{PS}$ is contained.

The convergence towards a better point given in the Nelder and Mead algorithm is achieved at most in $n + 1$ iterations (at least in convex problems with low dimensionality) [82]. Thus, for solving each subproblem (of the decomposition) a considerable number of function evaluations could be required. On the other hand, the execution of the shrinkage step in the NSS algorithm could become inefficient. This can be caused by two main facts:

1. Once the simplex is transformed by the shrinkage step, the new vertices need to be evaluated. Thus, when the dimensionality of the MOP is high, the number of the objective function evaluations could significantly increase,

2. Since the shrinkage step reduces the simplex volume, the search is then restricted to a small portion of the search space. Therefore, the risk to collapse the search in a specific region of the search space is increased. Whereupon, the diversity of solutions along the $\mathcal{PF}$ could be reduced, which is a disadvantage for the *Decision Maker (DM)* in a multi-objective optimization context.

Therefore, an appropriate strategy for approximating the $\mathcal{PS}$ and maintaining a good representation of the $\mathcal{PF}$ needs to be adopted.

We have taken into account the above observations to design an effective NSS approach for solving unconstrained MOPs. The proposed direct search multi-objective optimization method is described in the following section.

5.2.3 The Multi-Objective Nonlinear Simplex Search

The *Multi-objective Nonlinear Simplex Search (MONSS)* algorithm proposed here, decomposes a MOP into several single-objective scalarization subproblems by using the PBI approach. Therefore, in order to obtain a good representation of the $\mathcal{PF}$, a well-distributed set of weight vectors $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ needs to be previously defined. In this thesis, the method used by Zhang and Li [155] is adopted for that sake. However, other methods can be also used—see e. g. [18, 141].

At the beginning, a set of N vectors $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, having a uniform distribution, is randomly initialized. Each vector $\mathbf{x}_i \in \mathcal{S}$ represents a solution for the i^{th} subproblem defined by the i^{th} weight vector $\mathbf{w}_i \in W$. In this way, different subproblems are simultaneously solved by the MONSS algorithm and the set of solutions $\mathcal{S}$ will constitute an approximation of the $\mathcal{PS}$ lengthwise of the search process. In order to find different solutions along the $\mathcal{PF}$, the search is directed towards different non-overlapping regions (or partitions) C_i's from the set of weight vectors W, such that, each C_i defines a neighborhood. That is, let $C = \{C_1, \ldots, C_m\}$ be a set of partitions from W, then, the claim is the following:

$$\bigcap_{i=1}^{m} C_i = \emptyset \text{ and } \bigcup_{i=1}^{m} C_i = W \tag{5.2}$$

and all the weight vectors $\mathbf{w}_c \in C_i$ are contiguous among themselves.

The NSS algorithm is focused on minimizing a subproblem defined by a weight vector $\mathbf{w}_s$ which is randomly chosen from C_i. In order to save the objective function evaluations, the shrinkage step is omitted in the proposed approach. The n-simplex (Δ) used by the NSS algorithm, is defined as:

$$\Delta = \{\mathbf{x}_s, \mathbf{x}_1, \ldots, \mathbf{x}_n\} \tag{5.3}$$

such that: $\mathbf{x}_s \in \mathcal{S}$ is a minimum of $g(\mathbf{x}_s|\mathbf{w}_s, \mathbf{z}^\star)$ for any $\mathbf{w}_s \in W$. $\mathbf{x}_j \in \mathcal{S}$ represents the j^{th} solution that minimizes the subproblems defined by the n nearest weight vectors of $\mathbf{w}_s$, where $j = 1, \ldots, n$ and n represents the number of decision variables of the MOP.

After any movement made by the NSS algorithm, it is common that the new obtained solution ($\mathbf{x}_n$), leaves the search space. In order

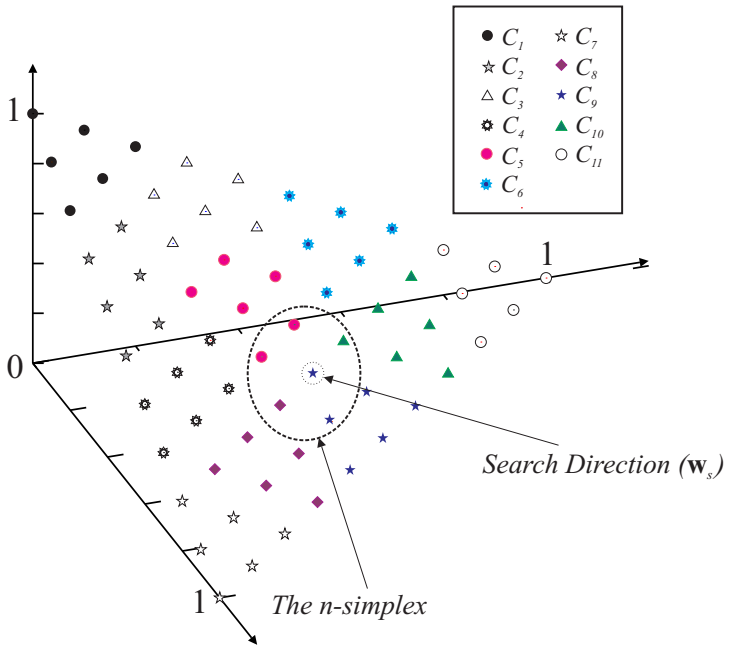


Figure 5.6.: Illustration of a well-distributed set of weight vectors for a MOP with three objectives, five decision variables and 66 weight vectors, i.e. $m = \left\lfloor \frac{|W|}{n+1} \right\rfloor = 11$ partitions. The n-simplex is constructed by six solutions that minimize different problems defined by different weight vectors contained in four partitions (C_5, C_8, C_9 and C_{10}). The search is focused on the direction defined by the weight vector $\mathbf{w}_s$.

to deal with this problem, as in [146] we bias the boundaries in a deterministic way. Therefore, the i^{th} bound of the new solution $\mathbf{x}_n$ is re-established as follows:

$$x_n^i = \begin{cases} x_{lb}^i, & \text{if } x_n^i < x_{lb}^i \\ x_{ub}^i, & \text{if } x_n^i > x_{ub}^i \end{cases} \tag{5.4}$$

where x_{lb}^i and x_{ub}^i are, respectively, the lower and upper bounds in the i^{th} component of the search space.

To speed up convergence to the $\mathcal{PS}$, the search is relaxed at each iteration by changing the direction vector for any other direction $\hat{\mathbf{w}}_s \in C_i$. In this way, an agile search into the partition C_i is performed avoiding a collapse of the search in the same direction $\mathbf{w}_s$. Here, we define $m = \left\lfloor \frac{|W|}{n+1} \right\rfloor$ partitions of the set W, guaranteeing at least

Algorithm 13: $\mathtt{update}(W, \mathcal{S}, \mathcal{I})$

Input:
W: a well-distributed set of weight vectors;
$\mathcal{I}$: the intensification set;
$\mathcal{S}$: the current approximation to the $\mathcal{PF}$;
Output:
$\mathcal{R}$: a new approximation to the $\mathcal{PF}$;

```
1 begin
2     T = S ∪ I;
3     R = ∅;
4     forall the w_i ∈ W do
5         R = R ∪ {x*| min_{x*∈T} g(x*|w_i, z*)};
6         T = T \ {x*};
7     end
8     return R;
9 end
```

$n + 1$ iterations of the NSS algorithm for each partition, which can be constructed using a naive modification of the well-known k-means algorithm [90].

One iteration of the MONSS algorithm is carried out when the NSS iterates $n + 1$ times in each defined partition C_i. Therefore, at each iteration, the proposed algorithm performs $|W|$ function evaluations. All of the new solutions found in the search process are stored in a pool called *intensification set* denoted as $\mathcal{I}$. At the end of each iteration, the set of solutions $\mathcal{S}$ is updated using both the intensification set $\mathcal{I}$ and the weight set W, as shown in Algorithm 13.

In this way, the NSS minimizes each subproblem, generating new search trajectories among the solutions of the simplex, while the updating mechanism replaces the misguided paths by selecting the best solutions according to the PBI approach, simulating the Path Relinking method [48]. In Figure 5.6, we show a possible partition of the weight set W for a MOP with three objective functions and five decision variables, i.e., defining an n-simplex with six vertices. For an easy interpretation of the proposed MONSS, Algorithm 14 describes the complete methodology to deal with MOPs using the Nelder and Mead algorithm.

Algorithm 14: The *Multi-objective Nonlinear Simplex Search (MONSS)* algorithm

Input:
W: a well-distributed set of weight vectors;
max_{it}: a maximum number of iterations;
Output:
$\mathcal{S}$: an approximation to the $\mathcal{PF}$;

begin
  $t = 0$;
  Generate initial solutions: Generate a set $\mathcal{S}^t = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N random solutions;
  Generate partitions: Generate $m = \frac{|W|}{n+1}$ partitions $C = \{C_1, \ldots, C_m\}$ from W (where n is the number of decision variables), such that: Equation (5.2) is satisfied;
  while $t < max_{it}$ **do**
    for $i = 0$ **to** m **do**
      Randomly choose $\mathbf{w}_s \in C_i$;
      Apply Nonlinear Simplex Search algorithm:
        a) **Build the n-simplex:** Construct the n-simplex from $\mathcal{S}^t$, such that: Equation (5.3) is satisfied.
        b) **Apply the NSS method:** Execute the NSS algorithm during $n+1$ iterations. At each iteration:
          * Repair the bounds according to Equation (5.4).
          * Relax the search changing the search direction $\mathbf{w}_s$ for any other $\hat{\mathbf{w}}_s \in C_i$.
          * Each new solution generated by any movements of the NSS algorithm is stored in the intensification set $\mathcal{I}$.
    end
    Update the leading set: Update the set $\mathcal{S}$ using Algorithm 13. That is: $\mathcal{S}^{t+1} = update(W, \mathcal{S}^t, \mathcal{I})$;
    $t = t + 1$;
  end
  return $\mathcal{S}^t$;
end

5.3 Experimental Study

5.3.1 Test Problems

In order to assess the performance of the proposed MONSS algorithm, we compare its results with respect to those obtained by a

state-of-the-art MOEA, the well-known *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*, which has shown a good performance compared to other MOEAs, see [155]. Similar to MONSS, MOEA/D decomposes a MOP into several scalarization problems. However, instead of using mathematical programming techniques, MOEA/D uses genetic operators to approximate the $\mathcal{PF}$.

In our experiments, we adopted ten MOPs with two and three objectives, whose $\mathcal{PF}$s have different characteristics including convexity, concavity and disconnections. The adopted test problems correspond to the nine MOPs defined as classic problems in Appendix A.1 and the DTLZ5 test problem taken from the *Deb-Thiele-Laumanns-Zitzler (DTLZ)* test suite (see Appendix A.3).

5.3.2 Performance Assessment

In order to assess the performance of our proposed MONSS algorithm, we compared it with respect to MOEA/D using the *Hypervolume* (I_H) and the *Two Set Coverage* (I_C) performance measures. The characteristics of such performance measures were presented in Chapter 3, and we refer to section 3.4 for a more detailed description of these performance indicators.

5.3.3 Parameters Settings

As indicated before, we compared the results obtained by our proposed MONSS with respect to those obtained by MOEA/D [155]. The weight vectors for the algorithms were generated as in [155], i.e., the setting of N and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ is controlled by a parameter H. More precisely, $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}$$

Therefore, the number of such vectors in W is given by:

$$N = C^{k-1}_{H+k-1}.$$

where k is the number of objective functions.

For all the MOPs, MONSS and MOEA/D were tested with $H = 99$ for MOP with two objectives, i.e. 100 weight vectors, and $H = 23$ for MOPs with three objectives, i.e. 300 weight vectors. For a fair comparison, both approaches used the same scalarization function in the decomposition approach, in this case the PBI method.

For each MOP, 30 independent runs were performed with each algorithm. The parameters for both algorithms are summarized in Table 7, where N_{sol} represents the number of initial solutions (100 for bi-objective problems and 300 for three-objective problems). N_{it} represents the maximum number of iterations, which was set to 40 for all test problems. Therefore, both algorithms performed 4,000 (for the bi-objective problems) and 12,000 (for the three-objective problems) function evaluations for each problem. For the proposed MONSS, ρ, χ and γ represent the control parameters for the reflection, expansion and contraction movements of the NSS algorithm, respectively. For MOEA/D, the parameters T_n, η_c, η_m, P_c and P_m represent the neighborhood size, crossover index, mutation index, crossover rate and mutation rate, respectively. Finally, the parameter θ, represents the penalty value used in the PBI approach for both MONSS and MOEA/D.

Parameter	MONSS	MOEA/D
N_{sol}	100/300	100/300
N_{it}	40	40
T_n	–	30
P_c	–	1
P_m	–	$1/n$
ρ	1	–
χ	2	–
γ	1/2	–
θ	5	5

Table 1.: Parameters for MONSS and MOEA/D

For each MOP, the algorithms were evaluated using the two performance measures previously defined: (I_H and I_C). The results obtained are summarized in Tables 2 and 3. Each table displays both the *average* and the *standard deviation* (σ) of each performance measure for each MOP. The reference vectors used for computing the I_H performance measure are shown in Table 2. These vectors are established near to the individual minima for each MOP, i.e., close to the extremes of the $\mathcal{PF}$. With that, a good measure of approximation and distribution is reported when the algorithms converge along the $\mathcal{PF}$. In the

case of the statistics for the I_C comparing pairs of algorithms (i.e., $I_C(A, B)$), they were obtained as average values of the comparison of all the independent runs from the first algorithm with respect to all the independent runs from the second algorithm. For an easier interpretation, the best results are presented in **boldface** for each performance measure and test problem adopted.

5.4 Numerical Results

Tables 2 and 3 show the results obtained for the *Hypervolume* (I_H) and the *Two Set Coverage* (I_C) performance measures, respectively. From this table, it can be seen that the results obtained by our proposed MONSS outperform those obtained by MOEA/D in most of the test problems adopted. This means that the proposed approach achieved a better convergence and spread of solutions along the $\mathcal{PF}$. The exception was VNT2, where MOEA/D obtained a better value in the I_H indicator. However, given the small difference obtained in this performance measure, we consider that MONSS was not significantly outperformed by MOEA/D in this case.

Regarding the I_C performance measure, our proposed MONSS obtained better results when compared against those produced by MOEA/D in most of the test problems adopted. This means that the solutions obtained by MONSS dominated a higher ratio of solutions produced by MOEA/D. However, MOEA/D was better for DTLZ5 and REN1, although the ratio of solutions dominated by MOEA/D was not significantly high in these cases. Although the I_C performance measure is better for MOEA/D in these two problems, it is worth noting that our proposed approach reached better results in the I_H performance measure. The reason for that is that I_H also measures the spread of solutions along the $\mathcal{PF}$, and our approach was better in that regard for DTLZ5 and REN1.

Figures 5.7 and 5.8 show the hypervolume values at each iteration of the two algorithms compared. From these plots, it is possible to see that the performance of both algorithms (MONSS and MOEA/D) was similar in most cases. However, there were also some cases in which our proposed MONSS reached the $\mathcal{PF}$ faster than MOEA/D. This illustrates the effectiveness of our proposed approach for solving

MOP	MONSS	MOEA/D	Reference vector ($\mathbf{r}$)
	average (σ)	average (σ)	
DEB2	**0.981552** (0.004504)	0.969845 (0.049164)	$(1.1, 1.1)^T$
DTLZ5	**0.429676** (0.000917)	0.426429 (0.001175)	$(1.1, 1.1, 1.1)^T$
FON2	**0.542006** (0.001476)	0.539159 (0.001406)	$(1.1, 1.1)^T$
LAU	**13.934542** (0.008218)	13.868946 (0.029341)	$(4.1, 4.1)^T$
LIS	**0.309713** (0.007686)	0.259479 (0.009430)	$(1, 1)^T$
MUR	**3.141629** (0.003791)	3.140806 (0.001290)	$(4.1, 4.1)^T$
REN1	**3.612650** (0.000958)	3.596241 (0.019682)	$(37.1, 1.1)^T$
REN2	**18.925039** (0.016614)	18.918943 (0.023277)	$(-1.9, 2.1)^T$
VNT2	2.11357 (0.003068)	**2.114601** (0.002688)	$(4.5, -16.0, -11.5)^T$
VNT3	**11.685911** (0.013195)	11.599974 (0.018481)	$(8.5, 17.5, 0.5)^T$

Table 2.: Results of I_H performance measure for MONSS and MOEA/D

unconstrained nonlinear MOPs with low and moderate dimensionality.

5.5 Remarks

We have proposed a new method based on the use of mathematical programming techniques for approximating solutions along the $\mathcal{PF}$ of a MOP. The proposed approach was, in principle, designed for dealing with unconstrained, and unimodal multi-objective optimization problems having low and moderate dimensionality (2, 3 and 12 decision variables).

Our experimental study indicates that our proposed MONSS outperforms a powerful state-of-the-art multi-objective evolutionary algorithm (MOEA/D) regarding convergence in most of the test problems adopted. The number of objective function evaluations in these test problems was restricted to 4,000 for the bi-objective problems and to 12,000 for the three-objective problems. The good results obtained by our proposed approach with this relatively low number of objective function evaluations suggest that it can be a good choice for dealing

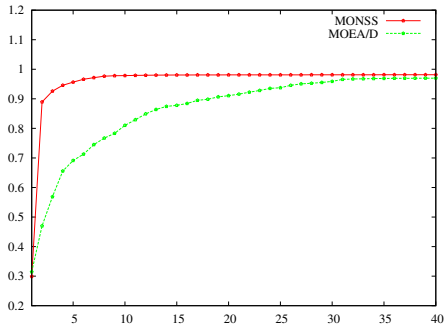
MOP	I_C(MONSS, MOEA/D)	I_C(MOEA/D, MONSS)
	average (σ)	average (σ)
DEB2	**0.190446** (0.053016)	0.146296 (0.035893)
DTLZ5	0.21025 (0.019020)	**0.311705** (0.051739)
FON2	**0.354962** (0.090241)	0.116333 (0.030275)
LAU	**0.072572** (0.060321)	0.056333 (0.028459)
LIS	**0.340798** (0.124927)	0.097992 (0.045691)
MUR	**0.147827** (0.058459)	0.092632 (0.011971)
REN1	0.105443 (0.053141)	**0.146599** (0.042929)
REN2	**0.026274** (0.022809)	0.013468 (0.006563)
VNT2	**0.080900** (0.014464)	0.057426 (0.011687)
VNT3	**0.029109** (0.012831)	0.000501 (0.001278)

Table 3.: Results of I_C performance measure for MONSS and MOEA/D

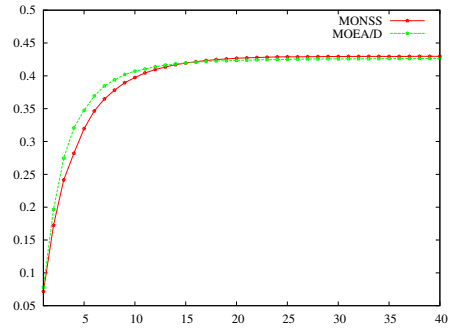
with expensive MOPs having similar characteristics of the problems adopted here.

The main motivation for the algorithm presented in this chapter, has been to show that it is possible to design a competitive multi-objective optimization algorithm using only direct search methods, and without relying on metaheuristic search mechanisms. It is, however, also clear to us that our proposed approach has some disadvantages with respect to multi-objective metaheuristics. The main ones have to do with the difficulties of the NSS method for moving in highly accidented search spaces. It is possible, however, to improve the performance of our proposed approach in such cases by varying the step sizes (i.e., the control parameters ρ, χ and γ) until finding a suitable region of the search space in which the NSS movements can be properly conducted. This is, however, an issue that deserves further research.

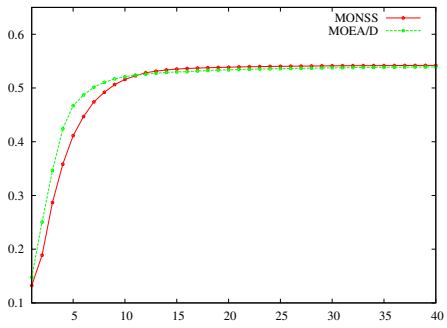
Motivated by the limitations of the proposed approach presented here, in the next chapter, we attempt to hybridize it with a MOEA. Our main motivation for such hybridization is that it can be applied to MOPs of higher dimensionality and having highly accidented search



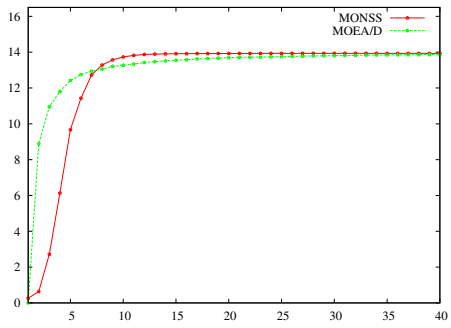
A) Hypervolume convergence for DEB2 problem



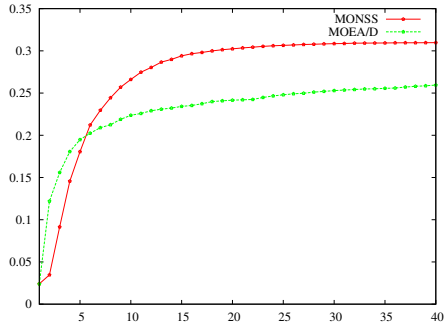
B) Hypervolume convergence for DTLZ5 problem



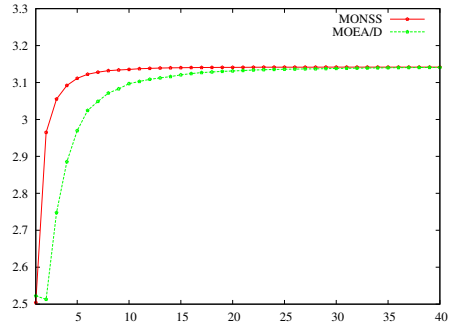
C) Hypervolume convergence for FON2 problem



D) Hypervolume convergence for LAU problem



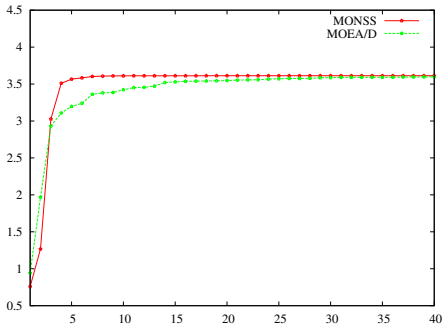
E) Hypervolume convergence for LIS problem



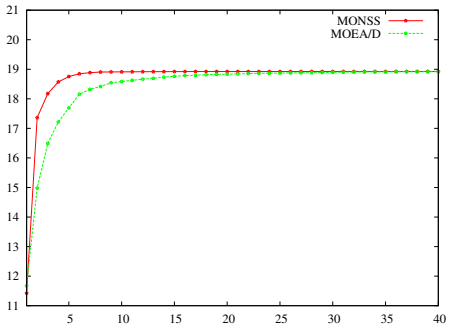
F) Hypervolume convergence for MUR problem

Figure 5.7.: Convergence plot for MONSS and MOEA/D in the test problems DEB2, DTLZ5, FON2, LAU, LIS and MUR.

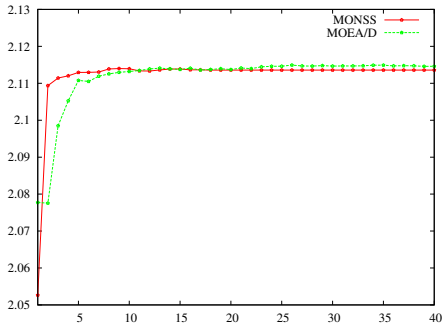
spaces. The idea of such hybridization is to use a MOEA to locate the promising regions of the search space and then adopt our MONSS algorithm to exploit such regions in an efficient manner. We hypothesized that this sort of *Multi-Objective Memetic Algorithm (MOMA)*



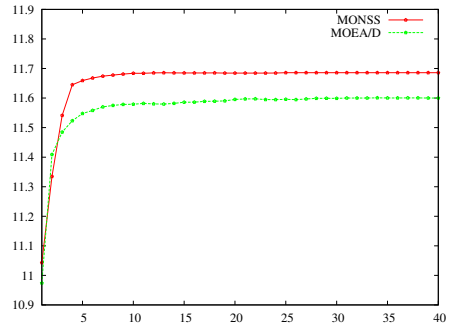
A) Hypervolume convergence for REN1 problem



B) Hypervolume convergence for REN2 problem



C) Hypervolume convergence for VNT2 problem



D) Hypervolume convergence for VNT3 problem

Figure 5.8.: Convergence plot for MONSS and MOEA/D in the test problems REN1, REN2, VNT2 and VNT3

could be a powerful tool for solving complex and computationally expensive MOPs in an efficient and effective manner.

6

A Multi-objective Memetic Algorithm Based on Decomposition

In the previous chapter, we presented a preliminary study about the capabilities of the *Nonlinear Simplex Search (NSS)* to solve *Multi-objective Optimization Problems (MOPs)*. The search strategy employed by the proposed *Multi-objective Nonlinear Simplex Search (MONSS)* proved to be effective on the test functions adopted. The good performance of MONSS was shown not only on benchmark functions, but also on expensive optimization problems, see [145]. Since MONSS is based on the NSS, it inherits its search properties.Therefore, the search performed by MONSS could become inefficient and, in some cases, impractical, when dealing with more complex MOPs, for example those having high dimensionality, high multi-modality or disconnected *Pareto fronts ($\mathcal{PF}$s)*. This has naturally motivated the idea to hybridize the proposed MONSS with a *Multi-Objective Evolutionary Algorithm (MOEA)*.

In this Chapter, we precisely focus on the design of a *Multi-Objective Memetic Algorithm (MOMA)* that combines the search properties of MONSS with the exploratory power of a MOEA. In the proposed approach presented here, the multi-objective direct search method acts as a local search procedure, whose goal is to improve the search performed by the MOEA. Because of its nature, the proposed local search mechanism can be easily coupled to any other decomposition-based MOEA, for example those presented in [98, 107, 149]. In the following sections we present in detail the proposed MOMA.

6.1 The Multi-Objective Memetic Algorithm

6.1.1 General Framework

The proposed *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS)* adopts the well-known *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)* [155] as its baseline algorithm. The local search engine is based on the MONSS framework. However, in order to couple it to MOEA/D and to improve the search, some modifications have been introduced. In this way, the MOEA/D+LS, explores the global search space using MOEA/D, while the local search mechanism exploits promising regions given by the same MOEA/D. Both MOEA/D and MONSS are algorithms that decompose a MOP into several single-objective scalarization problems. Thus, the proposed MOEA/D+LS also decomposes a MOP into several single-objective optimization problems. Such optimization problems are defined by a set of weight vectors. If the weight vectors are evenly distributed, a good representation of the $\mathcal{PF}$ could be reached. Therefore, before starting the search, a well-distributed set of weight vectors needs to be generated.

As indicated before, in this thesis, we employ the *Penalty Boundary Intersection (PBI)* approach to transform a MOP into a single-objective optimization problem, which consists in minimizing:

The full description of the PBI approach was presented in Chapter 3 (Section 3.2.2)

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^\star) = d_1 + \theta d_2 \tag{6.1}$$

such that:

$$d_1 = \frac{||(F(\mathbf{x}) - \mathbf{z}^\star)^T \mathbf{w}||}{||\mathbf{w}||}$$

$$\text{and} \quad d_2 = \left|\left|(F(\mathbf{x}) - \mathbf{z}^\star) - d_1 \frac{\mathbf{w}}{||\mathbf{w}||}\right|\right|$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^n$, θ is the penalty value and $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ is the utopian vector, i.e., $z_i^\star = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}, \forall i = 1, \ldots, k$.

At each iteration, MOEA/D+LS performs one iteration of MOEA/D (see Algorithm 8). After that, the offspring population produced by MOEA/D is improved by using the local search procedure. In Algorithm 15, we present the general framework of MOEA/D+LS while the local search mechanism adopted for MOEA/D+LS is described in detail in the next section.

Algorithm 15: The *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search* (MOEA/D+LS)

Input:
a stopping criterion;
N: the number of the subproblems considered in MOEA/D+LS;
W: a well-distributed set of weighting vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$;
T: the neighborhood size of each weight vector;
R_{ls}: the maximum number of solutions to be replaced by the local search;
A_r: the action range for the local search.

Output:
P: the final population found by MOEA/D+LS.

begin
  Step 1. INITIALIZATION:
  Generate an initial population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ randomly;
  $FV^i = \mathbf{F}(\mathbf{x}_i)$; $B(\mathbf{w}_i) = \{\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}\}$ where $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}$ are the T closest weighting vectors to $\mathbf{w}_i$, for each $i = 1, \ldots, N$;
  $\mathbf{z} = (+\infty, \ldots, +\infty)^T$;
  Step 2. THE MEMETIC ALGORITHM:
  while *stopping criterion is not satisfied* **do**
    Step 2.1) MOEA/D iteration: Perform **Step 2** of the MOEA/D framework for obtaining P (the next population), see Algorithm 8.
    Step 2.2) THE LOCAL SEARCH MECHANISM:
    if *the percentage of nondominated solutions in* P *is less than 50%* **then**
      Step 2.2.1) Selection Mechanism: Select a solution from P as the initial search solution ($\mathbf{p}_{ini}$) according to Section 6.1.2.1;
      Step 2.2.2) Build the Simplex: Build the simplex according to Section 6.1.2.2;
      Step 2.2.3) Search Direction: Select the search direction for the nonlinear simplex search according to Section 6.1.2.3;
      Step 2.2.4) Deform the Simplex: Perform any movement (reflection, contraction or expansion) for obtaining $\mathbf{p}_{new}$ according to Nelder and Mead's algorithm (see Section 5.1);
      Step 2.2.5) Update the population: Update the population P using the new solution $\mathbf{p}_{new}$ according to the rules presented in Section 6.1.2.5;
      Step 2.2.6) Stopping Criterion: If the stopping criterion is satisfied then stop the search. Otherwise go to **Step 2.2.1** or **Step 2.2.3** according to the rules detailed in Section 6.1.2.6;
    end
  end
  return P;
end

6.1.2 Local Search

MOEA/D+LS exploits the promising neighborhoods of the nondominated solutions found by MOEA/D. In the following description, let P be the set of solutions found by MOEA/D in any generation. We assume that if a solution $\mathbf{p} \in P$ is nondominated, there exists another nondominated solution $\mathbf{q} \in \Omega$ such that $\|\mathbf{p} - \mathbf{q}\| < \delta$ for any small $\delta \in \mathbb{R}_+$. In other words, the probability that $\mathbf{q}$ is nondominated with respect to $\mathbf{p}$ in the neighborhood defined by δ is equal to one, which implies that $\mathbf{q}$ is also nondominated.

The local search mechanism presented here takes into account this property to obtain new nondominated solutions departing from nondominated solutions located in the current population P. Let's consider that MOEA/D solves the set of subproblems along the search process. If all solutions in P are nondominated, we assume that the minimum value to each subproblem has been achieved and then, the execution of the local search might no longer be necessary.

The degrees of freedom of the local search depend of the process used for building the simplex, which (as we will see later on) adopts solutions from the current population. Considering that at the end of the evolutionary process the population converges to a particular region of the search space (the place where the nondominated solutions are contained), the performance of the local search engine should be better when the diversity in the population is higher, i. e.., when having a low number of nondominated solutions. Thus, in this algorithm, the local search procedure is applied when the percentage of nondominated solutions in P is less than a certain percentage (we used 50% in this thesis). In the following sections, we will detail the local search steps included in the outlined description of our proposed MOEA/D+LS presented in Algorithm 15.

6.1.2.1 *Selection Mechanism*

Let $P^\star \subseteq P$ be the set of nondominated solutions found by MOEA/D in any generation. Assuming that all the nondominated solutions in $P^\star$ are equally efficient, the solution $\mathbf{p}_{ini}$ which starts the local search is randomly taken from $P^\star$. Solution $\mathbf{p}_{ini}$ represents not only the initial search point, but also the simplex head from which the simplex will be built.

6.1.2.2 *Building the Simplex*

Let $\mathbf{w}_{ini}$ be the weight vector that defines the subproblem for which the initial search solution $\mathbf{p}_{ini}$ is minimum. Let $S(\mathbf{w}_{ini})$ be the neighborhood of the n closest weight vectors to $\mathbf{w}_{ini}$ (where n is the number of decision variables of the MOP). Then, the simplex employed by the local search is defined as:

$$\Delta = \{\mathbf{p}_{ini}, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$$

which is built in two different ways by using a probability P_s, according to the two following strategies:

Since the dimensionality of the simplex depends of the number of decision variables of the MOP, the population size of the MOEA needs to be larger than the number of decision variables.

i. *Neighboring solutions:* The remaining n solutions $\mathbf{p}_i \in P$ $(i = 1, \ldots, n)$ are chosen, such that, $\mathbf{p}_i$ minimizes each subproblem defined by each weight vector in $S(\mathbf{w}_{ini})$. This is the same strategy employed for constructing the simplex used in MONSS, see Chapter 5.

ii. *Sample solutions:* The remaining n solutions $\mathbf{p}_i \in \Omega$ $(i = 1, \ldots, n)$ are generated by using a low-discrepancy sequence. The Hammersley sequence [55] is adopted in this work, to generate a well-distributed sampling of solutions in a determined search space. As in [146], we use a strategy based on the genetic analysis of a sample from the current population for reducing the search space. However, here, we compute the average $(\mathbf{m})$ and standard deviation (σ) of the chromosomes (solutions) that minimize each subproblem defined by the weight vectors in $S(\mathbf{w}_{ini})$. In this way, the new bounds are defined by:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma \end{aligned}$$

where $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively.

Once the search space has been reduced, the n remaining solutions are generated by means of the Hammersley sequence using as bounds $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$.

Here, we use $P_s = 0.3$ as the probability that the construction of the simplex using sample solutions is chosen. Otherwise, the construction using neighboring solutions is employed.

6.1.2.3 *Defining the Search Direction*

Let $B(\mathbf{w}_{ini})$ be the neighborhood of the T closest weight vectors to $\mathbf{w}_{ini}$, such that $\mathbf{w}_{ini}$ defines the subproblem for which the initial search solution $\mathbf{p}_{ini}$ is minimum. Let $D(\mathbf{w}_{ini})$ be the A_r closest weight vectors to $\mathbf{w}_{ini}$.

The nonlinear simplex search focuses on minimizing a subproblem defined by the weight vector $\mathbf{w}_{obj}$, which is defined according to the following rules:

i. The farthest weight vector in $B(\mathbf{w}_{ini})$ to $\mathbf{w}_{ini}$, if it is the first iteration of the local search,

ii. otherwise, a random weight vector taken from $D(\mathbf{w}_{ini})$ is employed.

It is noteworthy that (in *ii*) the search is relaxed defining as our action range the A_r weight vectors closest to $\mathbf{w}_{ini}$. The idea of relaxing the search is taken from the MONSS framework. However, the neighborhood $D(\mathbf{w}_{ini})$ is used instead of a partition as in MONSS. Here, we used $A_r = 5$ as the size of the action range for the local search.

6.1.2.4 *Deforming the Simplex*

At each iteration of the local search, the $n + 1$ vertices of the simplex Δ are sorted according to their value for the subproblem that it tries to minimize (the best value is the first element). In this way, a movement into the simplex is performed for generating the new solution $\mathbf{p}_{new}$. The movements are calculated according to the equations provided by Nelder and Mead in [102] (see Section 5.1), however, in order to save objective function evaluations and to avoid the search collapses, the shrinkage step is omitted. Each movement is controlled by three scalar parameters: reflection (ρ), expansion (χ) and contraction (γ).

The NSS algorithm was conceived to deal with unbounded problems. When dealing with bounded variables, the created solutions can be located outside the allowable bounds after any movement of the

NSS algorithm. In order to deal with this, we bias the new solution if any component of $\mathbf{p}_{new}$ lies outside the bounds according to:

$$\mathbf{p}_{new}^{(j)} = \begin{cases} \mathbf{L}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} < \mathbf{L}_{bound}^{(j)} \\ \mathbf{U}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} > \mathbf{U}_{bound}^{(j)} \\ \mathbf{p}_{new}^{(j)} & \text{, otherwise.} \end{cases} \tag{6.2}$$

where $\mathbf{L}_{bound}^{(j)}$ and $\mathbf{U}_{bound}^{(j)}$ are the lower and upper bounds of the j^{th} parameter of $\mathbf{p}_{new}$, respectively.

6.1.2.5 *Updating the Population*

The information provided by the local search engine is introduced to MOEA/D using a Lamarckian evolution scheme [139]. However, since we are dealing with MOPs, the new solution generated by the local search mechanism could be better than more than one solution in the current population. For this, we adopt the following mechanism in which some solutions from the population could be replaced:

Let P be the current population reported by the MOEA. Let $\mathbf{p}_{new}$ be the solution generated by any movement of the simplex search. Let $B(\mathbf{w}_{obj})$ and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ be the neighborhood of the T closest weight vectors to $\mathbf{w}_{obj}$, and the well-distributed set of all weight vectors, respectively. We define

$$Q = \begin{cases} B(\mathbf{w}_{obj}) & \text{, if } r < \delta \\ W & \text{otherwise} \end{cases}$$

where r is a random number having uniform distribution. In this work, we use $\delta = 0.9$.

The current population P is updated by replacing at most R_{ls} solutions from P such that, $g(\mathbf{p}_{new}|\mathbf{w}_i, z) < g(\mathbf{x}_i|\mathbf{w}_i, z)$, where $\mathbf{w}_i \in Q$ and $\mathbf{x}_i \in P$, such that $\mathbf{x}_i$ minimizes the subproblem defined by $\mathbf{w}_i$.

Note that the loss of diversity is avoided by replacing a maximum number of solutions from P, instead of all the solutions that minimize the subproblems defined by the complete neighborhood Q, as in MOEA/D. In our study, we set $R_{ls} = 15$ as the maximum number of solution to replace.

6.1.2.6 *Stopping Criterion*

A maximum number of fitness function evaluations E_{ls} is adopted as our stopping criterion. If the nonlinear simplex search overcomes this maximum number of evaluations, the simplex search is stopped and the evolutionary process of MOEA/D continues. However, the search could be inefficient if the simplex has been deformed so that it has collapsed into a region where there are local minima. According to Lagarias et al. [82] the simplex search finds a better solution in at most $n+1$ iterations (at least in convex functions with low dimensionality). where n is the number of decision variables of the MOP. Thus, we have considered this observation and adopt a stopping criterion for reconstructing the simplex by using another nondominated solution from P as simplex head. Therefore, if the simplex search does not find a minimum value in $n+1$ iterations, we reset the search by going to **Step 2.2.1**. Otherwise, we perform other movement into the simplex using a new search direction, i.e., by going to **Step 2.2.3**.

6.2 Experimental Study

6.2.1 Test Problems

In order to assess the performance of our proposed memetic algorithm, we compare its results with respect to those obtained by the original MOEA/D [155]. We adopted 12 test problems whose $\mathcal{PF}$ have different characteristics including convexity, concavity, disconnections and multi-modality. In the following, we describe the test suites that we have adopted.

- *Zitzler-Deb-Thiele (ZDT)* test suite [158]. The five bio-objective MOPs (except for ZDT5, which is a discrete problem) were adopted. We used 30 decision variables for ZDT1 to ZDT3, while ZDT4 and ZDT6 were tested using 10 decision variables.

- *Deb-Thiele-Laumanns-Zitzler (DTLZ)* test suite [28, 29]. The seven unconstrained MOPs were adopted. DTLZ1 was tested using 7 decision variables. For DTLZ2 to DTLZ6, we employed 12 decision variables, while DTLZ7 was tested using 22 decision

variables. The algorithms were tested by using three objective functions for each MOP.

The mathematical description of these two test suites can be seen in Appendices A.2 and A.3, respectively.

6.2.2 Performance Measures

In order to assess the performance of our proposed MOEA/D+LS, we compared it with respect to the original MOEA/D using the *Hypervolume* (I_H) and the *Two Set Coverage* (I_C) performance measures. The characteristics of such performance measures were presented in Chapter 3, and we refer to section 3.4 for a more detailed description of these performance indicators.

6.2.3 Parameters Settings

As indicated before, we compared our proposed MOEA/D+LS with respect to MOEA/D (using the PBI approach). The weight vectors for the algorithms were generated as in [155], i.e., the setting of N and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ is controlled by a parameter H. More precisely, $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}$$

Therefore, the number of such vectors in W is given by:

$$N = C^{k-1}_{H+k-1}.$$

where k is the number of objective functions.

Both MOEA/D+LS and MOEA/D, were tested with $H = 99$ for the bi-objective problems, i.e. 100 weight vectors. $H = 23$ was used for the three-objective problems, i.e. 300 weight vectors. For a fair comparison, the set of weight vectors was the same for both algorithms.

For each MOP, 30 independent runs were performed with each algorithm. The parameters for both algorithms are summarized in Table 4, where N represents the number of initial solutions (100 for bi-objective problems and 300 for three-objective problems). N_{it} represents the maximum number of iterations, which was set to 100

for all test problems. Therefore, both algorithms performed 10,000 (for the bi-objective problems) and 30,000 (for the three-objective problems) fitness function evaluations for each problem. For MOEA/D+LS, ρ, χ and γ represent the control parameters for the reflection, expansion and contraction movements of the NSS, respectively. The parameters T_n, η_c, η_m, P_c and P_m represent the neighborhood size, crossover index (for *Simulated Binary Crossover (SBX)*), mutation index (for *Polynomial-Based Mutation (PBM)*), crossover rate and mutation rate, respectively. A_r, R_{ls} and E_{ls} represent the action range, the number of solutions to be replaced and the maximum number of fitness function evaluations employed by the local search mechanism, respectively.

Finally, the parameter θ, represents the penalty value used in the PBI approach for both MOEA/D+LS and MOEA/D.

Parameter	MOEA/D+LS	MOEA/D
N	100/300	100/300
N_{it}	100	100
T_n	20	20
η_c	20	20
η_m	20	20
P_c	1	1
P_m	$1/n$	$1/n$
α	1	–
β	2	–
γ	1/2	–
A_r	5	–
R_{ls}	15	–
E_{ls}	300	–
θ	5	5

Table 4.: Parameters for MOEA/D+LS and MOEA/D

For each MOP, the algorithms were evaluated using the two performance measures described in section 3.4 (i.e., the *Hypervolume* (I_H) and *Two Set Coverage* (I_C) indicators). The results obtained are summarized in Tables 5 and 6. These tables display both the *average* and the standard deviation (σ) of each performance measure for each MOP. The reference vectors used for computing the I_H performance measure are shown in Table 5. These vectors are established close to the individual minima for each MOP, i.e., close to the extremes of the $\mathcal{PF}$. With that, a good measure of approximation and spread is reported when the algorithms converge along the $\mathcal{PF}$. In the case of the statistics for the I_C performance measure comparing pairs of algorithms—i.e. $I_C(A, B)$, they were obtained as average values of the

comparison of all the independent runs from the first algorithm with respect to all the independent runs from the second algorithm. For an easier interpretation, the best results are presented in **boldface** for each performance measure and test problem adopted.

MOP	MOEA/D+LS average (σ)	MOEA/D average (σ)	reference vector $\mathbf{r}$
ZDT1	**0.819246** (0.038088)	0.751315 (0.033339)	$(1.1, 1.1)^T$
ZDT2	**0.384962** (0.151212)	0.210410 (0.080132)	$(1.1, 1.1)^T$
ZDT3	**0.995692** (0.158499)	0.990212 (0.089499)	$(1.1, 1.1)^T$
ZDT4	0.169257 (0.212639)	**0.600217** (0.138989)	$(1.1, 1.1)^T$
ZDT6	**0.462559** (0.050484)	0.425904 (0.010630)	$(1.1, 1.1)^T$
DTLZ1	0.316904 (0.001091)	**0.317249** (0.000957)	$(0.7, 0.7, 0.7)^T$
DTLZ2	0.768621 (0.000466)	**0.768696** (0.000644)	$(1.1, 1.1, 1.1)^T$
DTLZ3	0.221197 (0.282045)	**0.383622** (0.245603)	$(1.1, 1.1, 1.1)^T$
DTLZ4	**0.768966** (0.000664)	0.768935 (0.000645)	$(1.1, 1.1, 1.1)^T$
DTLZ5	**0.426307** (0.000167)	0.426115 (0.000675)	$(1.1, 1.1, 1.1)^T$
DTLZ6	**0.426345** (0.000714)	0.000228 (0.001226)	$(1.1, 1.1, 1.1)^T$
DTLZ7	**1.922224** (0.012057)	1.916040 (0.016969)	$(1.1, 1.1, 6.1)^T$

Table 5.: Results of I_H for MOEA/D+LS and MOEA/D

6.3 Numerical Results

As indicated before, the results obtained by the proposed MOEA/D+LS were compared against those produced by the original MOEA/D. According to the results presented in Tables 5 and 6, MOEA/D+LS had a better performance than MOEA/D in most of the MOPs adopted. These tables provide a quantitative assessment of the performance of MOEA/D+LS in terms of the I_H and I_C indicators. That means that the solutions obtained by MOEA/D+LS achieved a better approximation to the $\mathcal{PF}$ than those solutions obtained by

MOP	I_C(MOEA/D+LS, MOEA/D)	I_C(MOEA/D, MOEA/D+LS)
	average (σ)	average (σ)
ZDT1	**0.893657** (0.122230)	0.004889 (0.011666)
ZDT2	**0.432435** (0.149436)	0.001333 (0.007180)
ZDT3	0.667901 (0.021117)	**0.690476** (0.093046)
ZDT4	0.000000 (0.000000)	**1.000000** (0.000000)
ZDT6	0.170720 (0.028694)	**0.867949** (0.036735)
DTLZ1	**0.155326** (0.165805)	0.126444 (0.093361)
DTLZ2	0.120572 (0.028892)	**0.150281** (0.031948)
DTLZ3	**0.469164** (0.376265)	0.227174 (0.260595)
DTLZ4	**0.178360** (0.033641)	0.077111 (0.019450)
DTLZ5	**0.033682** (0.022515)	0.031905 (0.022034)
DTLZ6	**1.000000** (0.000000)	0.000000 (0.000000)
DTLZ7	**0.122837** (0.021196)	0.108987 (0.016292)

Table 6.: Results of I_C for MOEA/D+LS and MOEA/D

MOEA/D when a low number of fitness function evaluations was adopted.

However, for ZDT4, DTLZ1, DTLZ2 and DTLZ3, the I_H indicator showed that the local search did not improve the performance of MOEA/D. In contrast, for DTLZ2, MOEA/D was not significantly better than the memetic algorithm, and for the case of ZDT4, DTLZ1 and DTLZ3, MOEA/D+LS was significantly outperformed by MOEA/D. The poor performance of MOEA/D+LS for these problems (ZDT4, DTLZ1 and DTLZ3) is attributed to their high multi-frontality. For a more detailed description of these problems see Appendices A.2 and A.3. The effectiveness of MONSS when dealing with unimodal optimization problems having low dimensionality has been shown in Chapter 5. Here, we have designed a local search mechanism based on the MONSS framework for dealing with MOPs with higher dimensionality (in decision variable space). However, when dealing with multi-frontal MOPs, the convergence of the simplex search considerably slows down and may even fail.

Regarding the I_C performance measure, MOEA/D+LS obtained better results than those produced by MOEA/D in the majority of the test problems adopted. This means that the solutions obtained by MOEA/D+LS dominated a higher portion of the solutions produced by MOEA/D. However, MOEA/D was better for ZDT3, ZDT4, ZDT6 and DTLZ2, although the ratio of solutions dominated by MOEA/D was not significantly high for DTLZ2. Although the I_C performance measure benefits MOEA/D in ZDT3 and ZDT6, it is worth noting that our proposed MOMA reached better results regarding the I_H performance measure in those problems. I_H not only measures the convergence but also the maximum spread of solutions along the $\mathcal{PF}$, which is the reason why our MOEA/D+LS obtained better results regarding this performance measure. High multi-frontality, however, remains as a limitation of our proposed approach. This can be exemplified in ZDT4, in which our proposed approach was clearly outperformed by the original MOEA/D with respect to the two performance measures adopted in our study.

6.4 Remarks

In this Chapter, we have presented a hybridization of MOEA/D with a NSS algorithm, in which the former acts as the global search engine, and the latter works as a local search engine. The local search mechanism is based on the MONSS framework, which adopts a decomposition approach similar to the one used in MOEA/D. Therefore, its use could be easily coupled within other decomposition-based MOEAs, such as those reported in [107, 98, 149]. Our proposed MOEA/D+LS, was found to be competitive with respect to the original MOEA/D over a set of test functions taken from the specialized literature, when performing 10,000 and 30,000 fitness function evaluations, for problems having two and three objectives, respectively. The use of a low number of fitness function evaluations in MOEAs is an important issue in multi-objective optimization, because there are several real-world problems that are computationally expensive to solve. We consider that the strategy employed to hybridize the MONSS framework with MOEA/D was, in general, appropriate for dealing with the MOPs adopted here.

In the next Chapter, we focus on improving the local search mechanism adopted here. We hypothesized that the use of an appropriate simplex and a good hybridization strategy could be a powerful combination for solving complex and computationally expensive MOPs (see for example [63, 155]). Given the nature of the methods used here (they do not require gradient information), the use of this hybrid approach could be an efficient alternative when dealing with some real-world applications for which the gradient information is not available. This is the reason why hybridizing non-gradient mathematical programing methods with MOEAs is an important research area that is worth exploring.

7

An Improved Multi-objective Memetic Algorithm Based on Decomposition

In the previous Chapter, we presented a *Multi-Objective Memetic Algorithm (MOMA)* which hybridized the *Multi-objective Nonlinear Simplex Search (MONSS)* approach with the well-known *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*. The resulting *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS)* was found to be a competitive algorithm, and in some cases, it turned out to be significantly better than the original MOEA/D on the test problems adopted. In the design of MOMAs, some important decisions should be considered, such as:

1. *When should the local search be performed?*
2. *What search direction should be taken?* and
3. *How should the knowledge of the local search mechanism be introduced into the evolutionary algorithm?*

The good performance of a MOMA depends mainly on giving appropriate answers to these questions.

MOEA/D+LS performs the local search procedure after each iteration of MOEA/D, if and only if, the percentage of nondominated solutions in the population is less than 50%. The use of this strategy, answers the first question posed above. MOEA/D+LS decomposes a *Multi-objective Optimization Problem (MOP)* into several single-objective optimization problems. Such problems are defined by a well-distributed set of weight vectors. The local search used by MOEA/D+LS, directs the search towards different neighborhoods of the whole set of weight vectors. In this way, the second question is

solved. The third question is answered by updating the solutions in the population that solve the problems defined by the neighborhood of weight vectors.

In this Chapter, we investigate an alternative strategy for hybridizing the *Nonlinear Simplex Search (NSS)* with the MOEA/D. Similar to MOEA/D+LS, the MOMA presented in this chapter, incorporates the Nelder and Mead method [102] as a local search engine into the well-known MOEA/D [155]. However, in order to improve the local search, some modifications have been introduced. Such modifications attend the three above questions in different ways, as done by MOEA/D+LS. With that, an improved version of MOEA/D+LS presented in the previous chapter, is introduced. In the following, we present in detail the components of the improved MOEA/D+LS.

7.1 The Proposed Approach

7.1.1 General Framework

As indicated before, the MOMA presented here, adopts MOEA/D [155] as its baseline algorithm. The local search mechanism is based on Nelder and Mead's method [102]. In this way, the proposed *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II)* explores the global search space using MOEA/D, while the local search engine exploits the promising regions provided by MOEA/D.

Similar to MOEA/D+LS, MOEA/D+LS-II decomposes a MOP into several single-objective optimization problems. Such optimization problems are defined by a set of weight vectors. If the weight vectors are evenly distributed, a good representation of the *Pareto front (PF)* could be reached. Therefore, before starting the search, a well-distributed set of weight vectors needs to be generated. Here, we employ the *Penalty Boundary Intersection (PBI)* approach to transform a MOP into a single-objective optimization problem, which consists in minimizing:

See Section 3.2.2 for a more detailed description of the PBI approach

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^{\star}) = d_1 + \theta d_2 \tag{7.1}$$

such that:

$$d_1 = \frac{||(F(\mathbf{x}) - \mathbf{z}^\star)^T \mathbf{w}||}{||\mathbf{w}||}$$
$$\text{and} \quad d_2 = \left|\left|(F(\mathbf{x}) - \mathbf{z}^\star) - d_1 \frac{\mathbf{w}}{||\mathbf{w}||}\right|\right|$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^n$, θ is the penalty value and $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ is the utopian vector, i.e., $z_i^\star = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}, \forall i = 1, \ldots, k$.

At each iteration, MOEA/D+LS-II performs an iteration of MOEA/D (see Algorithm 8). After that, the offspring population produced by MOEA/D is then improved by using the local search procedure. For a better understanding of the proposed approach, Algorithm 16 presents the general framework of the proposed MOEA/D+LS-II. **Step 3** refers to the complete local search mechanism which is performed after each iteration of MOEA/D. In the following sections, we describe in detail the components of the improved local search mechanism.

7.1.2 Local Search Mechanism

MOEA/D+LS-II exploits the promising neighborhood of the solutions found by MOEA/D at each generation. As it was mentioned before, MOEA/D+LS-II uses Nelder and Mead's method as a local search engine for continuous search spaces, in order to improve the solutions provided by MOEA/D. In contrast to MOEA/D+LS, the local search mechanism of MOEA/D+LS-II approximates solutions to the extremes and the maximun bulge (sometimes called knee) of the $\mathcal{PF}$. Instead of using the neighborhoods as MOEA/D+LS does. The NSS is employed for minimizing a subproblem defined by a weighting vector using the PBI approach. In the following, we present in detail the components of our local search engine outlined in Algorithms 16 and 17.

Algorithm 16: The *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II* (MOEA/D+LS-II)

Input:
a stopping criterion;
N: the number of the subproblems considered in MOEA/D+LS-II;
W: a well-distributed set of weighting vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$;
T: the neighborhood size of each weight vector;
S_t: the similarity threshold for the local search;
E_{ls}: the maximum number of evaluations for the local search.
Output:
P: the final population found by MOEA/D+LS-II.

begin
  Step 1. Initialization:
  Generate an initial population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ randomly; $FV^i = \mathbf{F}(\mathbf{x}_i)$; $B(\mathbf{w}_i) = \{\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}\}$ where $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}$ are the T closest weighting vectors to $\mathbf{w}_i$, for each $i = 1, \ldots, N$; $\mathbf{z} = (+\infty, \ldots, +\infty)^T$;
  Step 2. The Memetic Algorithm:
  while *stopping criterion is not satisfied* **do**
    Step 2.1. MOEA/D **iteration:** Perform **Step 2** of the MOEA/D framework for obtaining P (the next population), see Algorithm 8.
    Step 3. The Local Search Mechanism:
    for $j = 1, \ldots, k+1$ **do**
      Step 3.1. Defining the Search Direction:.
      if $j < k$ **then**
        // Search towards the extremes of the $\mathcal{PF}$
        $\mathbf{w_s} = \mathbf{e}^j$, where $\mathbf{e}^j$ is the j^{th} canonical basis in $\mathbb{R}^k$ and k is the number of objective functions.
      else
        // Search towards the maximum bulge of the $\mathcal{PF}$
        $\mathbf{w_s} = (1/k, \ldots, 1/k)$
      end
      Step 3.2. Selecting initial solution: Select the initial solution for the local search according to Section 7.1.2.2.
      Step 3.3. Local Search: Apply nonlinear simplex search according to Algorithm 17.
    end
  end
  return *P*;
end

Algorithm 17: Use of Local Search for the MOEA/D+LS-II

Input:
a stopping criterion;
P: the current population of the MOEA/D+LS-II;
S_t: the similarity threshold for the local search;
E_{ls}: the maximum number of evaluations for the local search.
Output:
P: the updated population P.

```
1  begin
2      Step 1. Checking Similarity: Obtain the similarity (S_ls) between p_ini
       and the previous initial solution (p'_ini) for the local search—see
       Section 8.3.1.3;
3      if there are enough resources and S_t < S_ls then
4          Step 2. Building the Simplex: Build the initial simplex for the
           nonlinear simplex search—see Section 8.3.1.4;
5          Step 3. Deforming the Simplex: Perform any movement
           (reflection, contraction or expansion) for obtaining p_new according
           to Nelder and Mead's method—see Section 8.3.1.5;
6          Step 4. Updating the Population: Update the population P using
           the new solution p_new according to the rules presented in
           Section 8.3.1.6.
7          Step 5. Stopping Criterion: If the stopping criterion is satisfied
           then stop the local search. Otherwise, go to Step 3—see
           Section 8.3.1.7.
8      end
9      return P; // The updated population P
10 end
```

7.1.2.1 *Defining the Search Direction*

In contrast to the strategy employed by MOEA/D+LS, the local search mechanism proposed here, approximates solutions to the $\mathcal{PF}$ in two different stages:

1. Initially, the search is directed to the extremes of the $\mathcal{PF}$. Therefore, the weight vectors that define the subproblems that approximate solutions (when they are solved) to the extremes are defined by the canonical basis in $\mathbb{R}^k$—i.e., the search direction that approximates solutions to the j^{th} extreme of the $\mathcal{PF}$ is defined by the weighting vector:

Assuming the use of the PBI approach.

$$\mathbf{w}_s = \mathbf{e}^j$$

where $\mathbf{e}^j$ is the j^{th} canonical vector in $\mathbb{R}^k$ and $j = 1, \ldots, k$ (where k is the number of objective functions).

2. Once the solutions lying at the extremes of the $\mathcal{PF}$ have been approximated, the local search is focused on minimizing the subproblem that approximates the solutions lying on the knee of the $\mathcal{PF}$. Therefore, the search direction is now defined by the weight vector:

$$\mathbf{w}_s = (1/k, \ldots, 1/k)^T$$

where k is the number of objective functions.

Considering the use of the PBI approach, the penalty value θ is set as $\theta = 5$ for approximating solutions to the extremes, whereas for the knee, a value $\theta = 10$ is employed.

7.1.2.2 *Selecting Initial Solution*

Let P be the set of solutions found by MOEA/D at any generation. Let $\mathbf{w}_s$ be the weighting vector that defines the search direction for the NSS. The solution $\mathbf{p}_{ini}$ which starts the search is defined by:

$$\mathbf{p}_{ini} = \mathbf{x} \in P, \text{ such that minimizes: } g(\mathbf{x}|\mathbf{w}_s, \mathbf{z}^\star)$$

Solution $\mathbf{p}_{ini}$ represents not only the initial search point, but also the simplex head from which the simplex will be built.

7.1.2.3 *Checking Similarity*

The NSS explores the neighborhood of the solution $\mathbf{p}_{ini} \in P$. Since the simplex search is applied after each iteration of MOEA/D, most of the time, the initial solution $\mathbf{p}_{ini}$ does not change its position from one generation to another. For this reason, the proposed local search mechanism stores a record ($\mathbf{p}'_{ini}$) of the last position from which the nonlinear simplex search starts. At the beginning of the execution of MOEA/D+LS-II, the initial position record is set as empty, that is: $\mathbf{p}'_{ini} = \emptyset$. Once the simplex search is performed, the initial solution is stored in the historical record, i.e., $\mathbf{p}'_{ini} = \mathbf{p}_{ini}$. In this way, for the next call of the local search, a previous comparison of similarity is performed. That is, the local search will be performed, if and only if,

$\|\mathbf{p}_{ini} - \mathbf{p}'_{ini}\| > S_t$, where S_t represents the similarity threshold. Since in the first iteration of the simplex search there is no previous record of the initial solution, the simplex search is automatically performed. Both the updating of the historical record and the similarity operator are performed for each initial solution $\mathbf{p}_{ini}$ which minimizes the subproblem defined by $\mathbf{w}_s$. In our study, we adopted a similarity threshold $S_t = 0.001$.

This strategy to employ the local search is the main difference with respect to MOEA/D+LS, where local search is applied when the population has less than 50% of nondominated solutions.

7.1.2.4 *Building the Simplex*

Let $\mathbf{w}_{ini}$ be the weighting vector that defines the subproblem for which the initial search point $\mathbf{p}_{ini}$ is minimum. Let $S(\mathbf{w}_{ini})$ be the neighborhood of the n closest weighting vectors to $\mathbf{w}_{ini}$ (where n is the number of decision variables of the MOP). Then, the simplex defined as:

$$\Delta = \{\mathbf{p}_{ini}, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$$

is built in two different ways, depending on the direction on which the simplex search is focused.

i. For the extremes of the $\mathcal{PF}$: The remaining n solutions $\mathbf{p}_i \in \Omega$ $(i = 1, \ldots, n)$ are generated by using a low-discrepancy sequence. In this work, we adopted the Hammersley sequence [55] to generate a well-distributed sampling of solutions in a determined search space. In an analogous way to MOEA/D+LS, we use a strategy based on the genetic analysis of a sample from the current population for reducing the search space. Therefore, we compute the average $(\mathbf{m})$ and standard deviation (σ) of the chromosomes (solutions) that minimize each subproblem defined by the weight vectors in $S(\mathbf{w}_{ini})$. In this way, the new bounds are defined by:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma \end{aligned}$$

where $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively. Once the search space has been reduced, the n remaining solutions are generated by means of the Hammersley sequence using as bounds $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$.

ii. For the knee of the $\mathcal{PF}$: The remaining n solutions $\mathbf{p}_i \in P$ ($i = 1, \ldots, n$) are chosen, such that, $\mathbf{p}_i$ minimizes each subproblem defined by each weighting vector in $S(\mathbf{w}_{ini})$. This is the same strategy employed in MONSS for constructing the simplex.

Note however that, since the dimensionality of the simplex depends of the number of decision variables of the MOP, the population size of MOEA/D+LS-II needs to be larger than the number of decision variables.

7.1.2.5 *Deforming the Simplex*

Let $\mathbf{w}_s$ be the weighting vector that defines the search direction for the local search. Let Δ be the simplex defined by the above description. NSS will be focused on minimizing the subproblem defined by the weighting vector $\mathbf{w}_s$. At each iteration of the nonlinear simplex search, the $n+1$ vertices of the simplex Δ are sorted according to their value for the subproblem that it tries to minimize (the best value is the first element). In this way, a movement into the simplex is performed for generating the new solution $\mathbf{p}_{new}$. The movements are calculated according to the equations provided by Nelder and Mead, see Section 5.1. However, the shrinkage step is omitted. Each movement is controlled by three scalar parameters: **reflection** (ρ), **expansion** (χ) and **contraction** (γ).

NSS was conceived for unbounded problems. When dealing with bounded variables, the created solutions can be located outside the allowable bounds after some movements of the simplex search. In order to deal with this, we bias the new solution if any component of $\mathbf{p}_{new}$ lies outside the bounds according to:

$$\mathbf{p}_{new}^{(j)} = \begin{cases} \mathbf{L}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} < \mathbf{L}_{bound}^{(j)} \\ \mathbf{U}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} > \mathbf{U}_{bound}^{(j)} \\ \mathbf{p}_{new}^{(j)} & \text{, otherwise.} \end{cases} \tag{7.2}$$

where $\mathbf{L}_{bound}^{(j)}$ and $\mathbf{U}_{bound}^{(j)}$ are the lower and upper bounds of the j^{th} parameter of $\mathbf{p}_{new}$, respectively. This is the same strategy employed by MOEA/D+LS.

7.1.2.6 *Updating the Population*

The information provided by the local search mechanism is introduced into the population of MOEA/D. Since we are dealing with MOPs, the new solution generated by any movement of the NSS could be better than more than one solution in the current population. Thus, we adopt the following mechanism in which more than one solution from the population could be replaced.

Let P be the current population reported by the MOEA/D+LS-II. Let $\mathbf{p}_{new}$ be the solution generated by any movement of the NSS. Let $B(\mathbf{w}_s)$ and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ be the neighborhood of the T closest weighting vectors to $\mathbf{w}_s$, and the well-distributed set of all weight vectors, respectively. We define

$$Q = \begin{cases} B(\mathbf{w}_s) & \text{, if } r < \delta \\ W & \text{otherwise} \end{cases}$$

where r is a random number having uniform distribution. In this work, we use $\delta = 0.5$.

The current population P is updated by replacing at most R_{ls} solutions from P such that, $g(\mathbf{p}_{new}|\mathbf{w}_i, z) < g(\mathbf{x}_i|\mathbf{w}_i, z)$, where $\mathbf{w}_i \in Q$ and $\mathbf{x}_i \in P$, such that $\mathbf{x}_i$ minimizes the subproblem defined by $\mathbf{w}_i$.

In this way, the loss of diversity is avoided by replacing a maximum number of solutions from P, instead of all the solutions that minimize the subproblems defined by the complete neighborhood Q. In our study, we set $R_{ls} = 15$ as the maximum number of solution to replace. This strategy of updating also differs to the one proposed by MOEA/D+LS, where we only considered the neighborhood of solutions from which the local search is directed.

7.1.2.7 *Stopping Criterion*

The local search mechanism encompasses the search of solutions towards both the extremes and the knee of the $\mathcal{PF}$. This mechanism is limited to a maximum number of fitness function evaluations defined by E_{ls}. In this way, the proposed local search has the following stopping criteria:

1. If the nonlinear simplex search overcomes the maximum number of evaluations (E_{ls}), the simplex search is stopped and the evolutionary process of MOEA/D continues by going to **Step 2** of Algorithm 8.

2. The search could be inefficient if the simplex has been deformed so that it has collapsed into a region in which there are no local minima. According to Lagarias et al. [82] the simplex search finds a better solution in at most $n+1$ iterations (at least in convex functions with low dimensionality). Therefore, if the simplex search does not find a better value for the subproblem defined by $\mathbf{w}_s$ in $n+1$ iterations, we stop the search and continue with the next direction defined by going to **Step 3.1** of Algorithm 16. Otherwise, we perform other movement into the simplex by going to **Step 3** of Algorithm 17.

7.2 Experimental Results

7.2.1 Test Problems

In order to assess the performance of our proposed memetic algorithm, we compare its results with respect to those obtained by the original MOEA/D [155] and the proposed MOEA/D+LS. We adopted 21 test problems whose $\mathcal{PF}$s have different characteristics including convexity, concavity, disconnections and multi-modality. In the following, we describe the test suites that we have adopted.

- *Zitzler-Deb-Thiele (ZDT)* test suite [158]. The five bio-objective MOPs (except for ZDT5, which is a discrete problem) were adopted. We used 30 decision variables for ZDT1 to ZDT3, while ZDT4 and ZDT6 were tested using 10 decision variables.

- *Deb-Thiele-Laumanns-Zitzler (DTLZ)* test suite [28, 29]. The seven unconstrained MOPs were adopted. DTLZ1 was tested using 7 decision variables. For DTLZ2 to DTLZ6, we employed 12 decision variables, while DTLZ7 was tested using 22 decision variables. For all problems we tested the algorithms using three objective functions for each MOP.

- *Walking-Fish-Group (WFG)* test suite [63]. The nine MOPs from this test suite were adopted. We used $k = 4$ for the position related parameters and $l = 20$ for the distance related parameters—i.e., 24 decision variables (as it was suggested by Huband et al. [63])—adopting three objective functions for each MOP.

The mathematical description of these three test suites can be found in Appendices A.2, A.3 and A.4, respectively.

7.2.2 Performance Measures

To assess the performance of our proposed MOEA/D+LS-II and the other two *Multi-Objective Evolutionary Algorithms (MOEAs)* (i.e., the original MOEA/D and MOEA/D+LS) on the test problems adopted, the *Hypervolume* (I_H) indicator was employed. This performance measure is Pareto compliant [162], and quantifies both approximation and maximum spread of nondominated solutions along the $\mathcal{PF}$. In order to compare the quality of solutions between two sets of non-dominated solutions, the *Two Set Coverage* (I_C) indicator was employed.

For a more detailed description of the adopted performance measures, the interested reader is referred to Section 3.4.

7.2.3 Parameters Settings

We compared the results obtained by our proposed MOEA/D+LS-II with respect to those obtained by MOEA/D and MOEA/D+LS (using the PBI approach). The weight vectors for the algorithms were generated as in [155], i.e., the setting of N and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ is controlled by a parameter H. More precisely, $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}$$

Therefore, the number of such vectors in W is given by:

$$N = C_{H+k-1}^{k-1}.$$

where k is the number of objective functions.

Both MOEA/D+LS and MOEA/D, were tested with $H = 99$ for the bi-objective problems, i.e., 100 weight vectors. $H = 23$ was used for the three-objective problems, i.e., 300 weight vectors. For a fair comparison, the set of weight vectors was the same for both algorithms.

For each MOP, 30 independent runs were performed with each algorithm. The parameters for the algorithms are summarized in Table 7, where N represents the number of initial solutions (100 for

bi-objective problems and 300 for three-objective problems). N_{it} represents the maximum number of iterations, which was set to 100 for all test problems. Therefore, both algorithms performed 10,000 (for the bi-objective problems) and 30,000 (for the three-objective problems) fitness function evaluations for each problem. The parameters T_n, η_c, η_m, P_c and P_m represent the neighborhood size, crossover index (for *Simulated Binary Crossover (SBX)*), mutation index (for *Polynomial-Based Mutation (PBM)*), crossover rate and mutation rate, respectively. For MOEA/D+LS-II and MOEA/D+LS, ρ, χ and γ represent the control parameters for the reflection, expansion and contraction movements of the NSS, respectively. R_{ls} and E_{ls} represent the number of solutions to be replaced and the maximum number of fitness function evaluations employed by the local search engine, respectively. A_r and S_t, represent the action range and the similarity threshold employed by the local search for MOEA/D+LS and MOEA/D+LS-II, respectively. Finally, the parameter θ, represents the penalty value used in the PBI approach for the three approaches compared herein.

Parameter	MOEA/D	MOEA/D+LS	MOEA/D+LS-II
N	100/300	100/300	100/300
N_{it}	100	100	100
T_n	20	20	20
η_c	20	20	20
η_m	20	20	20
P_c	1	1	1
P_m	$1/n$	$1/n$	$1/n$
α	–	1	1
β	–	2	2
γ	–	1/2	1/2
R_{ls}	–	15	15
E_{ls}	–	300	300
A_r	–	5	–
S_t	–	–	0.001
θ	5	5	5

Table 7.: Parameters for MOEA/D, MOEA/D+LS and MOEA/D+LS-II

For each MOP, the algorithms were evaluated using the I_H and I_C indicators. The results of such indicators are summarized in Tables 8 and 9, respectively. These tables display both the *average* and the standard deviation (σ) of each performance measure for each MOP. The reference vectors used for computing the I_H performance measure are shown in Table 8. These vectors are established close to the individual minima for each MOP, i.e., close to the extremes of the $\mathcal{PF}$. With that, a good measure of approximation and spread

is reported when the algorithms converge along the $\mathcal{PF}$. In the case of the statistics for the I_C performance measure comparing pairs of algorithms (i. e.$I_C(A, B)$), they were obtained as average values of the comparison of all the independent runs from the first algorithm with respect to all the independent runs from the second algorithm. For an easier interpretation, the best results are presented in **boldface** for each performance measure and test problem adopted.

7.3 Numerical Results

As indicated before, the results obtained by the proposed MOEA/D+LS-II were compared against those produced by the original MOEA/D and MOEA/D+LS. In the following, we present the results obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT, DTLZ and WFG test suites. The results for each test suite, are presented in a separate way for an easier understanding.

7.3.1 Results for the ZDT test suite

Hypervolume (I_H) Performance Measure. According to Table 8, the proposed MOEA/D+LS-II obtained better results in terms of the I_H indicator than those obtained by both MOEA/D and MOEA/D+LS in most of the ZDT test problems. That means that the solutions obtained by MOEA/D+LS-II achieved a better approximation of the true $\mathcal{PF}$ than those solutions obtained by both MOEA/D+LS and MOEA/D. The exceptions were ZDT2 and ZDT4, where MOEA/D+LS and MOEA/D obtained better results than those achieved by MOEA/D+LS-II, respectively. Note however, that MOEA/D+LS was not significantly better than MOEA/D+LS-II for ZDT2.

In general, the performance of MOEA/D+LS-II and MOEA/D+LS was very similar for the ZDT test suite. The proposed MOMAs (i.e. MOEA/D+LS-II and MOEA/D+LS) outperformed the original MOEA/D in most of the ZDT test problems. However, for ZDT4, the I_H indicator showed that the local search did not improve the performance of MOEA/D, i.e., the proposed MOMAs did not outperform the original

MOEA/D. We attributed the poor performance of these hybrid algorithms to the high multi-frontality that ZDT4 has.

Two Set Coverage (I_C) Performance Measure. According to Table 9, MOEA/D+LS-II obtained better results (in terms of the I_C indicator) than those produced by MOEA/D+LS and MOEA/D in the majority of the ZDT test problems. This means that the solutions obtained by MOEA/D+LS-II dominated a higher portion of the solutions produced by MOEA/D+LS and MOEA/D, respectively. However, as we can see, MOEA/D was significantly better in ZDT4.

7.3.2 Results for the DTLZ test suite

Hypervolume (I_H) Performance Measure. According to Table 8, the proposed MOEA/D+LS-II obtained better results in terms of the I_H indicator than those obtained by both MOEA/D and MOEA/D+LS in most of the DTLZ test problems. Therefore, the solutions obtained by MOEA/D+LS-II achieved a better approximation of the $\mathcal{PF}$ than those solutions obtained by both MOEA/D+LS and MOEA/D. The exceptions were DTLZ1, DTLZ3 and DTLZ4, where MOEA/D+LS and MOEA/D obtained better results than those achieved by MOEA/D+LS-II, respectively. Note however, that for DTLZ4, MOEA/D+LS was not significantly better than MOEA/D+LS-II.

In general, the performance of MOEA/D+LS-II and MOEA/D+LS was very similar for the DTLZ test suite. The proposed MOMAs outperformed the original MOEA/D in most of the DTLZ test problems. Although, for DTLZ1 and DTLZ3, the I_H indicator showed that the local search mechanisms employed by both MOEA/D+LS and MOEA/D+LS-II did not improve the performance of the original MOEA/D. The poor performance of these MOMAs for DTLZ1 and DTLZ3 is attributed to the high multi-frontality that these problems have.

Two Set Coverage (I_C) Performance Measure. According to Table 9, MOEA/D+LS-II obtained a better I_C value than the one achieved by MOEA/D+LS and MOEA/D, in most of the DTLZ

MOP	MOEA/D+LS-II average (σ)	MOEA/D+LS average (σ)	MOEA/D average (σ)	reference vector $\mathbf{r}$
ZDT1	**0.842309** (0.009087)	0.819246 (0.038088)	0.751315 (0.033339)	$(1.1, 1.1)^T$
ZDT2	0.363225 (0.133365)	**0.384962** (0.151212)	0.210410 (0.080132)	$(1.1, 1.1)^T$
ZDT3	**1.055714** (0.230182)	0.995692 (0.158499)	0.990212 (0.089499)	$(1.1, 1.1)^T$
ZDT4	0.185765 (0.156602)	0.169257 (0.212639)	**0.600217** (0.138989)	$(1.1, 1.1)^T$
ZDT6	**0.462714** (0.022012)	0.462559 (0.050484)	0.425904 (0.010630)	$(1.1, 1.1)^T$
DTLZ1	0.317083 (0.001075)	0.316904 (0.001091)	**0.317249** (0.000957)	$(0.7, 0.7, 0.7)^T$
DTLZ2	**0.768727** (0.000594)	0.768621 (0.000466)	0.768696 (0.000644)	$(1.1, 1.1, 1.1)^T$
DTLZ3	0.128942 (0.219193)	0.221197 (0.282045)	**0.383622** (0.245603)	$(1.1, 1.1, 1.1)^T$
DTLZ4	0.768122 (0.000574)	**0.768966** (0.000664)	0.768935 (0.000645)	$(1.1, 1.1, 1.1)^T$
DTLZ5	**0.426492** (0.000114)	0.426307 (0.000167)	0.426115 (0.000675)	$(1.1, 1.1, 1.1)^T$
DTLZ6	**0.426416** (0.000254)	0.426345 (0.000714)	0.000228 (0.001226)	$(1.1, 1.1, 1.1)^T$
DTLZ7	**1.929710** (0.162598)	1.922224 (0.012057)	1.916040 (0.016969)	$(1.1, 1.1, 6.1)^T$
WFG1	**16.510348** (0.202859)	15.921475 (0.955856)	14.964720 (1.030077)	$(3, 4, 4)^T$
WFG2	8.882838 (0.822917)	8.973534 (0.857198)	**8.996212** (0.964342)	$(2, 2, 4)^T$
WFG3	**40.721010** (0.745928)	39.594021 (0.987888)	39.740488 (1.120458)	$(4, 3, 6)^T$
WFG4	68.763272 (0.993944)	**69.193123** (1.001366)	69.160679 (0.877216)	$(3, 5, 7)^T$
WFG5	65.825280 (0.525636)	**66.050850** (0.727933)	65.818947 (0.828483)	$(3, 5, 7)^T$
WFG6	**66.323221** (0.364806)	64.694658 (2.015885)	65.712844 (1.167871)	$(3, 5, 7)^T$
WFG7	**67.179656** (0.141568)	66.844937 (1.478663)	66.490864 (1.388620)	$(3, 5, 7)^T$
WFG8	**62.988349** (0.229227)	62.880565 (1.148814)	62.742809 (1.249541)	$(3, 5, 7)^T$
WFG9	**64.601092** (0.437234)	62.835454 (2.171200)	63.019018 (1.486697)	$(3, 5, 7)^T$

Table 8.: Comparison of results with respect to the I_H indicator for MOEA/D+LS-II, MOEA/D+LS and MOEA/D.

test problems. This means that the solutions obtained by MOEA/D+LS-II dominated to more of the solutions generated by the other MOEAs with respect to which it was compared. Although MOEA/D+LS obtained better results for DTLZ3 and

MOP	I_C(MOEA/D+LS-II, MOEA/D+LS)	I_C(MOEA/D+LS, MOEA/D+LS-II)	I_C(MOEA/D+LS-II, MOEA/D)	I_C(MOEA/D, MOEA/D+LS-II)
	average (σ)	average (σ)	average (σ)	average (σ)
ZDT1	**0.488783** (0.242740)	0.187745 (0.115837)	**0.857230** (0.112002)	0.020588 (0.027799)
ZDT2	**0.139349** (0.080375)	0.069892 (0.083325)	**0.324347** (0.154109)	0.092473 (0.073905)
ZDT3	**0.641866** (0.360988)	0.066667 (0.092721)	**0.910252** (0.120419)	0.011667 (0.030092)
ZDT4	**0.604067** (0.448448)	0.286458 (0.363906)	0.005000 (0.026926)	**0.903125** (0.096606)
ZDT6	**0.182720** (0.259879)	0.075877 (0.288685)	**0.703818** (0.167528)	0.142105 (0.124614)
DTLZ1	**0.128864** (0.134213)	0.090123 (0.060379)	0.088727 (0.118301)	**0.118519** (0.072457)
DTLZ2	**0.138093** (0.023027)	0.134011 (0.022464)	**0.136058** (0.028879)	0.133446 (0.026424)
DTLZ3	0.339699 (0.411232)	**0.574434** (0.394037)	0.130119 (0.317745)	**0.778317** (0.311620)
DTLZ4	0.131181 (0.027343)	**0.131203** (0.026156)	**0.199246** (0.028263)	0.156701 (0.030018)
DTLZ5	**0.059061** (0.029332)	0.051323 (0.030307)	**0.040094** (0.019201)	0.029101 (0.026901)
DTLZ6	**0.046298** (0.029818)	0.024510 (0.029493)	**1.000000** (0.000000)	0.000000 (0.000000)
DTLZ7	**0.176586** (0.032866)	0.095659 (0.017961)	**0.174799** (0.028298)	0.088217 (0.017506)
WFG1	0.063529 (0.136050)	**0.340676** (0.225264)	0.000264 (0.000990)	**0.553730** (0.100654)
WFG2	**0.595429** (0.284197)	0.035220 (0.083130)	**0.659266** (0.270807)	0.010482 (0.038371)
WFG3	**0.149590** (0.108504)	0.061418 (0.053089)	**0.136721** (0.115116)	0.077163 (0.050562)
WFG4	**0.298331** (0.161050)	0.245455 (0.150202)	**0.276878** (0.147371)	0.250399 (0.117450)
WFG5	**0.597973** (0.114974)	0.022591 (0.021742)	**0.659285** (0.127168)	0.018009 (0.020085)
WFG6	**0.450379** (0.229348)	0.153476 (0.155554)	**0.320391** (0.138133)	0.198039 (0.139328)
WFG7	0.261405 (0.108373)	**0.280901** (0.118356)	**0.308449** (0.107897)	0.234595 (0.096892)
WFG8	**0.297450** (0.142725)	0.229609 (0.112690)	**0.306217** (0.142979)	0.210615 (0.092903)
WFG9	**0.499652** (0.243981)	0.142812 (0.158706)	**0.485352** (0.213164)	0.126540 (0.110188)

Table 9.: Comparison of results with respect to the I_C indicator for MOEA/D+LS-II compared to MOEA/D+LS and MOEA/D

DTLZ5, it was not significantly better than MOEA/D+LS-II. On the other hand, MOEA/D, in fact, was better for the DTLZ1 and DTLZ3 test problems, which are multi-frontal.

7.3.3 Results for WFG test suite

Hypervolume (I_H) PERFORMANCE MEASURE. According to Table 8, the proposed MOEA/D+LS-II obtained better results in terms of the I_H indicator than those obtained by both MOEA/D and MOEA/D+LS in most of the WFG test problems. That means that the solutions obtained by MOEA/D+LS-II achieved a better approximation of the $\mathcal{PF}$ than those solutions obtained by both MOEA/D+LS and MOEA/D. The exceptions were WFG2, WFG4 and WFG5, where MOEA/D+LS and MOEA/D obtained better results than those achieved by MOEA/D+LS-II. Note however, that for WFG4 and WFG5, MOEA/D+LS was not significantly better than MOEA/D+LS-II. On the other hand, for WFG2, the I_H indicator showed that the local search mechanisms employed by both MOEA/D+LS and MOEA/D+LS-II did not improve the performance of the original MOEA/D. The multi-modality of WFG2 (presented in the last function of the MOP) has an influence on the performance of the MOMAs for this problem. It is worth noting, however, that MOEA/D was not significantly better than the proposed MOMAs for this specific problem.

In general, MOEA/D+LS-II showed its robustness outperforming both MOEA/D+LS and the original MOEA/D in most of the WFG test problems, which are considered more difficult to solve [63]. In some cases, such as WFG1, WFG3, WFG6, WFG7 and WFG9, the improved version of MOEA/D+LS, i.e., MOEA/D+LS-II, was significantly better than MOEA/D+LS.

Two Set Coverage (I_C) PERFORMANCE MEASURE. According to Table 9, MOEA/D+LS-II obtained a better I_C value than the one achieved by MOEA/D+LS and MOEA/D, in most of the WFG test problems. This means that the solutions obtained by MOEA/D+LS-II dominated to more of the solutions generated by the other MOEAs with respect to which it was compared. Although MOEA/D+LS and MOEA/D obtained better results for WFG1 and WFG7, it is worth noting that MOEA/D+LS-II reached better results regarding the I_H performance measure in those problems. I_H not only measures the convergence but also the maximum spread of solutions along the $\mathcal{PF}$, which is the

reason why MOEA/D+LS-II obtained better results regarding the I_H performance measure for these problems.

Finally, in order to appreciate the results obtained by each algorithm compared here, Appendix A.2, A.3 and A.4, show the plots of the final approximations to the $\mathcal{PF}$ obtained by the MOEA/D, the MOEA/D+LS and MOEA/D+LS-II.

7.4 Remarks

We have proposed an improved version of MOEA/D+LS. The proposed approach hybridizes the well-known MOEA/D with the NSS algorithm. The mathematical programming method works as a local search engine and it is employed to approximate solutions to the extremes and the maximum bulge of the $\mathcal{PF}$. Our preliminary results indicate that the proposed local search mechanism incorporated to MOEA/D, gives robustness and better performance when it is compared with respect to the original MOEA/D and MOEA/D+LS, over the set of 21 test problem adopted in this work. The proposed MOMA was found to be competitive with respect to the original MOEA/D and the MOEA/D+LS when performing 10,000 and 30,000 fitness function evaluations, for problems having two and three objectives, respectively. We consider that the strategy employed to hybridize Nelder and Mead's method with MOEA/D was appropriate for dealing with the MOPs adopted here. However, we also confirmed that multi-frontality is the main obstacle to accelerate converge to the $\mathcal{PF}$ in the proposed MOEA/D+LS-II. Because of its nature, the proposed local search mechanism could be easily coupled within other decomposition-based MOEAs, such as those reported in [98, 107, 149].

As indicated before, the use of a low number of fitness function evaluations in MOEAs is an important issue in multi-objective optimization, because there are several real-world applications that are computationally expensive to solve. In the last few years, the use of MOEAs assisted by surrogate models has been one of the most common techniques adopted to solve complex problems, see e. g. [36, 74, 105, 147, 156]. However, the prediction error of such models often directs the search towards regions in which no Pareto optimal solutions are found. This naturally motivates the idea of incorporating procedures to refine the solutions provided by surro-

gate models, such as adopting local search mechanisms. In the next Chapter, we focus on coupling the proposed local search mechanism into a MOEA assisted by surrogate models. We hypothesized that an appropriate combination of the explorative power of a MOEA assisted by surrogate models with the exploitative power of a local search engine, could improve the performance of a MOEA when performing a low number of fitness function evaluations.

8

Combining Surrogate Models and Local Search for Multi-objective Optimization

Multi-objective evolutionary algorithms have been successfully adopted to solve *Multi-objective Optimization Problems (MOPs)* in a wide variety of engineering and scientific problems [14]. However, in real-world applications is common to find objective functions which are very expensive to evaluate (in terms of computational time). This has considerably limited the use of evolutionary techniques to these types of problems. In recent years, several researchers have developed different strategies for reducing the computational time (measured in terms of the number of fitness function evaluations) that a *Multi-Objective Evolutionary Algorithm (MOEA)* requires to solve a determined problem. From such strategies, the use of surrogate models has been one of the most common techniques adopted to solve complex problems. In the specialized literature, several authors have reported the use of surrogate models to deal with MOPs, see e. g. [36, 64, 74, 105, 147, 156] among others. However, the high modality and dimensionality of some problems, often constitute major obstacles for surrogate models. Therefore, if a surrogate model is not able to shape the region in which the *Pareto optimal set (PS)* is contained, the search could be misinformed and converge to wrong regions. This has motivated the idea of incorporating procedures to refine the solutions provided by surrogate models, such as local search mechanisms. In general, the use of local search mechanisms based on mathematical programming methods combined with MOEAs assisted by surrogate models has been scarcely explored in the specialized literature.

In 2009, Georgopoulou and Giannakoglou [47] proposed a *Multi-Objective Memetic Algorithm (MOMA)* assisted by *Radial Basis Functions (RBFs)*. The local search mechanism uses a function which cor-

responds to an ascent method that incorporates gradient values provided by the surrogate model. Recently, Zapotecas and Coello [148] proposed a MOMA assisted by *Support Vector Regresion (SVR)* [136]. The local search mechanism is directed by several scalarization functions, which are solved using the Hooke and Jeeves algorithm [60]. The local search mechanism is assisted by SVR and the improved solutions are incorporated into the current population of the MOEA by using Pareto ranking.

The two above approaches assist their local search procedures with surrogate models. Therefore, even though these approaches use refinement mechanisms, the prediction error may misguide the local search. In Chapters 6 and 7, we showed the effectiveness of the Nelder and Mead method [102] when it was used as a local search engine into a MOEA. In this Chapter, we introduce a MOEA assisted by RBF networks adopting as refinement mechanism, a modified version of the local search procedure presented in Chapter 7. We hypothesized that an appropriate combination between exploration and exploitation could improve the performance of a MOEA when performing a low number of fitness function evaluations. In the next section, we present the foundations of RBFs which are important for understanding the proposed approach.

8.1 Radial Basis Function Networks

Radial Basis Function (RBF) networks are a feed-forward kind of neural network, which are commonly represented with three layers: an input layer with n nodes, a hidden layer with h nonlinear RBFs (or neurons), and an output node φ. The function value in a RBF depends on the distance from each point $\mathbf{x}$ to the origin, i.e. $g(\mathbf{x}) = g(\|\mathbf{x}\|)$. This function value can be generalized to distances from some other point $\mathbf{c}_j$, commonly called *center* of the basis function, that is:

$$g(\mathbf{x}, \mathbf{c}_j) = g(\|\mathbf{x} - \mathbf{c}_j\|)$$

The output $\varphi : \mathbb{R}^n \mapsto \mathbb{R}$ of the network is defined as:

$$\varphi(\mathbf{x}) = \sum_{j=1}^{h} w_j g(\|\mathbf{x} - \mathbf{c}_j\|), \quad z = 1, \ldots, k \tag{8.1}$$

where h is the number of neurons in the hidden layer, $\mathbf{c}_j$ is the center vector for the j^{th} neuron, and w_j's are the weights of the linear output neuron. In its basic form, all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance and the basis function g or kernel is taken to be Gaussian, although other basis functions are also possible (see for example those shown in Table 10).

Kernel	Description
Cubic	$g(r) = r^3$
Thin Plate Spline	$g(r) = r^2 \ln(r)$
Gaussian	$g(r, \sigma) = \exp(-r^2/2\sigma^2)$
Multi-quadratic	$g(r, \sigma) = \sqrt{r^2 + \sigma^2}$
Inverse multi-quadratic	$g(r, \sigma) = \sqrt{r^2 + \sigma^2}$

Table 10.: Kernels for a RBF neural network, where $r = \|\mathbf{x} - \mathbf{c}_i\|$

RBF networks can be used to interpolate a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ when the values of that function are known on a finite number of points: $f(\mathbf{x}_i) = y_i, i = 1 \ldots, N$. Taking into account the h centers $\mathbf{c}_j$'s ($j = 1, \ldots, h$) and evaluating the values of the basis functions at the points $\mathbf{x}_i$, i.e., $\phi_{ij} = g(\|\mathbf{c}_j - \mathbf{x}_i\|, \sigma_j)$ the weights can be solved from the equation:

$$\begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \vdots & & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nm} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{8.2}$$

Therefore, the weights w_i's can be solved by simple linear algebra, using the least squares method, that is:

$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} \tag{8.3}$$

The parameter σ_j of the kernels (Gaussian, multi-quadratic and inverse multi-quadratic) determines the amplitude of each basis function and it can be adjusted to improve the model accuracy.

Until now, we have presented the theoretical foundations of RBF networks. In the next section, we shall present, the detailed description of the proposed MOEA assisted by RBF networks.

8.2 A MOEA based on Decomposition Assisted by RBF Networks

8.2.1 General Framework

The proposed *Multi-Objective Evolutionary Algorithm based on Decomposition assisted by Radial Basis Functions (MOEA/D-RBF)* decomposes the MOP (3.1) into N single-objective optimization problems. MOEA/D-RBF uses a well-distributed set of N weight vectors $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ to define a set of single-objective optimization subproblems. Here, we employ the *Penalty Boundary Intersection (PBI)* approach to transform a MOP into a single-objective optimization problem, which consists in minimizing:

See Section 3.2.2 for a more detailed description of the PBI approach

$$\text{Minimize:} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^\star) = d_1 + \theta d_2 \tag{8.4}$$

such that:

$$d_1 = \frac{||(F(\mathbf{x}) - \mathbf{z}^\star)^T \mathbf{w}||}{||\mathbf{w}||}$$
$$\text{and} \quad d_2 = \left|\left|(F(\mathbf{x}) - \mathbf{z}^\star) - d_1 \frac{\mathbf{w}}{||\mathbf{w}||}\right|\right|$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^n$, θ is the penalty value and $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ is the utopian vector, i.e., $z_i^\star = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \Omega\}, \forall i = 1, \ldots, k$.

Each subproblem is solved by the *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*, which is assisted by a surrogate model based on RBF networks. For a better understanding of this approach, Algorithm 18 shows the general framework of the proposed MOEA/D-RBF. In the following sections, we describe in detail the components of the MOEA/D-RBF which are outlined in Algorithm 18.

8.2.2 Initialization

Initially, a training set $T_{set} = \{\mathbf{x}_i, \ldots, \mathbf{x}_{N_t}\}$ of N_t well-spread solutions is generated. For this task, we employed the Latin hypercube sampling method [92]. The set of solutions T_{set} is evaluated by using the real fitness function. The number of current fitness function evaluations

Algorithm 18: General framework of MOEA/D-RBF

Input:
$W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$: a well-distributed set of weight vectors.
N_t: the number of points in the initial training set.
E_{max}: the maximum number of evaluations allowed in MOEA/D-RBF.
Output:
A: an approximation to the *Pareto front* ($\mathcal{PF}$).

1 **begin**
2 **Initialization:** Generate a set $T_{set} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}\}$ of N_t points such that $\mathbf{x}_i \in \Omega$ $(i = 1, \ldots N_t)$, by using an experimental design method. Evaluate the **F**-functions values of these points. Set A as the set of nondominated solutions found in T_{set}. Set $n_{eval} = N_t$. Generate a population $\hat{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N individuals such that $\mathbf{x}_i \in \Omega$ $(i = 1, \ldots N)$, by using an experimental design method. $stopping\_criterion = FALSE$. For details of this step see **Section 8.2.2**.
3 **while** $(stopping\_criterion == FALSE)$ **do**
4 **Model Building:** Using the **F**-function values of the points in T_{set}, build the predictive surrogate model by using different RBF networks. Calculate the weights for each RBF network according to its training error in T_{set}. For details of this step see **Section 8.2.3**.
5 **Evaluate** $\hat{P}$**:** Evaluate the population $\hat{P}$ using the surrogate model.
6 **Find an approximation to** $\mathcal{PF}$**:** By using MOEA/D, the surrogate model and the population $\hat{P}$, obtain $\hat{P}^\star = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{N_t}\}$, where $\hat{P}^\star$ is an approximation to $\mathcal{PF}$, see **Section 8.2.4**.
7 **Select points for updating** T_{set}**:** By using the selection scheme, select a set of solutions from $\hat{P}^\star$ to be evaluated and included in the training set T_{set}. Update A using the selected solutions. For each evaluated solution, set $n_{eval} = n_{eval} + 1$. **If** $n_{eval} < E_{max}$ **then** $stopping\_criterion = TRUE$. For a detailed description of this step see **Section 8.2.5**.
8 **Update population** $\hat{P}$**:** Update the population $\hat{P}$ according to the updating scheme, see **Section 8.2.6**.
9 **end**
10 **return** A;
11 **end**

n_{eval} is initially set as $n_{eval} = N_t$. MOEA/D-RBF uses an external archive A to store the nondominated solutions found so far in the evolutionary process. This archive is initialized with the nondominated solutions found in T_{set}. At the beginning, a population $\hat{P} = \{\mathbf{x}_i, \ldots, \mathbf{x}_N\}$ of N solutions is generated by employing the Latin hypercube sampling method. The stopping criterion considered in MOEA/D-RBF is

the number of fitness function evaluations and, therefore, the stopping criterion is initially set as false, i.e. $\texttt{stopping\_criterion} = \texttt{FALSE}$.

8.2.3 Building the Model

As previously indicated, we use a surrogate model based on RBF networks. In order to improve the prediction of the surrogate model, the Gaussian, the multi-quadratic and the inverse multi-quadratic kernels are used in a cooperative way for obtaining the approximated value of a solution. In the following sections, we describe the necessary components for building the surrogate model.

8.2.3.1 *Hidden Nodes*

The hidden nodes in an RBF network play an important role in the performance of the RBF network. In general, there is no method available for estimating the number of hidden nodes in an RBF network. However, it has been suggested in [56, 57, 86, 127] that Kolmogorov's theorem [80] concerning the realization of arbitrary multivariate functions, provides theoretical support for neural networks that implement such functions.

Theorem 2 (Kolmogorov [80])
A continuous real-valued function defined as $f : [0,1]^n \mapsto \mathbb{R}$, $n \geqslant 2$, can be represented in the form:

$$f(x_1,\ldots,x_n) = \sum_{j=1}^{2n+1} g_j \left(\sum_{i=1}^{n} \phi_{ij}(x_i) \right) \tag{8.5}$$

where the g_j's are properly chosen continuous functions of one variable, and the ϕ_{ij}'s are continuous monotonically increasing functions independent of f.

The basic idea in Kolmogorov's theorem is captured in the network architecture of Figure 8.1, where a universal transformation M maps $\mathbb{R}^n$ into several uni-dimensional transformations. The theorem states that one can express a continuous multivariate function on a compact set in terms of sums and compositions of a finite number of single variable functions.

Motivated by this idea, the surrogate model built here, uses $2n+1$ hidden nodes (where n is the number of decision variables of the

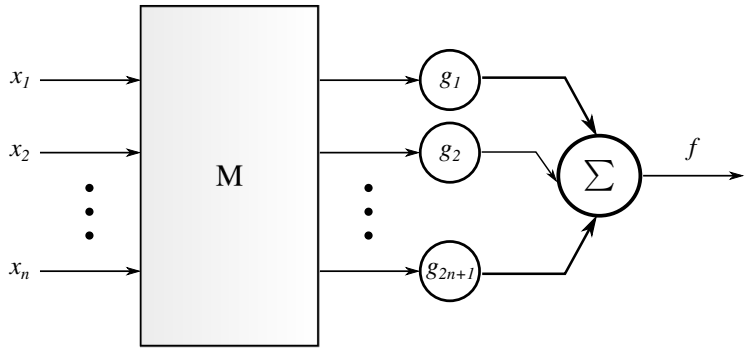


Figure 8.1.: Network representation of Kolmogorov's theorem

MOP). Considering T_{set} as the training set of N_t solutions used by the surrogate model, the centers of the $2n+1$ basis functions are defined by using the well-known k-means algorithm [90] on the training set T_{set} (with $k = 2n+1$). This criterion establishes that the cardinality of T_{set} should be greater than $2n+1$, i.e., $2n+1 < N_t$.

8.2.3.2 *Building the surrogate model*

The high modality and dimensionality of some functions, often produce problems to surrogate models. When the surrogate model is not able to properly shape the region of the search space in which the $\mathcal{PS}$ is located, then the search may be biased towards inappropriate regions. In order to improve the function prediction, MOEA/D-RBF uses different kernels for building different RBF networks. Each RBF network provides different shape of the search space and all of them provide information to predict the value of an arbitrary solution. Here, three different kernels are adopted: Gaussian, multi-quadratic and inverse multi-quadratic; these kernels are chosen because they possess the parameter σ which can be adjusted to improve the model accuracy, see Table 10. Note however that other types of kernels can also be adopted, although the use of more kernels could significantly increase the training time. In the following description, we consider the case with one single output node, i.e. with a single function. Note however, that this model can be generalized for more than one function.

Let $T_{set} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}\}$ be the set of N_t solutions evaluated with the real fitness function. Let h be the number of hidden nodes (or basis functions) considered in the RBF network. Let $\mathbf{c}_j$ and σ_j ($j = 1, \ldots, m$) be the center and the amplitude of each basis function, respectively.

The training of the RBF network for a determined kernel K consists in finding the weight vector $\mathbf{w} = (w_1, \ldots, w_m)^T$ such that it solves equation (8.3). Each parameter σ_j of each basis function is initially defined by the standard deviation of the solutions contained in each cluster obtained by the k-means algorithm (with mean $\mathbf{c}_j$).

Once the weight vector $\mathbf{w}$ is obtained, the model accuracy is improved by adjusting the vector of parameters $\sigma = (\sigma_1, \ldots, \sigma_m)^T$. Since the value of the adopted kernel depends of σ_j, from equation (8.2), the training error on the training set T_{set}, can be written as:

$$\psi(\sigma) = \|\Phi\mathbf{w} - \mathbf{y}\| \tag{8.6}$$

where $\mathbf{y} = (y_1, \ldots, y_{Nt})^T$ is the vector of the real function values for each solution $\mathbf{x}_i \in T_{set}$, i.e., $y_i = f(\mathbf{x}_i)$. Φ is the matrix which contains the evaluations of each point $\mathbf{x}_i \in T_{set}$ for each basis function, i.e., $\phi_{ij} = g(\|\mathbf{c}_j - \mathbf{x}_i\|, \sigma_j)$, for $i = 1, \ldots, N_t$ and $j = 1, \ldots, h$.

The parameters σ_j are then adjusted by using the *Differential Evolution (DE)* algorithm [129], whose objective is to minimize the training error defined in equation (8.6). Once the σ_j parameters are adjusted, the prediction function for a determined kernel K of a solution $\mathbf{x} \in \Omega$ can be calculated by:

$$\hat{\varphi}_K(\mathbf{x}) = \sum_{j=1}^{h} w_j \cdot g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j) \tag{8.7}$$

8.2.3.3 *Cooperative surrogate models and Function Prediction*

Once the three RBF networks are built, each of them using the three above mentioned kernels, the prediction of the function is carried out. Let $\varphi_{GK}(\mathbf{x})$, $\varphi_{MK}(\mathbf{x})$ and $\varphi_{IMK}(\mathbf{x})$ be the predicted value given by RBF networks using the Gaussian, multi-quadratic and inverse multi-quadratic kernel, respectively. These three RBF networks cooperate by providing information of the search space that they model. Therefore, the function prediction $\hat{f}$ for an arbitrary $\mathbf{x} \in \Omega$ is defined by:

$$\hat{f}(\mathbf{x}) = \lambda_1 \cdot \varphi_{GK}(\mathbf{x}) + \lambda_2 \cdot \varphi_{MK}(\mathbf{x}) + \lambda_3 \cdot \varphi_{IMK}(\mathbf{x}) \tag{8.8}$$

where $\Lambda = (\lambda_1, \lambda_2, \lambda_3)^T$ is a weight vector, i.e. $\lambda_i \geqslant 0$ and $\sum_{i=1}^{3} \lambda_i = 1$. Therefore, the weight for each predicted value needs to be calculated.

Let T_{set} be the knowledge set for training the different RBF networks. The weight vector Λ is then calculated by:

$$\lambda_i = \frac{\alpha_i}{|T_{set}|}, i = 1, 2, 3 \tag{8.9}$$

where α_i is the number of solutions in T_{set} with the lowest prediction error for the i^{th} RBF network (Gaussian, multi-quadratic and inverse multi-quadratic, respectively).

8.2.4 Finding an Approximation to $\mathcal{PF}$

MOEA/D-RBF approximates solutions to the $\mathcal{PF}$ by using the well-known MOEA/D [155] (see Algorithm 8). The search is conducted by the set of weight vectors $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$. MOEA/D searches the solutions to each scalar problem defined by each weight vector $\mathbf{w}_i \in W$. The evolutionary process of MOEA/D is performed during a determined number of generations by employing the prediction function defined in equation (8.8). The final population denoted as $\hat{P}^\star$ is then reported as an approximation to $\mathcal{PF}$.

8.2.5 Selecting Points to Evaluate

Let $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ be the well-distributed set of weight vectors used by MOEA/D. Let $\hat{P}^\star$ be the approximation to $\mathcal{PF}$ obtained by MOEA/D. Let $W_s = \{\mathbf{w}_1^s, \ldots, \mathbf{w}_{N_s}^s\}$ be a well-distributed set of weight vectors, such that $|W_s| < |W|$. For each $\mathbf{w}_i^s \in W_s$, we define $B_s(\mathbf{w}_i^s) = \{\mathbf{w}_1, \ldots, \mathbf{w}_{N_a}\}$, such that $\mathbf{w}_1, \ldots, \mathbf{w}_{N_a} \in W$ are the $N_a = \lfloor \frac{N}{N_s} \rfloor$ closest weight vectors from W to $\mathbf{w}_i^s$. With that, an association of weight vectors from W to W_s is defined. This association defines a set of neighborhoods $B_s(\mathbf{w}_i^s)$ which are distributed along the whole set of weight vectors W, see Figure 8.2. Once the neighborhoods $B_s(\mathbf{w}_i^s)$ have been defined, a set of solutions is selected to be included in the training set T_{set}, according to the next description.

8.2.5.1 *Selecting Points to be Evaluated using the Real Fitness Function*

A set $S = \{\mathbf{x}_1, \ldots, \mathbf{w}_{N_s}\}$ of N_s solutions taken from $\hat{P}$ is chosen to be evaluated using the real fitness function. Each solution in S is

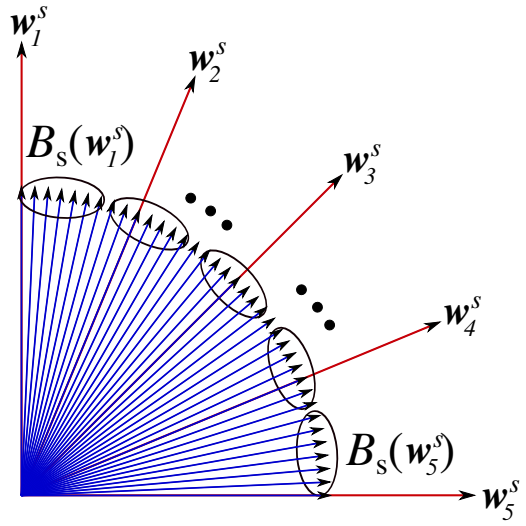


Figure 8.2.: Association of weight vectors from W to W_s. The vectors in blue represent the projection of W set, while the vectors in red represent the projection of W_s set. This association defines the neighborhoods $B_s(\mathbf{w}_1^s)$ to $B_s(\mathbf{w}_5^s)$

selected such that it minimizes the problem defined by a weight vector $\mathbf{w}_j \in B_s(\mathbf{w}_i^s)$, where $i = 1, \ldots, N_s$ and $j = 1, \ldots N_a$.

At each call of the selection procedure, the weight vector $\mathbf{w}_j$ is selected by sweeping the set of weight vectors in $B_s(\mathbf{w}_i^s)$ in a cyclic way, i.e., once the last weight vector is selected, the next one is picked up from the beginning. Since the neighborhoods $B_s(\mathbf{w}_i^s)$ are distributed along the whole weight set W, the selection of solutions in each neighborhood should obtain spread solutions along the $\mathcal{PF}$. No solution in S should be duplicated. If this is the case, the repeated solution should be removed from S. For each new evaluated solution, we set $n_{eval} = n_{eval} + 1$, if $n_{eval} \geqslant E_{max}$ then we set $\texttt{stopping\_criterion} = \texttt{TRUE}$, where n_{eval} and E_{max} are the current and the maximum number of fitness function evaluations, respectively.

8.2.5.2 *Updating the Training Set and the External Archive*

The maximum number of solutions in the training set T_{set} is defined by the parameter N_t. The updating of T_{set} is carried out by defining a well-distributed set of N_t weight vectors $W_t = \{\mathbf{w}_1^t, \ldots, \mathbf{w}_{N_t}^t\}$. Therefore, the best N_t different solutions from $T = \{T_{set} \cup S\}$, such that they

minimize the subproblems defined by each weight vector $\mathbf{w}_i^t \in W_t$, are used to update T_{set}. If after updating the training set, any solution $\mathbf{s}_j \in S$ was not selected to be included in T_{set}, then, it is added by replacing the closest solution (in the objective space) in T_s. With this, all solutions in S are included in T_{set} and the model can be improved even if it has been previously misinformed.

The external archive A contains the nodominated solutions found along the search. For each $\mathbf{s}_j \in S$, the external archive is updated by removing from A all the solutions dominated by $\mathbf{s}_j$, and then, $\mathbf{s}_j$ is stored in A if no solutions in A dominate $\mathbf{s}_i$.

8.2.6 Updating the Population

Once the external archive is updated, the population $\hat{P}$ is also updated for the next iteration of MOEA/D. Considering the external archive A as the set of nondominated solutions found by MOEA/D-RBF, the population $\hat{P}$ of N solutions is updated according to the following description.

Let $\mathbf{m}$ and σ be the average and standard deviation of the solutions contained in A. Then, new bounds in the search space are defined according to:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma \end{aligned}$$

where $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively.

Once the new bounds have been defined, a well-distributed set Q of $N - |A|$ solutions is generated by means of the Latin hypercube sampling method [92] in the new search space. The population $\hat{P}$ is then redefined by the union of Q and A, that is $\hat{P} = \{Q \cup A\}$.

The effectiveness of the proposed MOEA/D-RBF has been shown in [151]. In the next section, we present a hybridization between the proposed MOEA/D-RBF and the Nelder and Mead method (also known as *Nonlinear Simplex Search (NSS)*), which is the main aim in this chapter.

8.3 The MOEA/D-RBF with Local Search

The proposed *MOEA/D-RBF with Local Search (MOEA/D-RBF+LS)* decomposes the MOP (3.1) into N single-objective optimization problems. MOEA/D-RBF uses a well-distributed set of N weight vectors $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ to define a set of single-objective optimization subproblems by using the PBI approach. Each subproblem is solved by MOEA/D, which is assisted by RBF networks. After each iteration of MOEA/D, the local search procedure is applied. Algorithm 19 shows the general framework of the proposed MOEA/D-RBF+LS. In the following sections, we describe in detail the proposed local search mechanism adopted by MOEA/D-RBF+LS.

See Section 3.2.2 for a more detailed description of the PBI approach

8.3.1 Local Search Mechanism

As indicated before, the local search mechanism adopted by MOEA/D-RBF+LS is based on the Nelder and Mead algorithm [102]. The effectiveness of NSS as a local search mechanism in MOEAs has been shown by several authors, see e. g. [78, 79, 146, 145, 150, 152, 157]. Here, we adopt the local search mechanism employed by the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II)* which was presented in Chapter 7. However, in order to be coupled with the proposed MOEA/D-RBF, some modification have been introduced. The local search procedure is performed after each iteration of MOEA/D-RBF (see **Step 7** of Algorithm 19). With that, the solutions in the training set are refined and the function prediction for the next iteration of MOEA/D-RBF could be improved. In the following, we present in detail the components of the local search engine outlined in Algorithms 19 and 20.

8.3.1.1 *Defining the Population* P_{ls}

At the beginning, a new population P_{ls} of N_{ls} individuals is defined in order to direct the local search. Let W_{ls} and T_{set} be a well-distributed set of weight vectors and the training set, respectively. P_{ls} is stated by choosing different solutions $\mathbf{x}^t \in T_{set}$ such that they minimize:

$$g(\mathbf{x}^t | \mathbf{w_i}, \mathbf{z}^\star), \text{for each } \mathbf{w}_i^{ls} \in W_{ls}$$

Algorithm 19: General framework of MOEA/D-RBF+LS

Input:
$W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$: a well-distributed set of weight vectors.
N_t: the number of points in the initial training set.
E_{max}: the maximum number of evaluations allowed in MOEA/D-RBF+LS.

Output:
A: an approximation to the $\mathcal{PF}$.

begin
  Step 1. INITIALIZATION: Generate a set $T_{set} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}\}$ of N_t points such that $\mathbf{x}_i \in \Omega$ $(i = 1, \ldots N_t)$, by using an experimental design method. Evaluate the **F**-functions values of these points. Set A as the set of nondominated solutions found in T_{set}. Set $n_{eval} = N_t$. Generate a population $\hat{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N individuals such that $\mathbf{x}_i \in \Omega$ $(i = 1, \ldots N)$, by using an experimental design method. stopping_criterion = FALSE.
  while (stopping_criterion == FALSE) **do**
    Step 2. MODEL BUILDING: See section 8.2.3.
    Step 3. EVALUATE $\hat{P}$: Evaluate the population $\hat{P}$ using the surrogate model.
    Step 4. FIND AN APPROXIMATION TO $\mathcal{PF}$: By using MOEA/D, the surrogate model and the population $\hat{P}$, obtain the approximation $\hat{P}$ to $\mathcal{PF}$. See section 8.2.4.
    Step 5. SELECT POINTS FOR UPDATING T_{set}: See section 8.2.5.
    Step 6. UPDATE POPULATION $\hat{P}$: See section 8.2.6.
    Step 7. LOCAL SEARCH: Apply nonlinear simplex search by using the training set T_{set}, see section 8.3.1 and Algorithm 20.
  end
  return A;
end

The cardinality of W_{set} should be much less that the cardinality of the weight set W (which directs the search of MOEA/D-RBF+LS), i.e., $|W_{ls}| << |W|$. With that, a small portion of search directions are considered by the local search engine, in order to obtain well-spread solutions along the $\mathcal{PF}$.

8.3.1.2 *Defining the Search Direction and the Initial Solution*

The proposed local search mechanism, approximates solutions to the maximun bulge (sometimes called knee) of the $\mathcal{PF}$. Therefore, the local search is focused on minimizing the subproblem that

Algorithm 20: Use of Local Search

Input:
a stopping criterion;
T_{set}: the training set used by MOEA/D-RBF+LS.
$W_{ls} = \{\mathbf{w}_i, \ldots, \mathbf{w}_N\}$: a well-distributed set of weight vectors for the local search.
S_t: the similarity threshold for the local search;
E_{ls}: the maximum number of evaluations for the local search.

Output:
T_{set}: the updated training set T_{set}.

begin
- **Step 1.** DEFINING THE POPULATION P_{ls}: Using the weight set W_{ls} and the training set T_{set}, define the population P_{ls} from which the local search is performed, see Section 8.3.1.1;
- **Step 2** DEFINING THE SEARCH DIRECTION AND THE INITIAL SOLUTION: Define the search direction and the initial solutions from which the local search starts, according to Section 8.3.1.2
- **Step 3.** CHECKING SIMILARITY: Obtain the similarity (S_{ls}) between $\mathbf{p}_{ini}$ and the previous initial solution ($\mathbf{p}'_{ini}$) for the local search, see Section 8.3.1.3;
- **if** *there are enough resources* **and** $S_t < S_{ls}$ **then**
  - **Step 4.** BUILDING THE SIMPLEX: Build the initial simplex for the nonlinear simplex search, see Section 8.3.1.4;
  - **Step 5.** DEFORMING THE SIMPLEX: Perform any movement (reflection, contraction or expansion) for obtaining $\mathbf{p}_{new}$ according to Nelder and Mead's method, see Section 8.3.1.5;
  - **Step 6.** UPDATING THE POPULATION AND THE EXTERNAL ARCHIVE: Update the population P_{ls} and the external archive A using the new solution $\mathbf{p}_{new}$ according to the rules presented in Section 8.3.1.6.
  - **Step 7.** STOPPING CRITERION: If the stopping criterion is satisfied then stop the local search and go to **Step 7**. Otherwise, go to **Step 4**, see Section 8.3.1.7.
- **end**
- **Step 8.** UPDATING THE TRAINING SET: Update the training set T_{set} according to the updating scheme, see Section 8.3.1.8.

end

approximates the solutions lying on the knee of the $\mathcal{PF}$. Thus, the search direction is defined by the weighting vector:

$$\mathbf{w}_s = (1/k, \ldots, 1/k)^T$$

where k is the number of objective functions. Considering the use of the PBI approach, the penalty value θ is set to $\theta = 10$.

Let A be the set of nondominated solutions found during the search of MOEA/D-RBF+LS. Let $\mathbf{w}_s$ be the weighting vector that defines the search direction for the nonlinear simplex search. The solution $\mathbf{p}_{ini}$ which starts the search is defined by:

$$\mathbf{p}_{ini} = \mathbf{x} \in A, \text{ such that minimizes: } g(\mathbf{x}|\mathbf{w}_s, \mathbf{z}^\star)$$

Solution $\mathbf{p}_{ini}$ represents not only the initial search point, but also the simplex head from which the simplex will be built.

8.3.1.3 *Checking Similarity*

The NSS explores the neighborhood of the solution $\mathbf{p}_{ini} \in A$. Since the simplex search is applied after each iteration of the MOEA/D, most of the time, the initial solution $\mathbf{p}_{ini}$ does not change its position from one generation to another. For this reason, the proposed local search mechanism stores a record ($\mathbf{p}'_{ini}$) of the last position from which the nonlinear simplex search starts. At the beginning of the execution of MOEA/D-RBF+LS, the initial position record is set as empty, that is: $\mathbf{p}'_{ini} = \emptyset$. Once the simplex search is performed, the initial solution is stored in the historical record, i.e., $\mathbf{p}'_{ini} = \mathbf{p}_{ini}$. In this way, for the next call of the local search, a previous comparison of similarity is performed. That is, the local search will be applied, if and only if, $\|\mathbf{p}_{ini} - \mathbf{p}'_{ini}\| > S_{ls}$, where S_{ls} represents the similarity threshold. Both the updating of the historical record and the similarity operator are performed for each initial solution $\mathbf{p}_{ini}$ which minimizes the subproblem defined by $\mathbf{w}_s$. This is the same strategy used by MOEA/D+LS-II in Chapter 7. However, here, we adopted a similarity threshold $S_t = 0.01$.

8.3.1.4 *Building the Simplex*

Let A be the set of nondominated solutions found during the search of MOEA/D-RBF+LS. Then, the simplex Δ is built in three different ways, depending of the cardinality of A.

i. $|A| = 1$: Set $\sigma = (0.01, \ldots, 0.01)^T$ and the simplex is defined as:

$$\Delta = \{\mathbf{a}, \Delta_2, \ldots, \Delta_{n+1}\}$$

where $\mathbf{a} \in A \subset \Omega$ and the remaining n vertices $\Delta_i \in \Omega$ ($i = 1, \ldots, n$) are generated by using a low-discrepancy sequence.

In our study, we adopted the Hammersley sequence [55] to generate a well-distributed sampling of solutions in a determined search space. The search space is defined by:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{a} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{a} + \sigma \end{aligned}$$

In this way, the vertices are generated by means of the Hammersley sequence using as bounds $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$.

ii. $1 < |A| < (n+1)$: The simplex is defined by using all solutions in A and the remaining $l = (n+1) - |A|$ solutions are generated by using the Hammersley sequence. However, the bounds are defined as:

$$\begin{aligned} \mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma \end{aligned}$$

where $(\mathbf{m})$ and (σ) are the average and the standard deviation of the solutions contained in A, respectively. $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the lower and upper bounds of the new search space, respectively.

iii. $|A| \geqslant (n+1)$: In this case, the simplex is built by choosing in a random way, $n+1$ solutions taken from A.

8.3.1.5 *Deforming the Simplex*

Let $\mathbf{w}_s$ be the weight vector that defines the search direction for the NSS. Let Δ be the simplex defined by the above description. The simplex search will be focused on minimizing the subproblem defined by the weighting vector $\mathbf{w}_s$. At each iteration of the simplex search, the $n+1$ vertices of the simplex Δ are sorted according to their value for the subproblem that it tries to minimize (the best value is the first element). In this way, a movement into the simplex is performed for generating the new solution $\mathbf{p}_{new}$. The movements are calculated according to the equations provided by Nelder and Mead, see Section 5.1. Each movement is controlled by three scalar parameters: **reflection** (ρ), **expansion** (χ) and **contraction** (γ).

The NSS was conceived for unbounded problems. When dealing with bounded variables, the created solutions can be located outside the allowable bounds after some movements of the NSS. In order to

deal with this, we bias the new solution if any component of $\mathbf{p}_{new}$ lies outside the bounds according to:

$$\mathbf{p}_{new}^{(j)} = \begin{cases} \mathbf{L}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} < \mathbf{L}_{bound}^{(j)} \\ \mathbf{U}_{bound}^{(j)} & \text{, if } \mathbf{p}_{new}^{(j)} > \mathbf{U}_{bound}^{(j)} \\ \mathbf{p}_{new}^{(j)} & \text{, otherwise.} \end{cases} \tag{8.10}$$

where $\mathbf{L}_{bound}^{(j)}$ and $\mathbf{U}_{bound}^{(j)}$ are the lower and upper bounds of the j^{th} parameter of $\mathbf{p}_{new}$, respectively.

8.3.1.6 *Updating the Population and the External Archive*

The information provided by the local search mechanism is introduced into the population P_{ls}. Since we are dealing with MOPs, the new solution generated by any movement of the simplex search could be better than more than one solution in the current population. Thus, we adopt the following mechanism in which more than one solution from the population could be replaced.

Let $\mathbf{p}_{new}$ be the solution generated by any movement of the NSS. Let $B(\mathbf{w}_s)$ and W_{ls} be the neighborhood of the T closest weighting vectors to $\mathbf{w}_s$, and the well-distributed set of all weighting vectors, respectively. We define:

$$Q = \begin{cases} B(\mathbf{w}_s) & \text{, if } r < \delta \\ W & \text{otherwise} \end{cases}$$

where r is a random number having a uniform distribution. In this work, we use $\delta = 0.5$.

The population P_{ls} is updated by replacing at most R_{ls} solutions from P_{ls} such that, $g(\mathbf{p}_{new}|\mathbf{w}_i, z) < g(\mathbf{x}_i|\mathbf{w}_i, z)$, where $\mathbf{w}_i \in Q$ and $\mathbf{x}_i \in P_{ls}$, such that $\mathbf{x}_i$ minimizes the subproblem defined by $\mathbf{w}_i$. In our study, we set $R_{ls} = 15$ as the maximum number of solutions to replace.

Function $g(\mathbf{x}|\mathbf{w}_i, z)$ represents the scalarization function defined by the PBI approach, see Section 3.2.2.

The external archive A contains the nodominated solutions found during the search of MOEA/D-RBF+LS. For each new solution $\mathbf{p}_{new}$, the external archive is updated by removing from A all the solutions dominated by $\mathbf{p}_{new}$, and then, $\mathbf{p}_{new}$ is stored in A if no solutions in A dominate $\mathbf{p}_{new}$.

8.3.1.7 *Stopping Criterion*

The local search procedure is limited to a maximum number of fitness function evaluations defined by E_{ls}. In this way, the proposed local search has the following stopping criteria:

1. If the nonlinear simplex search overcomes the maximum number of evaluations (E_{ls}) or there are not enough resources for continuing the search of MOEA/D-RBF+LS, the local search is stopped.
2. The search could be inefficient if the simplex has been deformed so that it has collapsed into a region in which there are no local minima. According to Lagarias et al. [82] the simplex search finds a better solution in at most $n+1$ iterations (at least in convex functions with low dimensionality). Therefore, if the simplex search does not find a better value for the subproblem defined by $\mathbf{w}_s$ in $n+1$ iterations, we stop the search. Otherwise, we perform another movement into the simplex by going to **Step 3** of Algorithm 20.

8.3.1.8 *Updating the Training Set*

The knowledge obtained by the local search is introduced to MOEA/D-RBF+LS by updating the training set T_{set}. The maximum number of solutions in the training set T_{set} is defined by the parameter N_t. The updating of T_{set} is carried out by defining a well-distributed set of N_t weight vectors $W_t = \{\mathbf{w}_1^t, \ldots, \mathbf{w}_{N_t}^t\}$. Therefore, the best N_t different solutions from $T = \{T_{set} \cup A\}$, such that they minimize the subproblems defined by each weight vector $\mathbf{w}_i^t \in W_t$, are used to update T_{set}. With this, the nondominated solutions found by the local search are included in T_{set} and the model can be improved even if it has been previously misinformed.

8.4 Experimental Results

8.4.1 Test Problems

The effectiveness of MOEA/D-RBF has been shown in [151], where it was compared with respect to two state-of-the-art MOEAs: the *Multi-Objective Evolutionary Algorithm based on Decomposition with Gaussian*

Process Model (MOEA/D-EGO) [156] and the original MOEA/D [155]. Here, we limit the comparative study of MOEA/D-RBF+LS to be performed with respect to MOEA/D-RBF and the original MOEA/D.

Therefore, in order to assess the performance of the proposed MOEA/D-RBF+LS, we compare its results with respect to those obtained by MOEA/D-RBF and the original MOEA/D. We adopted five test problems whose $\mathcal{PF}$s have different characteristics including convexity, concavity, disconnections and multi-modality. Thus, the *Zitzler-Deb-Thiele (ZDT)* test suite [158] (except for ZDT5, which is a binary problem) is adopted (see Appendix A.2). We used 30 decision variables for problems from ZDT1 to ZDT3, while ZDT4 and ZDT6 were tested using 10 decision variables, as suggested in [158]. Besides, we adopt a real-world problem related to the airfoil design problem (for a more detailed description of the case study tackled here, see Appendix B).

8.4.2 Performance Assessment

To assess the performance of the proposed MOEA/D-RBF+LS and MOEA/D-RBF on the test problems adopted, the *Hypervolume* (I_H) indicator was employed [160]. This performance measure is Pareto compliant [162], and quantifies both approximation and maximum spread of nondominated solutions along the $\mathcal{PF}$. A high I_H value, indicates that the approximation P is close to $\mathcal{PF}$ and has a good spread towards the extreme portions of the $\mathcal{PF}$. The interested reader is referred to section 3.4 for a more detailed description of this performance measure.

8.4.3 Experimental Setup

As indicated before, the proposed approach is compared with respect to MOEA/D-RBF and the original MOEA/D. For each MOP, 30 independent runs were performed with each algorithm. Each algorithm was restricted to 1,000 fitness function evaluations. For the airfoil design problem, the search was restricted to 5,000 fitness function evaluations.

The parameters used for MOEA/D, which is employed by MOEA/D-RBF and MOEA/D-RBF+LS, were set as in [155]. This is because there is empirical evidence that indicates that these are the

most appropriate parameters for solving the ZDT test suite, see [155]. The weight vectors for the algorithms were generated as in [155], i.e., the setting of N and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ is controlled by a parameter H. More precisely, $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are all the weight vectors in which each individual weight takes a value from:

$$\left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}$$

Therefore, the number of such vectors in W is given by $N = C^{k-1}_{H+k-1}$, where k is the number of objective functions (for the test problems adopted, $k = 2$). For MOEA/D-RBF and MOEA/D-RBF+LS, the set W was defined with $H = 299$, i.e., 300 weight vectors. The set W_t was generated with $H = 10n - 1$. Therefore, $N_t = 10n$ weight vectors (which define the cardinality of the training set), where n is the number of decision variables of the MOP. The set W_s uses $H = 9$, i.e., $N_s = 10$ weight vectors. Note that these values of the parameters are the ones used by MOEA/D-RBF in [151].

For the local search, the set W_{ls} was generated using $H = 99$; therefore, $N_{ls} = 100$. The NSS was performed using $\rho = 1, \chi = 2$ and $\gamma = 1/2$, for the reflection, expansion and contraction, respectively. The maximum number of solutions to be replaced was set to $R_{ls} = 15$ and the maximum number of fitness function evaluations was set to $E_{ls} = 2(n + 1)$. Finally, the similarity threshold was set to $S_t = 0.01$. The execution of the algorithms was carried out on a computer with a 2.66GHz processor and 4GB in RAM.

As indicated before, the algorithms were evaluated using the I_H performance measure. The results obtained are summarized in Table 11. This table displays both the *average* and the standard deviation (σ) of the I_H indicator for each MOP, respectively. The reference vector $\mathbf{r}$ used for computing I_H, for each MOP, is shown in Table 11. For an easier interpretation, the best results are presented in **boldface** for each test problem adopted.

8.5 Numerical Results

8.5.1 ZDT Test Problems

Table 11 shows the results obtained for the I_H indicator when the algorithms were tested on the ZDT test problems. From this table it is possible to see that the two MOEAs assisted by surrogate models (i. e., MOEA/D-RBF and MOEA/D-RBF+LS), obtained better results than those achieved by the original MOEA/D in most of the ZDT problems. The exception was ZDT4, where MOEA/D obtained a better I_H value. The poor performance of the MOEAs assisted by surrogate models is attributed to the high multi-frontality that ZDT4 has, which evidently confuses the surrogate approaches. This table shows also that MOEA/D-RBF+LS obtained a better approximation to $\mathcal{PF}$ than the one achieved by MOEA/D-RBF in most of the ZDT test problems. The exception was ZDT1 where MOEA/D-RBF was better than MOEA/D-RBF+LS. However, MOEA/D-RBF was not significantly better than MOEA/D-RBF+LS. The performance of MOEA/D-RBF+LS and MOEA/D-RBF was very similar for ZDT1 and ZDT2. The differences were more significant for ZDT3, ZDT4 and ZDT6. These last problems have special features that deteriorate the good performance of surrogate models. ZDT3 is a problem whose $\mathcal{PF}$ consists of several noncontiguous convex parts. ZDT4 is multi-modal problem, which causes difficulties to model the search space in a suitable way. ZDT6 has two difficulties caused by the nonuniformity of the search space: first, the Pareto optimal solutions are nonuniformly distributed along the $\mathcal{PF}$; second, the density of the solutions is lower near the $\mathcal{PF}$ and gets higher as we move away from the $\mathcal{PF}$. These features evidently present a major obstacle to the surrogate model employed by MOEA/D-RBF. However, the use of local search for these problems, improved the performance of MOEA/D-RBF. In fact, MOEA/D-RBF+LS obtained better approximations to the $\mathcal{PF}$ for these MOPs, and in some cases, such as in ZDT4 and ZDT6, it was significantly better.

8.5.2 Airfoil Design Problem

For this particular problem, the features of the $\mathcal{PF}$ are unknown. According to the results presented in Table 11, we can see that MOEA/D-RBF+LS obtained better I_H values than those reached by MOEA/D-RBF. This means that our proposed MOEA/D-RBF+LS obtained a better approximation and spread of solutions along the $\mathcal{PF}$ than MOEA/D-RBF.

The original MOEA/D employed, on average, 5,050 seconds to achieve the convergence with 5,000 fitness function evaluations. MOEA/D-RBF and MOEA/D-RBF+LS employed, on average, between 1,900 and 2,000 seconds to achieve a value in the I_H indicator similar to the one reported by MOEA/D in Table 11.Therefore, we argue that our proposed MOEA/D-RBF+LS is a good choice for dealing with computationally expensive MOPs.

MOP	MOEA/D-RBF+LS average (σ)	MOEA/D-RBF average (σ)	MOEA/D average (σ)	Reference vector **r**
ZDT1	0.868197 (0.002837)	**0.870908** (0.000371)	0.000000 (0.000000)	$(1.1, 1.1)^T$
ZDT2	**0.536389** (0.004921)	0.536265 (0.000593)	0.000000 (0.000000)	$(1.1, 1.1)^T$
ZDT3	**0.876380** (0.102611)	0.837894 (0.179280)	0.009974 (0.029880)	$(1.1, 1.1)^T$
ZDT4	12.441923 (30.715277)	5.739229 (30.906695)	**76.593009** (108.237963)	$(30.0, 30.0)^T$
ZDT6	**96.299610** (0.761493)	95.313012 (1.271933)	44.984399 (8.783998)	$(10, 10)^T$
MOPRW	**2.6818676e-07** (6.417924e-09)	2.493786e-07 (6.483342e-09)	2.149916e-07 (2.446593e-08)	$(0.007610, 0.005236)^T$

Table 11.: Results of the I_H metric for MOEA/D-RBF+LS, MOEA/D-RBF and MOEA/D.

8.6 Remarks

The effectiveness of MOEA/D-RBF was tested in [151], where it was compared with respect to the original MOEA/D and a current state-of-the-art MOEA assisted by surrogate models (the MOEA/D-EGO [156]). Here, we have introduced an extension of MOEA/D-RBF which includes a local search mechanism in order to

improve the convergence to the $\mathcal{PF}$, when a low number of fitness function evaluations is used. The local search mechanism adopted by MOEA/D-RBF+LS, is the result of previous studies presented in Chapter 7. However, in order to be coupled with MOEA/D-RBF, some modification were introduced. The resulting MOEA/D-RBF+LS was able to improve the convergence of MOEA/D-RBF, when the search was limited to a low number of fitness function evaluations. We also validated our proposed approach with a real-world computationally expensive MOP: an airfoil design problem. The obtained results have shown that MOEA/D-RBF+LS is a viable choice to deal with MOPs having different features, and the applicability to real-world applications could speed up convergence to the $\mathcal{PF}$ in comparison to conventional MOEAs.

9

Conclusions and Future Work

In this thesis, we have presented the contributions developed so far for improving *Multi-Objective Evolutionary Algorithms (MOEAs)* by using direct search methods. Such contributions follow two main directions:

1. *to reduce the number of fitness function evaluations*, and
2. *to maintain a good representation of the Pareto front (PF)*.

In our study, we adopted a popular direct search method reported in the specialized literature, the Nelder and Mead algorithm, also known as *Nonlinear Simplex Search (NSS)*. Throughout this thesis, we showed the capabilities of NSS when is used for dealing with *Multi-objective Optimization Problems (MOPs)*. Furthermore, we introduced distinct strategies to hybridize the NSS with different MOEAs. In the following, we present the final remarks observed in this study.

9.1 Conclusions

Multi-objective Nonlinear Simplex Search (MONSS). Among the direct search methods, NSS is one of the most popular methods for solving optimization problems reported in the specialized literature. Unlike modern optimization methods, NSS can converge to a non-stationary point unless the problem satisfies stronger conditions than are necessary for modern methods [109, 93]. However, in order to improve the performance of the NSS, several modifications have been introduced since the late 1970s, see e. g. [143, 142, 133, 132]. Because of its nature, which is based on movements over a set of solutions (called simplex), NSS has became a viable option to be hybridized with population-based search strategies, such as the *Evolutionary Algorithm (EA)*. In the last decade, many researchers have reported hybrid approaches

that combine NSS with EA for solving single-objective optimization problems. This has motivated the idea of using the NSS in a multi-objective context.

In Chapter 5, we introduced an extension of the NSS for multi-objective optimization. The proposed MONSS decomposes a MOP into several single-objective optimization problems. Therefore, MONSS uses the directions given by a weighted vector (which defines a scalar optimization problem) to approximate solutions to the *Pareto optimal set (PS)* by modifying a simplex shape according to the NSS method. Different from standard implementations of NSS, the proposed strategies implemented in our study, did not perform the *shrinkage* step. With that, a considerable number of fitness function evaluations was saved. Experimental results showed that the proposed strategy was effective when dealing with MOPs having low and moderate dimensionality in decision variable space. The effectiveness of MONSS was tested with several test problems taken from the specialized literature, and its performance was compared with respect to a state-of-the-art MOEA. Considering that the NSS was conceived to deal with single-objective optimization, and based on the results presented in Chapter 5, we conclude that the proposed MONSS satisfies our original research goal.

Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS). In Chapter 6, we presented a hybridization of the *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)* with the NSS algorithm, in which the former acts as the global search engine, and the latter works as a local search engine. The local search mechanism used by MOEA/D+LS is based on the MONSS framework, which adopts a decomposition approach similar to the one used in MOEA/D. Therefore, its use could be easily coupled within other decomposition-based MOEAs, such as those reported in [98, 107, 149]. Our proposed MOEA/D+LS, was found to be competitive with respect to the original MOEA/D over the *Zitzler-Deb-Thiele (ZDT)* and *Deb-Thiele-Laumanns-Zitzler (DTLZ)* test suites, which were adopted in our comparative study. We consider that the strategy employed to hybridize the MONSS framework with MOEA/D was, in general, appropriate

for dealing with the MOPs adopted here. However, in order to improve this *Multi-Objective Memetic Algorithm (MOMA)*, some issues could be better addressed.

Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II). In Chapter 8, we introduced an enhanced version of MOEA/D+LS. Similar to MOEA/D+LS, MOEA/D+LS-II decomposes a MOP into single-objective optimization problems. Therefore, in order to achieve a good representation of the $\mathcal{PF}$, the search is directed by a well-distributed set of weight vector. MOEA/D+LS-II incorporates the NSS algorithm as a local search engine into the well-known MOEA/D, as in MOEA/D+LS. However, in order to improve the local search, some modifications were introduced. The proposed MOEA/D+LS-II directs the search towards the extremes and the maximum bulge (sometimes called knee) of the $\mathcal{PF}$, in contrast to MOEA/D+LS which directs the search towards different neighborhoods of the whole set of weight vectors. Besides, the selection mechanism (from which the local search starts) used in MOEA/D+LS was also modified. MOEA/D+LS-II incorporates a new mechanism base on similarity of solutions to decide when the local search should be applied. The performance of MOEA/D+LS-II was tested over the ZDT, DTLZ and *Walking-Fish-Group (WFG)* test suites, which consist on several problems with different features in their $\mathcal{PF}$. Our preliminary results indicate that the proposed mechanisms incorporated to MOEA/D, give robustness and better performance when it is compared with respect to the original MOEA/D and MOEA/D+LS. We also confirmed that multi-frontality is the main obstacle to accelerate convergence to the $\mathcal{PF}$ in our proposed MOMAs, i. e., the MOEA/D+LS and MOEA/D+LS-II.

Multi-Objective Evolutionary Algorithm based on Decomposition assisted by Radial Basis Functions (MOEA/D-RBF). The use of a low number of fitness function evaluations in MOEAs is an important issue in multi-objective optimization, because there are several real-world applications that are computationally expensive to solve. In the last few years, the use of MOEAs assisted by surrogate models has been one of the most common techniques adopted to solve complex problems, see e. g. [36, 74, 105, 147,

156]. However, the prediction error of such models often directs the search towards regions in which no Pareto optimal solutions are found. In Chapter 8, we introduced an algorithm based on the well-known MOEA/D which is assisted by *Radial Basis Function (RBF)* networks. The resulting MOEA/D-RBF uses different kernels in order to have different shapes of the fitness landscape. With that, each RBF network provides information which is used to improve the value of the objective function. According to the results presented in [151], our proposed MOEA/D-RBF was able to outperform both to the original MOEA/D and to the *Multi-Objective Evolutionary Algorithm based on Decomposition with Gaussian Process Model (MOEA/D-EGO)* [156] when performing a low number of fitness function evaluations. The good performance of the proposed MOEA/D-RBF was tested not only with respect to standard test problems, but also with a real-world application (an airfoil design problem). Although the design of MOEAs assisted by surrogate models is not the main aim in this thesis, we have also contributed to the state of the art regarding these evolutionary techniques.

MOEA/D-RBF with Local Search (MOEA/D-RBF+LS). The high modality and dimensionality of some problems, often constitute major obstacles for surrogate models. If a surrogate model is not able to shape the region in which the $\mathcal{PS}$ is contained, the search could be misinformed and converge to wrong regions. This has motivated the idea of incorporating procedures to refine the solutions provided by surrogate models, such as local search mechanisms. In general, the use of local search mechanisms based on mathematical programming methods combined with MOEAs assisted by surrogate models has been scarcely explored in the specialized literature. As a final contribution of this thesis, in Chapter 8, we introduced a MOEA assisted by RBF networks which are adopted as its refinement mechanism, a modified version of the local search procedure presented in Chapter 7. The proposed MOEA/D-RBF+LS was able to improve the convergence of MOEA/D-RBF, when the search was limited to a low number of fitness function evaluations. We also validated our proposed approach with a real-world computationally expensive MOP: an airfoil design problem presented in Appendix B.

The obtained results have shown that MOEA/D-RBF+LS is a viable choice to deal with MOPs having different features, and the applicability to real-world applications could speed up convergence to the $\mathcal{PF}$ in comparison to conventional MOEA. However, here we confirmed that the multi-modality of MOPs confuses both the surrogate model and the local search procedure that we designed.

In general, the performance of the MOMAs presented in this thesis, showed improvements with respect to the baseline MOEA adopted. Each hybrid approach was widely tested over several MOPs taken from the specialized literature. It is important to note that the conclusions provided in this chapter, are restricted to the set of functions adopted, since by the No Free Lunch Theorem [140], general conclusions about the behavior of our proposed algorithms can not be possibly drawn. However, given the robustness shown by the proposed MOMAs in the large number of MOPs adopted, we expect a good performance when applied to solve other MOPs with similar features to the one adopted here. Note however, that multi-modal problems are the Achilles' heel of the hybrid approaches presented herein.

9.2 Future Work

As part of our future work, we are interested in designing other mechanism that helps us decide whether the local search engine will be triggered or not. The exploration of different strategies for constructing the simplex continues to be a good path for future research. We hypothesized that the use of an appropriate simplex and a good hybridization strategy could be a powerful combination for solving complex and computationally expensive MOPs (as for example those presented in [154]). One of the most difficult task to address, is to extend our proposed hybrid approaches to deal with constrained MOPs using for example, the Complex method [112] or any variants of the NSS algorithm. The modifications done to the NSS algorithm reported in the specialized literature, have shown better performance than the original NSS algorithm. This motivates the idea to adopt these modified approaches in order to be hybridized with MOEAs. Self-adaption techniques applied to MOEAs is a current area of re-

search, see e. g. [153, 1, 134]. These mechanisms provide to MOEAs, suitable parameters to speed up converge in the evolutionary process. Inspired on this techniques, it is possible to explore different mechanisms for adjusting, in dynamic way, the control parameters during the search of the NSS algorithm. On the other hand, investigating different stopping criteria for the local search mechanism and hybridizing the local search engines proposed in this thesis with other decomposition-based MOEAs, is also a task to address. Finally, in order to better support the validity of results, a more elaborated study regarding statistical analysis using a witness test such as either Wilcoxon test, T-test or an *Analysis of variance (ANOVA)*, is task left for future work.

In this thesis, we adopted the NSS algorithm to improve the performance of MOEAs. However, in the specialized literature there are other many direct search methods that could achieve better results than the one reported in this work. Therefore, the use of other direct search methods such as: the Hooke and Jeeves algorithm [60], the conjugate directions of Powell [108] or the Zangwill method [144], among many others, remains as an open research area.

A

Test Functions Description

A.1 Classic Multi-objective Optimization Problems

DEB2. The test function DEB2 was proposed by Deb in [24]. This test function has a disconnected and non-convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ f_2(\mathbf{x}) &= g(\mathbf{x}) \cdot h(\mathbf{x}) \end{aligned} \tag{DEB2}$$

such that:

$$\begin{aligned} g(\mathbf{x}) &= 1 + 10x_2 \\ h(\mathbf{x}) &= 1 - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)^2 - \frac{f_1(\mathbf{x})}{g(\mathbf{x})} \times \sin\left(12\pi f_1(\mathbf{x})\right) \end{aligned}$$

where $x_i \in [0, 1]$.

FON2. The test function FON2 was introduced by Fonseca and Fleming in [44]. This problem has a connected and non-convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ f_2(\mathbf{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{aligned} \tag{FON2}$$

where $n = 3$ and $x_i \in [-4, 4]$.

LAU. The test function LAU was proposed by Laummans [84]. This problem has a connected and convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1^2 + x_2^2 \\ f_2(\mathbf{x}) &= (x_1 + 2)^2 - x_2^2 \end{aligned} \tag{LAU}$$

where $x_i \in [-50, 50]$.

LIS. The test function LIS was introduced by Lis and Eiben in [87]. This problem has a connected and non-convex $\mathcal{PS}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= \sqrt[8]{x_1^2 + x_2^2} \\ f_2(\mathbf{x}) &= \sqrt[4]{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} \end{aligned} \tag{LIS}$$

where $x_i \in [-5, 10]$.

MUR. The test function MUR was proposed by Murata and Ishibuchi in [101]. This test function has a connected and convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= 2\sqrt{x_1} \\ f_2(\mathbf{x}) &= x_1(1 + x_2) + 5 \end{aligned} \tag{MUR}$$

where $x_1 \in [1, 4]$ and $x_2 \in [1, 2]$.

REN1. The test function REN1 was introduced by Valenzuela and Uresti in [135]. This problem has a connected and non-convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= \frac{1}{x_1^2 + x_2^2 + 1} \\ f_2(\mathbf{x}) &= x_1^2 + 3x_2^2 + 1 \end{aligned} \tag{REN1}$$

where $x_i \in [-3, 3]$.

REN2. The test function REN2 was introduced by Valenzuela and Uresti in [135]. This problem has a connected and non-convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 + x_2 + 1 \\ f_2(\mathbf{x}) &= x_1^2 + 2x_2^2 - 1 \end{aligned} \tag{REN2}$$

where $x_i \in [-3, 3]$.

VNT2. The test function VNT2 was proposed by Viennet et al. in [137]. This test function has a connected and non-convex $\mathcal{PF}$. The problem consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3 \\ f_2(\mathbf{x}) &= \frac{(x_1+x_2-3)^2}{36} + \frac{(-x_1+x_2+2)^2}{8} - 17 \\ f_3(\mathbf{x}) &= \frac{(x_1+2x_2-1)^2}{175} + \frac{(2x_2-x_1)^2}{17} - 13 \end{aligned} \quad \text{(VNT2)}$$

where $x_i \in [-4, 4]$.

VNT3. The test function VNT3 was proposed by Viennet et al. in [137]. This test function has a connected and non-convex $\mathcal{PF}$. The problem consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= 0.5(x_1^2+x_2^2) + \sin(x_1^2+x_2^2) \\ f_2(\mathbf{x}) &= \frac{(3x_1-2x_2+4)^2}{8} + \frac{(x_1-x_2+1)^2}{27} + 15 \\ f_3(\mathbf{x}) &= \frac{1}{(x_1^2+x_2^2+1)} - 1.1\exp(-x_1^2-x_2^2) \end{aligned} \quad \text{(VNT3)}$$

where $x_i \in [-3, 3]$.

A.2 Zitzler-Deb-Thiele Test Problems

ZDT1. The test function ZDT1 has a convex $\mathcal{PF}$ and consists in minimizing:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x) \cdot h(f_1(x), g(x)) \end{aligned} \tag{ZDT1}$$

such as:

$$\begin{aligned} g(x) &= 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\ h(f_1(x), g(x)) &= 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{aligned}$$

where $n = 30$, and $x_i \in [0, 1]$. The $\mathcal{PF}$ is formed with $g(x) = 1$, i.e., $x_j = 0$ for all $j = 2, \ldots, n$.

ZDT2. The test function ZDT2 is the non-convex counterpart to (ZDT1) and consists in minimizing:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x) \cdot h(f_1(x), g(x)) \end{aligned} \tag{ZDT2}$$

such as:

$$\begin{aligned} g(x) &= 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\ h(f_1(x), g(x)) &= 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{aligned}$$

where $n = 30$, and $x_i \in [0, 1]$. The $\mathcal{PF}$ is formed with $g(x) = 1$, i.e., $x_j = 0$ for all $j = 2, \ldots, n$.

ZDT3. The test function ZDT3 represents the discreteness feature; its $\mathcal{PF}$ consists of several noncontiguous convex part and it consists in minimizing:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(\mathbf{x}) \cdot h(f_1(\mathbf{x}), g(\mathbf{x})) \end{aligned} \tag{ZDT3}$$

such as:

$$\begin{aligned} g(x) &= 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\ h(f_1(x), g(x)) &= 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right) \sin(10\pi f_1(x)) \end{aligned}$$

where $n = 30$, and $x_i \in [0, 1]$. The t$\mathcal{PF}$ is formed with $g(x) = 1$, i.e., $x_j = 0$ for all $j = 2, \ldots, n$. The introduction of the sine function in h causes discontinuity in the $\mathcal{PF}$. However, there is no discontinuity in the parameter space (i.e. in the $\mathcal{PS}$).

ZDT4. This test function contains 21^9 local Pareto fronts and, therefore, it tests for the EA's ability to deal with multimodality; it consists in minimizing:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(\mathbf{x}) \cdot h(f_1(\mathbf{x}), g(\mathbf{x})) \end{aligned} \tag{ZDT4}$$

such as:

$$\begin{aligned} g(x) &= 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \\ h(f_1(x), g(x)) &= 1 - \sqrt{\tfrac{f_1(x)}{g(x)}} \end{aligned}$$

where $n = 10$, $x_1 \in [0, 1]$, and $x_2, \ldots, x_n \in [-5, 5]$. The $\mathcal{PF}$ is formed with $g(x) = 1$, i.e., $x_j = 0$ for all $j = 2, \ldots, n$, the best local Pareto front with $g(x) = 1.25$. Note that not all local Pareto sets are distinguishable in the objective space.

ZDT6. This test function includes two difficulties caused by the nonuniformity of the search space: first, the Pareto optimal solutions are nonuniformly distributed along the $\mathcal{PF}$; second, the density of the solutions is lower near the $\mathcal{PF}$ and gets higher as we move away from the front. This problem consists in minimizing:

$$\begin{aligned} f_1(x) &= 1 - \exp(-4x_1) \cdot \sin^6(6\pi x_1) \\ f_2(x) &= g(x) \cdot h(f_1(x), g(x)) \end{aligned} \tag{ZDT6}$$

such as:

$$\begin{aligned} g(x) &= 1 + 9\left(\tfrac{1}{(n-1)} \sum_{i=2}^{n} x_i\right)^{0.25} \\ h(f_1(x), g(x)) &= 1 - \left(\tfrac{f_1(x)}{g(x)}\right)^2 \end{aligned}$$

where $n = 10$, and $x_i \in [0, 1]$. The $\mathcal{PF}$ is formed with $g(x) = 1$, i.e., $x_j = 0$ for all $j = 2, \ldots, n$, and has a non-convex shape.

A.3 Deb-Thiele-Laummans-Zitzler Test Problems

DTLZ1. As a simple test problem, DTLZ1 constitutes a m-objective problem with a linear $\mathcal{PF}$. This problem consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= \tfrac{1}{2}(1+g(\mathbf{x}_m))x_1x_2\cdots x_{m-1} \\ f_2(\mathbf{x}) &= \tfrac{1}{2}(1+g(\mathbf{x}_m))(1-x_{m-1})\cdots x_1x_2 \\ &\vdots \\ f_{m-1}(\mathbf{x}) &= \tfrac{1}{2}(1+g(\mathbf{x}_m))(1-x_2)x_1 \\ f_m(\mathbf{x}) &= \tfrac{1}{2}(1+g(\mathbf{x}_m))(1-x_1) \end{aligned} \qquad \text{(DTLZ1)}$$

such as:

$$g(\mathbf{x}_m) = 100\left[|\mathbf{x}_m| + \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right)\right]$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0,1]$ for all $i = 1,\ldots,n$ (where n is the number of decision variables).

The Pareto optimal solutions correspond to $\mathbf{x}_m = \mathbf{0.5}$ and the objective function values lie on the linear hyperplane: $\sum_{i=1}^{m} f_i^\star = 0.5$. A value of $k = 5$ is suggested. In the above problem, the total number of variables is $n = m + k - 1$. The difficulty in this problem is to converge to the hyper-plane. The search space contains $(11^k - 1)$ local Pareto fronts, each of which can attract a MOEA.

The problem can be made more difficult by using other difficult multimodal g functions (using a larger k) and/or replacing x_i by nonlinear mapping $x_i = N_i(y_i)$ and treating y_i as decision variables. For a scale-up study, we suggest testing a MOEA with different values of m, perhaps in the range $m \in [2,10]$. It is interesting to note that for $m > 3$ cases all the Pareto optimal solutions on a three-dimensional plot involving f_m and any two other objectives will lie on or below the above hyper-plane.

DTLZ2. This test problem has a spherical $\mathcal{PF}$, and consists in minimizing:

$$\begin{aligned}
f_1(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\cos\left(x_{m-2}\tfrac{\pi}{2}\right)\cos\left(x_{m-1}\tfrac{\pi}{2}\right)\\
f_2(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\cos\left(x_{m-2}\tfrac{\pi}{2}\right)\sin\left(x_{m-1}\tfrac{\pi}{2}\right)\\
f_3(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\sin\left(x_{m-2}\tfrac{\pi}{2}\right)\\
&\vdots\\
f_{m-1}(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\sin\left(x_2\tfrac{\pi}{2}\right)\\
f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))\sin\left(x_1\tfrac{\pi}{2}\right)
\end{aligned} \tag{DTLZ2}$$

such that:

$$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0,1]$ for all $i = 1,\ldots,n$ (where n is the number of decision variables).

The Pareto optimal solutions correspond to $\mathbf{x}_m = \mathbf{0.5}$, and all objective function values must satisfy the equation $\sum_{i=1}^{m} f_i^2 = 1$. It is recommended to use $k = |\mathbf{x}_m| = 10$. The total number of variables is $n = m + k - 1$.

This function can also be used to investigate a MOEA's ability to scale up its performance with a large number of objectives. Like in (DTLZ1), for $m > 3$, the Pareto optimal solutions must lie inside the first quadrant of the unit sphere in a three-objective plot with f_m as one of the axes. To make the problem more difficult, each variable x_i (for $i = 1,\ldots,m-1$) can be replaced by the mean value of p variables: $x_i = \frac{1}{p}\sum_{k=(i-1)p+1}^{p} x_k$.

DTLZ3. This test problem is defined in the same way as (DTLZ2), except for a new g function. This introduces many local Pareto fronts, to which a MOEA can get attracted. This problem consists in minimizing:

$$\begin{aligned}
f_1(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\cos\left(x_{m-2}\tfrac{\pi}{2}\right)\cos\left(x_{m-1}\tfrac{\pi}{2}\right)\\
f_2(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\cos\left(x_{m-2}\tfrac{\pi}{2}\right)\sin\left(x_{m-1}\tfrac{\pi}{2}\right)\\
f_3(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\cdots\sin\left(x_{m-2}\tfrac{\pi}{2}\right)\\
&\vdots\\
f_{m-1}(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos\left(x_1\tfrac{\pi}{2}\right)\sin\left(x_2\tfrac{\pi}{2}\right)\\
f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))\sin\left(x_1\tfrac{\pi}{2}\right)
\end{aligned}$$

(DTLZ3)

such that:

$$g(\mathbf{x}_m) = 100\left[|\mathbf{x}_m| + \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2 - \cos\left(20\pi(x_i - 0.5)\right)\right]$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0, 1]$ for all $i = 1, \ldots, n$ (where n is the number of decision variables).

It is suggested that $k = |\mathbf{x}_m| = 10$. There are a total of $n = M + k - 1$ decision variables in this problem. The above g function introduces $(3^k - 1)$ local Pareto fronts, and only one $\mathcal{PF}$. All local Pareto fronts are parallel to the $\mathcal{PF}$ and a MOEA can get stuck at any of these local Pareto fronts, before converging to the $\mathcal{PF}$ (at $g^\star = 0$). The $\mathcal{PF}$ corresponds to $\mathbf{x}_m = \mathbf{0.5}$. The next local Pareto front is at $g^\star = 1$.

DTLZ4. This problem is a variation of (DTLZ2) with a modified meta-variable mapping $x_i \mapsto x_i^\alpha$ ($\alpha > 0$), and it consists in minimizing:

$$\begin{aligned}
f_1(\mathbf{x}) &= (1 + g(\mathbf{x}_m)) \cos\left(x_1^\alpha \tfrac{\pi}{2}\right) \cdots \cos\left(x_{m-2}^\alpha \tfrac{\pi}{2}\right) \cos\left(x_{m-1}^\alpha \tfrac{\pi}{2}\right) \\
f_2(\mathbf{x}) &= (1 + g(\mathbf{x}_m)) \cos\left(x_1^\alpha \tfrac{\pi}{2}\right) \cdots \cos\left(x_{m-2}^\alpha \tfrac{\pi}{2}\right) \sin\left(x_{m-1}^\alpha \tfrac{\pi}{2}\right) \\
f_3(\mathbf{x}) &= (1 + g(\mathbf{x}_m)) \cos\left(x_1^\alpha \tfrac{\pi}{2}\right) \cdots \sin\left(x_{m-2}^\alpha \tfrac{\pi}{2}\right) \\
&\vdots \\
f_{m-1}(\mathbf{x}) &= (1 + g(\mathbf{x}_m)) \cos\left(x_1^\alpha \tfrac{\pi}{2}\right) \sin\left(x_2^\alpha \tfrac{\pi}{2}\right) \\
f_m(\mathbf{x}) &= (1 + g(\mathbf{x}_m)) \sin\left(x_1^\alpha \tfrac{\pi}{2}\right)
\end{aligned}$$

(DTLZ4)

such as:

$$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0, 1]$ for all $i = 1, \ldots, n$ (where n is the number of decision variables).

This problem investigates a MOEA's ability to maintain a good distribution of solutions as they tend to find only the extremes of the $\mathcal{PF}$. It is suggested the use of both $\alpha = 100$ and $k = 10$. There are a total of $n = m + k - 1$ decision variables in this problem.

The above modification allows a dense set of solutions to exist near the $f_m - f_1$ plane. It is interesting to note that although the search space has a variable density of solutions, the classical weighted-sum approaches or other directional methods may not have any added difficulty in solving these problems compared to (DTLZ2). Since MOEAs attempt to find multiple and well-distributed optimal Pareto solutions in one simulation run, these problems may hinder MOEAs to achieve a well-distributed set of solutions.

DTLZ5. This test problem uses a mapping of the parameter θ_i employed in the sine and cosine functions of (DTLZ2). Then, the test problem DTLZ5 is defined as minimizing:

$$\begin{aligned}
f_1(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\cos(\theta_{m-1}) \\
f_2(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\sin(\theta_{m-1}) \\
f_3(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\sin(\theta_{m-2}) \\
&\vdots \\
f_{m-1}(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\sin(\theta_2) \\
f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))\sin(\theta_1)
\end{aligned} \tag{DTLZ5}$$

such that:

$$\begin{aligned}
\theta_1 &= x_1\frac{\pi}{2} \\
\theta_i &= \frac{\pi}{4(1+g(\mathbf{x}_m))}(1+2g(\mathbf{x}_m)x_i), \text{ for } i=2,3,\ldots,m-1 \\
g(\mathbf{x}_m) &= \sum_{x_i\in\mathbf{x}_m}(x_i-0.5)^2
\end{aligned}$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0,1]$ for all $i=1,\ldots,n$ (where n is the number of decision variables).

The g function with $k = |x_m| = 10$ variables is suggested. As before, there are $n = m+k-1$ decision variables in this problem. The $\mathcal{PF}$ corresponds to $\mathbf{x}_m = \mathbf{0.5}$. This problem tests a MOEA's ability to converge to a degenerated curve and also allows an easier way to visually show (just by plotting f_m with any other objective function) the performance of a MOEA. Since there is a natural bias for solutions close to this Pareto curve, this problem may be easy for an algorithm to solve.

DTLZ6. The above test problem can be made harder by doing a similar modification to the g function in (DTLZ5) as done in (DTLZ3). This test problem consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\cos(\theta_{m-1}) \\ f_2(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\cos(\theta_{m-2})\sin(\theta_{m-1}) \\ f_3(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\cdots\sin(\theta_{m-2}) \\ &\vdots \\ f_{m-1}(\mathbf{x}) &= (1+g(\mathbf{x}_m))\cos(\theta_1)\sin(\theta_2) \\ f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))\sin(\theta_1) \end{aligned} \tag{DTLZ6}$$

such that:

$$\begin{aligned} \theta_1 &= x_1\frac{\pi}{2} \\ \theta_i &= \frac{\pi}{4(1+g(\mathbf{x}_m))}(1+2g(\mathbf{x}_m)x_i), \text{ for } i=2,3,\ldots,m-1 \\ g(\mathbf{x}_m) &= \sum_{x_i\in\mathbf{x}_m} x_i^{0.1} \end{aligned}$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0,1]$ for all $i=1,\ldots,n$ (where n is the number of decision variables).

The size of $\mathbf{x}_|$ vector is chosen as 10 and the total number of variables is identical as in (DTLZ5). In this problem, the $\mathcal{PS}$ possesses the same characteristics as (DTLZ5). However, the difficulty to reach optimal solutions is augmented. The above modification of the g function makes that MOEAs have difficulties to converge to the $\mathcal{PF}$ as in (DTLZ5). In this case, MOEAs tend to find a dominated surface instead of the curve that corresponds to the $\mathcal{PF}$.

DTLZ7. This problem consists in minimizing:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ f_2(\mathbf{x}) &= x_2 \\ &\vdots \\ f_{m-1}(\mathbf{x}) &= x_{m-1} \\ f_m(\mathbf{x}) &= (1+g(\mathbf{x}_m))h(f_1,f_2,\ldots,f_{m-1},g) \end{aligned} \tag{DTLZ7}$$

such that:

$$g(\mathbf{x}_m) = 1+\frac{9}{|\mathbf{x}_m|}\sum_{x_i\in x_m} x_i$$

$$h(f_1, f_2, \ldots, f_{m-1}, g) = m - \sum_{i=1}^{m-1} \left[\frac{f_i}{1+g}(1 + \sin(3\pi f_i)) \right]$$

where m is the number of objective functions, $\mathbf{x}_m$ represents the last k variables of the decision vector $\mathbf{x}$, and $x_i \in [0, 1]$ for all $i = 1, \ldots, n$ (where n is the number of decision variables).

This test problem has 2^{m-1} disconnected Pareto optimal regions in the search space. The functional g requires $k = |\mathbf{x}_m|$ decision variables and the total number of variables is $n = m + k - 1$. It is suggested the use $k = 20$. The Pareto optimal solutions correspond to $\mathbf{x}_m = \mathbf{0}$. This problem tests an algorithm's ability to maintain subpopulations in different Pareto optimal regions.

A.4 Walking-Fish-Group Test Problems

The WFG test problems [63] apply a set of sequential transformations to the vector of decision variables. Each transformation adds a characteristic to the MOP. The transformations used in WFG test problems to define the shape of the $\mathcal{PF}$ are:

$$
\begin{aligned}
\mathrm{linear}_1(x_1,\cdots,x_{M-1}) &= \prod_{i=1}^{M-1} x_i \\
\mathrm{linear}_{m=2:M-1}(x_1,\cdots,x_{M-1}) &= \left(\prod_{i=1}^{M-m} x_i\right)(1-x_{M-m-1}) \\
\mathrm{linear}_M(x_1,\cdots,x_{M-1}) &= 1-x_1 \\
\mathrm{convex}_1(x_1,\cdots,x_{M-1}) &= \prod_{i=1}^{M-1}\left(1-\cos\left(\frac{\pi}{2}x_i\right)\right) \\
\mathrm{convex}_{m=2:M-1}(x_1,\cdots,x_{M-1}) &= \left(\prod_{i=1}^{M-m}\left(1-\cos\left(\frac{\pi}{2}x_i\right)\right)\right)\left(1-\sin\left(\frac{\pi}{2}x_{M-m+1}\right)\right) \\
\mathrm{convex}_1(x_1,\cdots,x_{M-1}) &= 1-\sin\left(\frac{\pi}{2}x_1\right) \\
\mathrm{concave}_1(x_1,\cdots,x_{M-1}) &= \prod_{i=1}^{M-1}\sin\left(\frac{\pi}{2}x_i\right) \\
\mathrm{concave}_{m=2:M-1}(x_1,\cdots,x_{M-1}) &= \left(\prod_{i=1}^{M-m}\sin\left(\frac{\pi}{2}x_i\right)\right)\cos\left(\frac{\pi}{2}x_{M-m+1}\right) \\
\mathrm{concave}_1(x_1,\cdots,x_{M-1}) &= \cos\left(\frac{\pi}{2}x_1\right) \\
\mathrm{mixed}_M(x_1,\cdots,x_{M-1}) &= \left(1-x_1-\frac{\cos(2A\pi x_1+\pi/2)}{2A\pi}\right)^{\alpha} \\
\mathrm{disc}_M(x_1,\cdots,x_{M-1}) &= 1-x_1^{\alpha}\cos^2(Ax_1^{\beta}\pi)
\end{aligned}
$$

According to Huband et al. [63], it is possible to add more characteristics to increase the difficulty to the problem. Such characteristics are presented below:

$$
\begin{aligned}
b\_poly(y,\alpha) &= y^{\alpha} \\
b\_flat(y,A,B,C) &= A + \min(0,\lfloor y-B\rfloor)\frac{A(B-y)}{B} \\
&\quad - \min(0,\lfloor C-y\rfloor)\frac{(1-A)(y-C)}{1-C} \\
b\_param(y,u(y'),A,B,C) &= y^{B+(C-B)(A-(1-2u(y'))|\lfloor 0.5-u(y')+A\rfloor|)} \\
s\_linear(y,A) &= \frac{|y-A|}{|\lfloor A-y\rfloor + A|} \\
s\_decept(y,A,B,C) &= 1+(|y-A|-B) \\
&\quad \left(\frac{\lfloor y-A+B\rfloor(1-C+\frac{A-B}{B})}{A-B} + \right. \\
&\quad \left. \frac{\lfloor A+B-y\rfloor(1-C+\frac{1-A-B}{B})}{1-A-B} + \frac{1}{B}\right) \\
s\_multi(y,A,B,C) &= \left(1+\cos\left((4A+2)\pi\left(0.5-\frac{|y-C|}{2(\lfloor C-y\rfloor + C)}\right)\right)\right. \\
&\quad \left. +4B\left(\frac{|y-C|}{2(\lfloor C-y\rfloor + C)}\right)^2\right)/(b+2) \\
r\_sum(\mathbf{y},\mathbf{w}) &= \frac{\sum_{i=1}^{|y|} w_i y_i}{\sum_{i=1}^{|y|} w_i} \\
r\_nonsep(\mathbf{y},A) &= \frac{\sum_{j=1}^{|y|}\left(y_j + \sum_{k=0}^{A-2}|y_j - y_{1+(j+k)\bmod|y|}|\right)}{\frac{|y|}{A}\lceil\frac{A}{2}\rceil(1+2A-2\lceil\frac{A}{2}\rceil)}
\end{aligned}
$$

Considering the above transformations, the nine WFG test problems are described below.

WFG1. This problem is separable and unimodal, but it has a flat region and is strongly biased toward small values of the variables, which makes it very difficult for some MOEAs. The problem consists in minimizing:

$$
\begin{aligned}
f_{m=1:M-1}(\mathbf{x}) &= x_M + S_m convex_m(x_1,\cdots,x_{M-1}) \\
f_M(\mathbf{x}) &= x_M + S_m mixed_M(x_1,\cdots,x_{M-1})
\end{aligned}
\quad \text{(WFG1)}
$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1},\cdots,y'_{ik/(M-1)}],\right. \\
&\quad \left.[2((i-1)k/(M-1)+1),\cdots,2ik/(M-1)]\right) \\
y_M &= r_sum\left([y'_{k+1},\cdots,y'_n],[2(k+1),\cdots,2n]\right) \\
y'_{i=1:n} &= b_poly\left(y''_i,0.02\right) \\
y''_{i=1:k} &= y'''_i \\
y''_{i=k+1:n} &= b_flat\left(y'''_i,0.8,0.75,0.85\right) \\
y'''_{i=1:k} &= z_{i,[0,1]} \\
y'''_{i=k+1:n} &= s_linear\left(z_{i,[0,1]},0.35\right)
\end{aligned}$$

WFG2. This problem is unseparable and multimodal. WFG2 consists in minimizing:

$$\begin{aligned}
f_{m=1:M-1}(\mathbf{x}) &= x_M + S_m convex_m\left(x_1,\cdots,x_{M-1}\right) \\
f_M(\mathbf{x}) &= x_M + S_m disc_M\left(x_1,\cdots,x_{M-1}\right)
\end{aligned} \qquad \text{(WFG2)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1},\cdots,y'_{ik/(M-1)}],[1,\cdots,1]\right) \\
y_M &= r_sum\left([y'_{k+1},\cdots,y'_{k+l/2}],[1,\cdots,1]\right) \\
y'_{i=1:k} &= y''_i \\
y'_{i=k+1:k+l/2} &= r_nonsep\left([y''_{k+2(i-k)-1},y''_{k+2(i-k)}],2\right) \\
y''_{i=1:k} &= z_{i,[0,1]} \\
y''_{i=k+1:n} &= s_linear\left(z_{i,[0,1]},0.35\right)
\end{aligned}$$

WFG3. This problem is unseparable but unimodal. It has a degenerated $\mathcal{PF}$ (the dimensionality of the $\mathcal{PF}$ is $M-2$). The problem consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m linear_m\left(x_1,\cdots,x_{M-1}\right) \qquad \text{(WFG3)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1},\cdots,y'_{ik/(M-1)}],[1,\cdots,1]\right) \\
y_M &= r_sum\left([y'_{k+1},\cdots,y'_{k+l/2}],[1,\cdots,1]\right) \\
y'_{i=1:k} &= y''_i \\
y'_{i=k+1:k+l/2} &= r_nonsep\left([y''_{k+2(i-k)-1},y''_{k+2(i-k)}],2\right) \\
y''_{i=1:k} &= z_{i,[0,1]} \\
y''_{i=k+1:n} &= s_linear\left(z_{i,[0,1]},0.35\right)
\end{aligned}$$

WFG4. This problem is separable, but highly multimodal. This, and the rest of the problems from this benchmark have concave $\mathcal{PF}$s. The problem consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\,(x_1, \cdots, x_{M-1}) \quad \text{(WFG4)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}], [1, \cdots, 1]\right)\\
y_M &= r_sum\left([y'_{k+1}, \cdots, y'_n], [1, \cdots, 1]\right)\\
y''_{i=1:n} &= s_linear\left(z_{i,[0,1]}, 30, 10, 0.35\right)
\end{aligned}$$

WFG5. This is a deceptive problem and separable. The problem consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\,(x_1, \cdots, x_{M-1}) \quad \text{(WFG5)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}], [1, \cdots, 1]\right)\\
y_M &= r_sum\left([y'_{k+1}, \cdots, y'_n], [1, \cdots, 1]\right)\\
y''_{i=1:n} &= s_decept\left(z_{i,[0,1]}, 0.35, 0.001, 0.05\right)
\end{aligned}$$

WFG6. This problem is unseparable and consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\,(x_1, \cdots, x_{M-1}) \quad \text{(WFG6)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_nonsep\left([y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}], k/(M-1)\right)\\
y_M &= r_nonsep\left([y'_{k+1}, \cdots, y'_n], l\right)\\
y'_{i=1:k} &= z_{i,[0,1]}\\
y'_{i=k+1:n} &= s_linear\left(z_{i,[0,1]}, 0.35\right)
\end{aligned}$$

WFG7. This problem is also separable and unimodal. The problem consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\,(x_1, \cdots, x_{M-1}) \quad \text{(WFG7)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left([y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}], [1, \cdots, 1]\right)\\
y_M &= r_sum\left([y'_{k+1}, \cdots, y'_n], [1, \cdots, 1]\right)\\
y'_{i=1:k} &= y''_i\\
y'_{i=k+1:n} &= s_linear\left(y''_i, 0.35\right)\\
y''_{i=k+1:n} &= b_param(z_{i,[0,1]}, r_sum([z_{i+1,[0,1]}, \ldots, z_{n,[0,1]}],\\
&\quad [1, \ldots, 1]), 0.98/49.98, 0.02, 50)
\end{aligned}$$

WFG8. This problem is also unseparable and consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\left(x_1, \cdots, x_{M-1}\right) \quad \text{(WFG8)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_sum\left(\left[y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}\right], [1, \cdots, 1]\right) \\
y_M &= r_sum\left(\left[y'_{k+1}, \cdots, y'_n\right], [1, \cdots, 1]\right) \\
y'_{i=1:k} &= y''_i \\
y'_{i=k+1:n} &= s_linear\left(y''_i, 0.35\right) \\
y''_{i=k+1:n} &= z_{i,[0,1]} \\
y''_{i=k+1:n} &= b_param(z_{i,[0,1]}, r_sum([z_{i+1,[0,1]}, \ldots, z_{n,[0,1]}], \\
&\quad [1, \ldots, 1]), 0.98/49.98, 0.02, 50)
\end{aligned}$$

WFG9. This problem is also unseparable and consists in minimizing:

$$f_{m=1:M}(\mathbf{x}) = x_M + S_m concave_m\left(x_1, \cdots, x_{M-1}\right) \quad \text{(WFG9)}$$

where:

$$\begin{aligned}
y_{i=1:M-1} &= r_nonsep\left(\left[y'_{(i-1)k/(M-1)+1}, \cdots, y'_{ik/(M-1)}\right], k/(M-1)\right) \\
y_M &= r_nonsep\left(\left[y'_{k+1}, \cdots, y'_n\right], 1\right) \\
y'_{i=1:k} &= s_decept\left(y''_i, 0.35, 0.001, 0.05\right) \\
y'_{i=k+1:n} &= s_multi(y''_i, 30, 95, 0.35) \\
y''_{i=k+1:n} &= b_param(z_{i,[0,1]}, r_sum([z_{i+1,[0,1]}, \ldots, z_{n,[0,1]}], \\
&\quad [1, \ldots, 1]), 0.98/49.98, 0.02, 50)
\end{aligned}$$

B

Airfoil Shape Optimization

In aeronautics, aerodynamics plays an important role in any aircraft design problem. Therefore, aerodynamic shape optimization is a crucial task and has been extensively studied and developed. In this design area, designers are frequently faced with the problem of considering not only a single design objective, but several of them, i.e., the designer needs to solve a multi-objective optimization problem.

In recent years, *Multi-Objective Evolutionary Algorithms (MOEAs)* have gained popularity as an optimization method in aeronautics, mainly because of their simplicity, their ease of use and their suitability to be coupled to specialized numerical simulation tools. However, the whole optimization process becomes costly in terms of computational time, mainly because many high-fidelity *Computational Fluid Dynamics (CFD)* simulations are needed. One option to alleviate this condition, is to design mechanisms for reducing this computational cost by exploiting the properties that, mathematical programming techniques possess.

Our case study consists of the multi-objective optimization of an airfoil shape problem adapted from [131] (called here MOPRW). This problem corresponds to the airfoil shape optimization of a standard-class glider, aiming to obtain an optimum performance for a sailplane.

B.1 Problem Statement

Two conflicting objective functions are defined in terms of a sailplane average weight and operating conditions [131]. They are defined as:

$$\begin{aligned} &\text{i)} \quad \text{Minimize} : C_D/C_L \\ &\qquad \text{s.t.} C_L = 0.63, Re = 2.04 \cdot 10^6, M = 0.12 \\ &\text{ii)} \quad \text{Minimize} : C_D/C_L^{3/2} \\ &\qquad \text{s.t.} C_L = 1.05, Re = 1.29 \cdot 10^6, M = 0.08 \end{aligned} \qquad \text{(MOPRW)}$$

where C_D/C_L and $C_D/C_L^{3/2}$ correspond to the inverse of the glider's gliding ratio and sink rate, respectively. Both are important performance measures for this aerodynamic optimization problem. C_D and C_L are the drag and lift coefficients.

The aim is to maximize the gliding ratio (C_L/C_D) for objective (*i*), while minimizing the sink rate in objective (*ii*). Each of these objectives is evaluated at different prescribed flight conditions, given in terms of Mach and Reynolds numbers. The aim of solving this *Multi-objective Optimization Problem (MOP)* is to find a better airfoil shape, which improves a reference design.

B.1.1 Geometry Parametrization

In the present case study, the *PARametric SECtion (PARSEC)* airfoil representation [124] was adopted. Fig. B.1 illustrates the 11 basic parameters used for this representation: r_{le} leading edge radius, X_{up}/X_{lo} location of maximum thickness for upper/lower surfaces, Z_{up}/Z_{lo} maximum thickness for upper/lower surfaces, Z_{xxup}/Z_{xxlo} curvature for upper/lower surfaces, at maximum thickness locations, Z_{te} trailing edge coordinate, ΔZ_{te} trailing edge thickness, α_{te} trailing edge direction, and β_{te} trailing edge wedge angle.

For the present case study, the modified PARSEC geometry representation adopted allows us to define independently the leading edge radius, both for upper and lower surfaces. Thus, a total of 12 variables are used. Their allowable ranges are defined in Table 12.

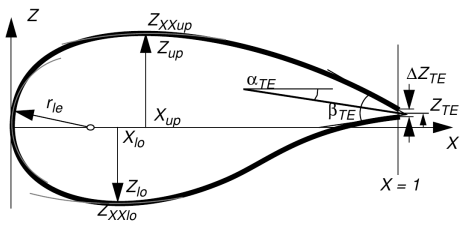


Figure B.1.: PARSEC airfoil parametrization.

Table 12.: Parameter ranges for modified PARSEC airfoil representation

Design Variable	Lower Bound	Upper Bound
r_{leup}	0.0085	0.0126
r_{lelo}	0.0020	0.0040
α_{te}	7.0000	10.0000
β_{te}	10.0000	14.0000
Z_{te}	-0.0060	-0.0030
ΔZ_{te}	0.0025	0.0050
X_{up}	0.4100	0.4600
Z_{up}	0.1100	0.1300
Z_{xxup}	-0.9000	-0.7000
X_{lo}	0.2000	0.2600
Z_{lo}	-0.0230	-0.0150
Z_{xxlo}	0.0500	0.2000

The PARSEC airfoil geometry representation uses a linear combination of shape functions for defining the upper and lower surfaces. These linear combinations are given by:

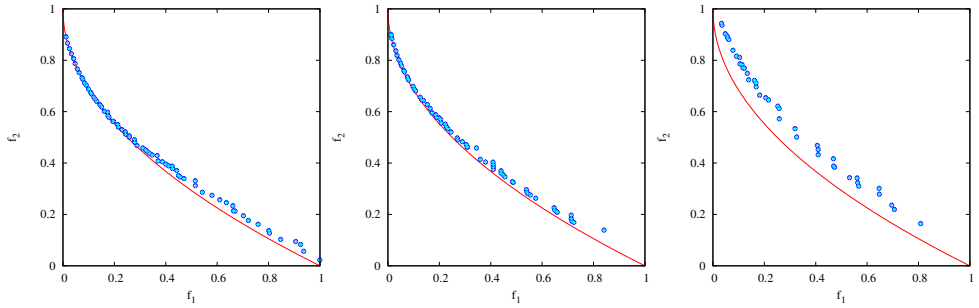
$$Z_{upper} = \sum_{n=1}^{6} a_n x^{\frac{n-1}{2}}, \quad Z_{lower} = \sum_{n=1}^{6} b_n x^{\frac{n-1}{2}} \tag{B.1}$$

In the above equations, the coefficients a_n, and b_n are determined as functions of the 12 described geometric parameters, by solving two systems of linear equations, one for each surface. It is important to note that the geometric parameters r_{leup}/r_{lelo}, X_{up}/X_{lo}, Z_{up}/Z_{lo}, Z_{xxup}/Z_{xxlo}, Z_{te}, ΔZ_{te}, α_{te}, and β_{te} are the actual design variables in the optimization process, and that the coefficients a_n, b_n serve as intermediate variables for interpolating the airfoil's coordinates, which are used by the CFD solver (we used the Xfoil CFD code [32]) for its discretization process.

C

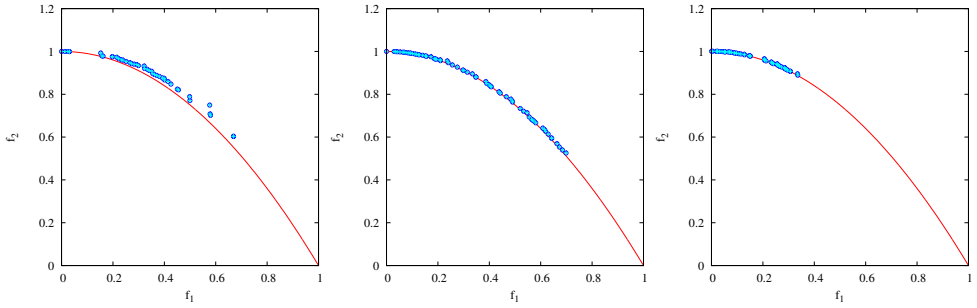
Pareto Front Approximations for ZDT Test Suite

In this appendix, we show the plots of the final approximations to the *Pareto front (PF)* obtained by the *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*, the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS)* and the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II)*. For an easy comparison, Figures C.1–C.5 present the plots of the nondominated solutions found by the different algorithms for each *Zitzler-Deb-Thiele (ZDT)* test problem. Each plot corresponds to the run with the value nearest to the mean value of the *Hypervolume* (I_H) performance measure reported in Chapter 7 (see Table 8) for each test problem. According to the comparative study presented in Chapter 7, the plots reported here, show the performance of the different algorithms, when they are restricted to perform 10,000 fitness function evaluations.



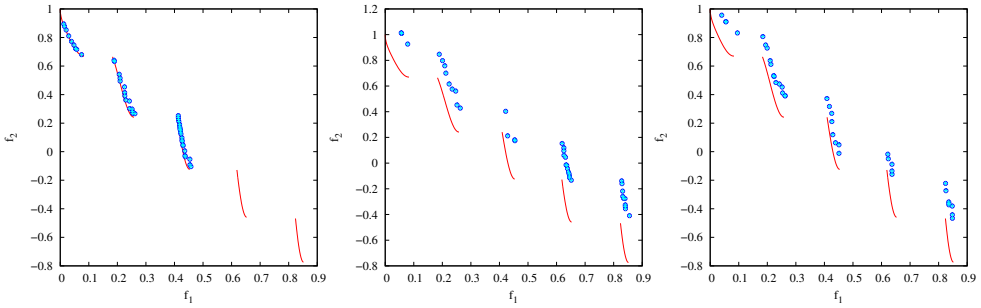
A) MOEA/D+LS-II B) MOEA/D+LS C) MOEA/D

Figure C.1.: Comparison of the PF approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT1 test problem.



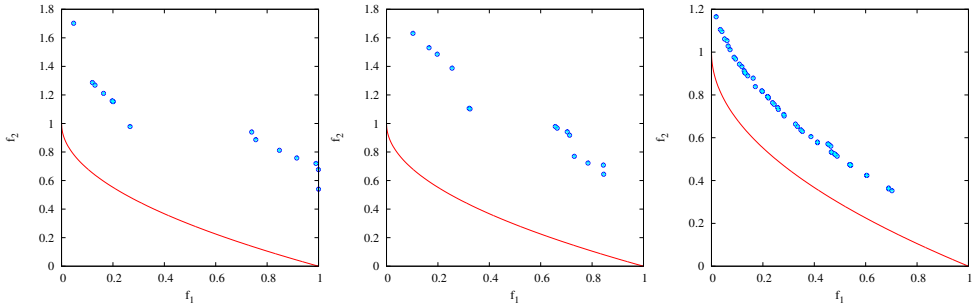
A) MOEA/D+LS-II B) MOEA/D+LS C) MOEA/D

Figure C.2.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT2 test problem.



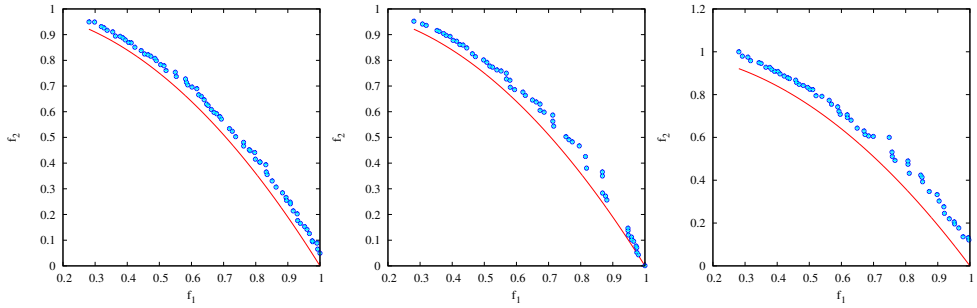
A) MOEA/D+LS-II B) MOEA/D+LS C) MOEA/D

Figure C.3.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT3 test problem.



A) MOEA/D+LS-II B) MOEA/D+LS C) MOEA/D

Figure C.4.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT4 test problem.



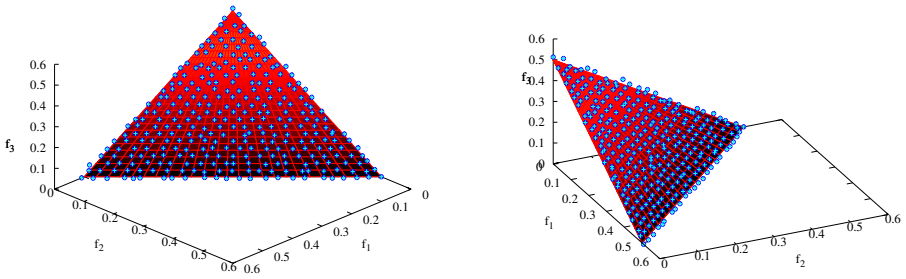
A) MOEA/D+LS-II B) MOEA/D+LS C) MOEA/D

Figure C.5.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the ZDT6 test problem.

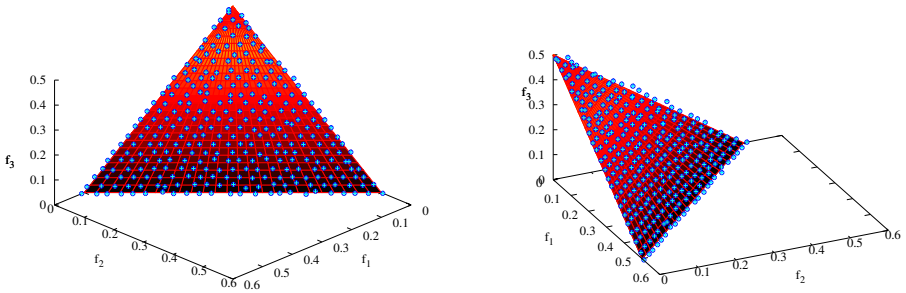
D

Pareto Front Approximations for DTLZ Test Suite

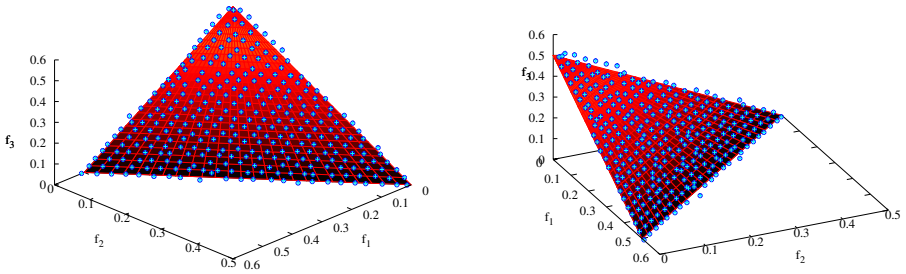
In this appendix, we show the plots of the final approximations to the *Pareto front (PF)* obtained by the *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*, the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS)* and the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II)*. For an easy comparison of results, Figures D.1–D.7 present the plots of the nondominated solutions found by the different algorithms for each *Deb-Thiele-Laumanns-Zitzler (DTLZ)* test problem. Each plot corresponds to the run with the value nearest to the mean value of the *Hypervolume* (I_H) performance measure reported in Chapter 7 (see Table 8) for each test problem. In order to appreciate the results of each algorithm in a better way, the PF approximations are presented in two different perspectives. First, we show the PF face of the obtained solutions and then, a rotation of these set of solutions is shown. The plots reported here, show the performance of the different algorithms, when they are restricted to perform 30,000 fitness function evaluations.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ1 test problem

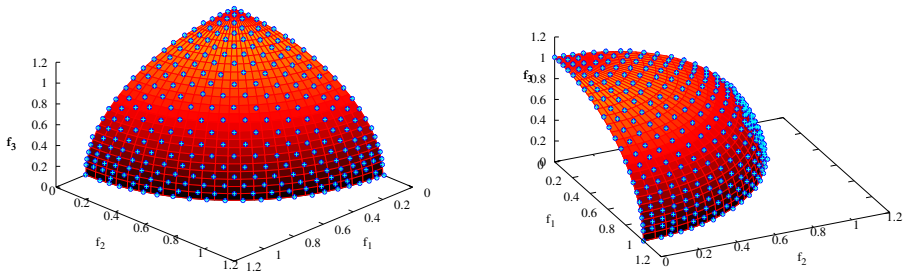


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ1 test problem

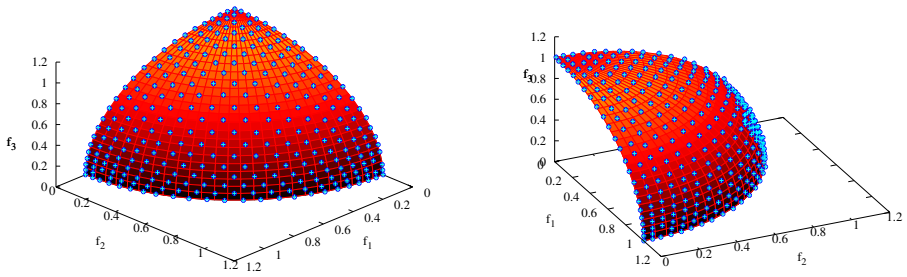


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ1 test problem

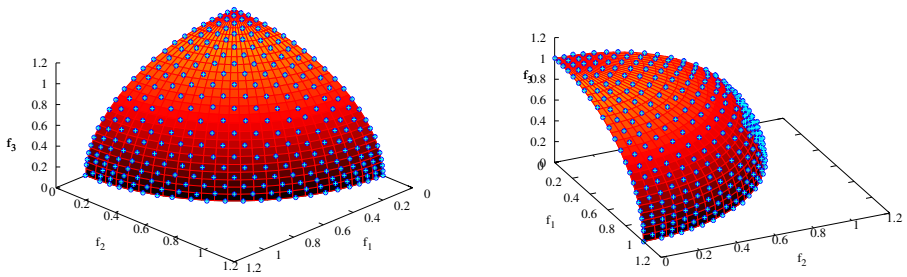
Figure D.1.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ1 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ2 test problem

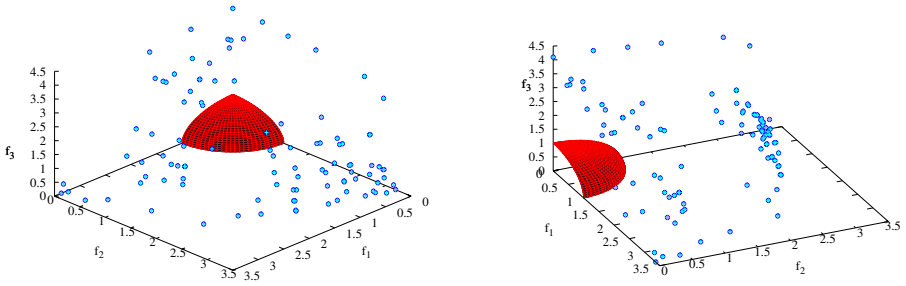


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ2 test problem

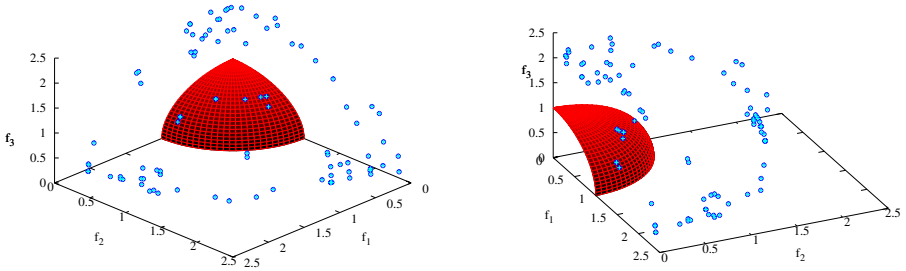


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ2 test problem

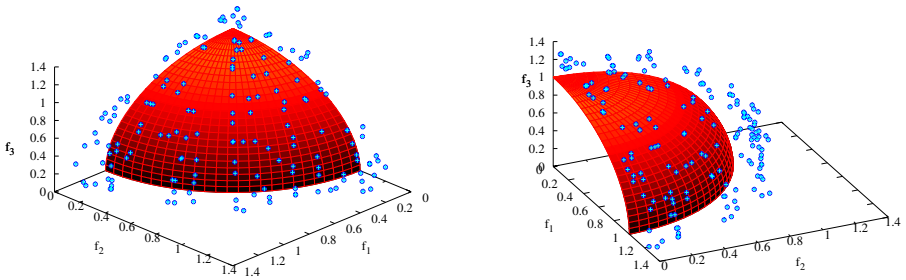
Figure D.2.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ2 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ3 test problem

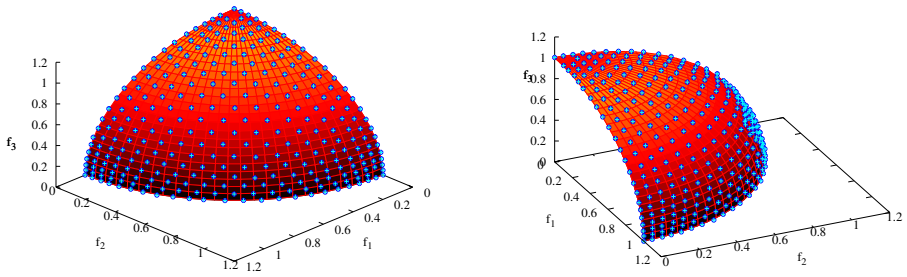


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ3 test problem

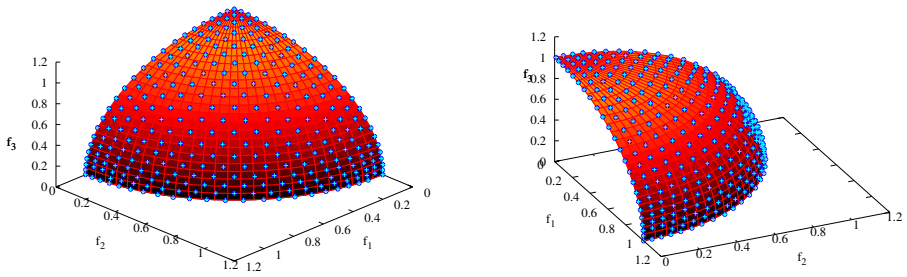


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ3 test problem

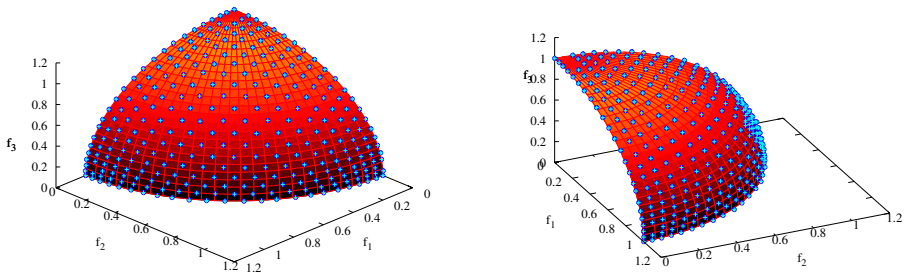
Figure D.3.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ3 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ4 test problem

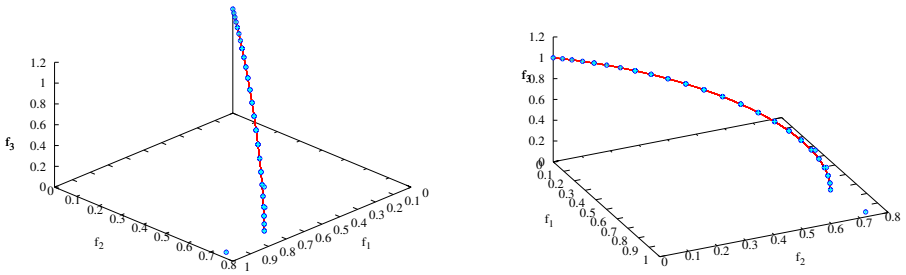


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ4 test problem

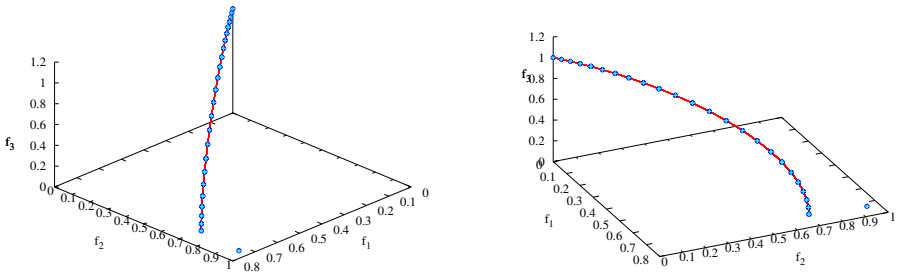


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ4 test problem

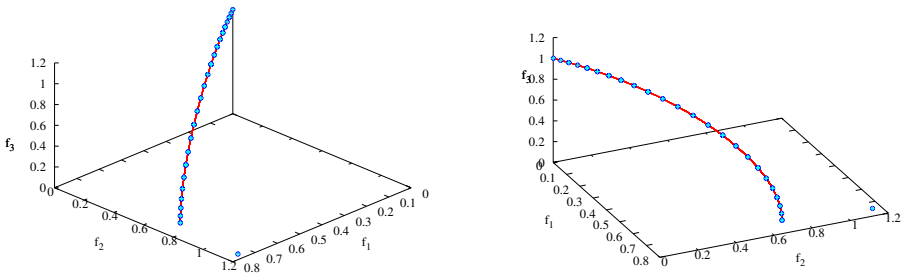
Figure D.4.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ4 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ5 test problem

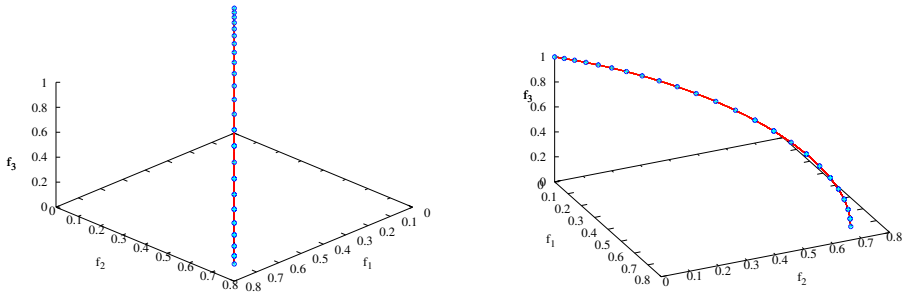


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ5 test problem

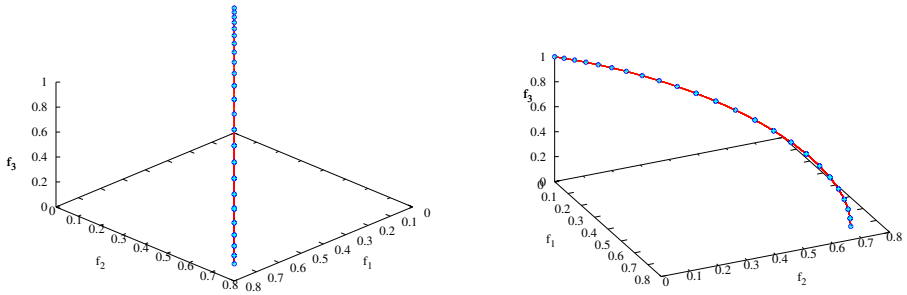


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ5 test problem

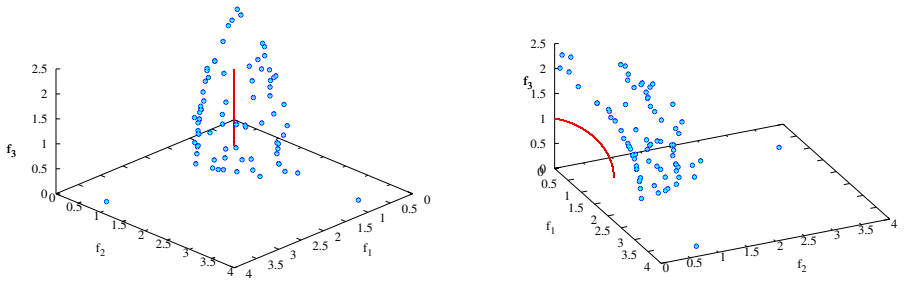
Figure D.5.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ5 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ6 test problem

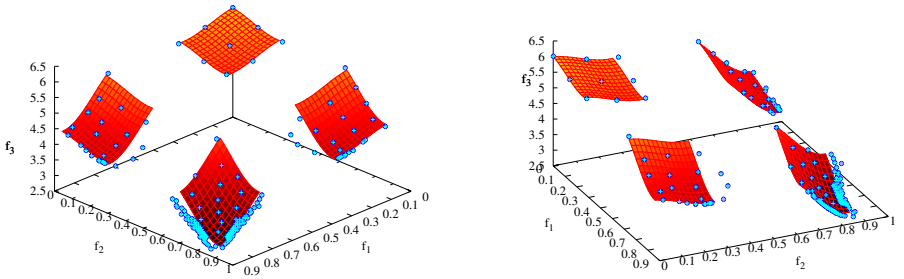


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ6 test problem

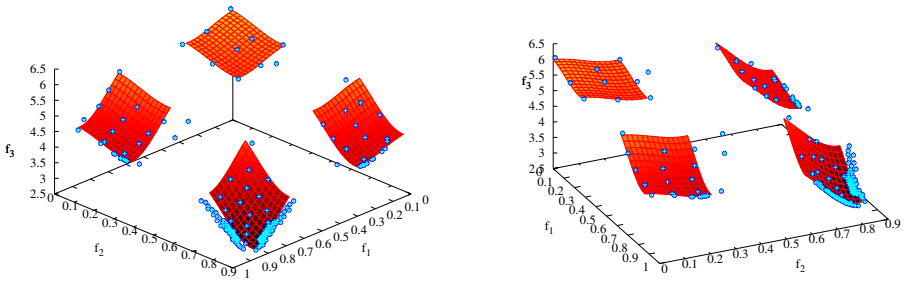


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ6 test problem

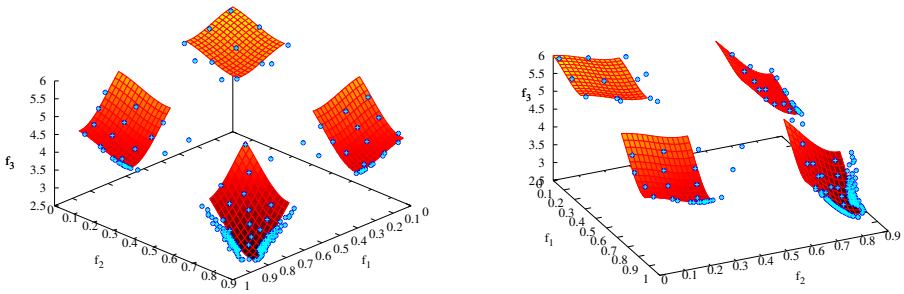
Figure D.6.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ6 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the DTLZ7 test problem



B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the DTLZ7 test problem



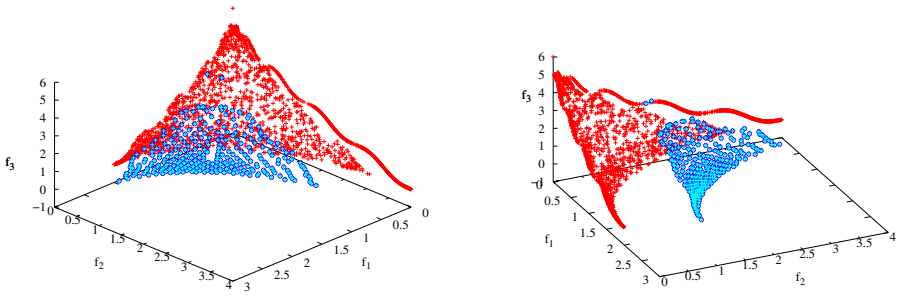
C) $\mathcal{PF}$ approximation obtained by MOEA/D for the DTLZ7 test problem

Figure D.7.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the DTLZ7 test problem.

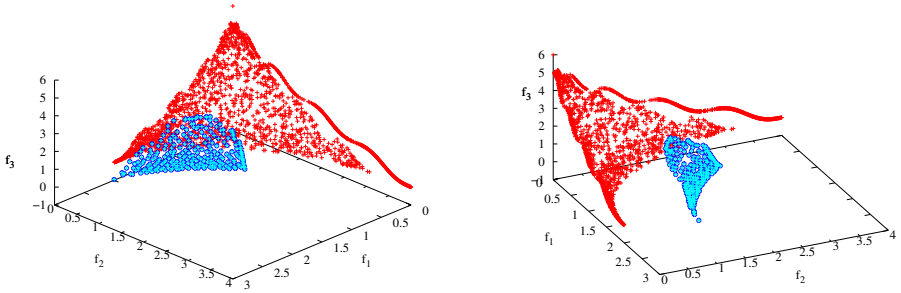
E

Pareto Front Approximations for WFG Test Suite

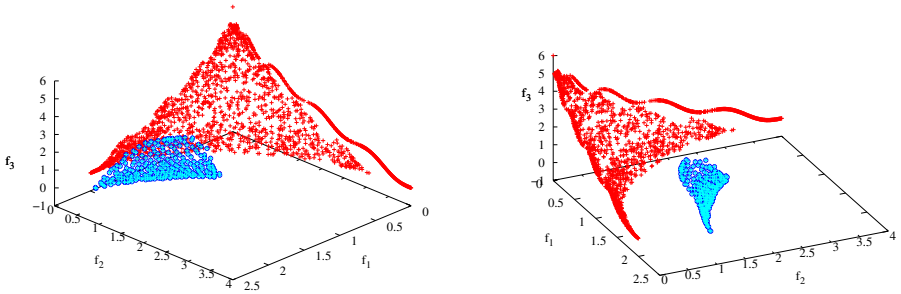
In this appendix, we show the plots of the final approximations to the *Pareto front (PF)* obtained by the *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*, the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search (MOEA/D+LS)* and the *Multi-Objective Evolutionary Algorithm based on Decomposition with Local Search II (MOEA/D+LS-II)*. For an easy comparison of results, Figures D.1–D.7 present the plots of the nondominated solutions found by the different algorithms for each *Walking-Fish-Group (WFG)* test problem. Each plot corresponds to the run with the value nearest to the mean value of the *Hypervolume* (I_H) performance measure reported in Chapter 7 (see Table 8) for each test problem. In order to appreciate the results of each algorithm in a better way, the $\mathcal{PF}$ approximations are presented in two different perspectives. First, we show the $\mathcal{PF}$ face of the obtained solutions and then, a rotation of these set of solutions is shown. The plots reported here, show the performance of the different algorithms, when they are restricted to perform 30,000 fitness function evaluations.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG1 test problem

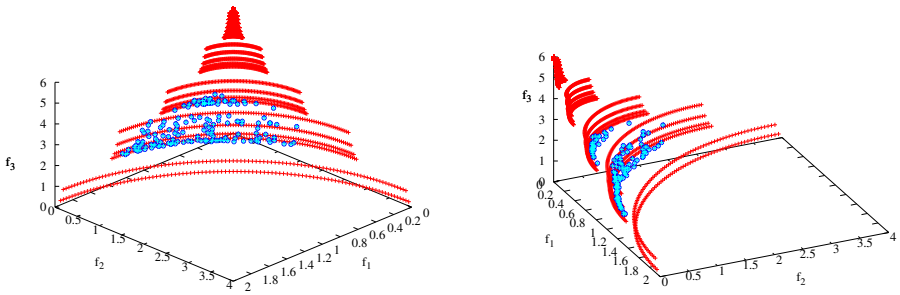


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG1 test problem

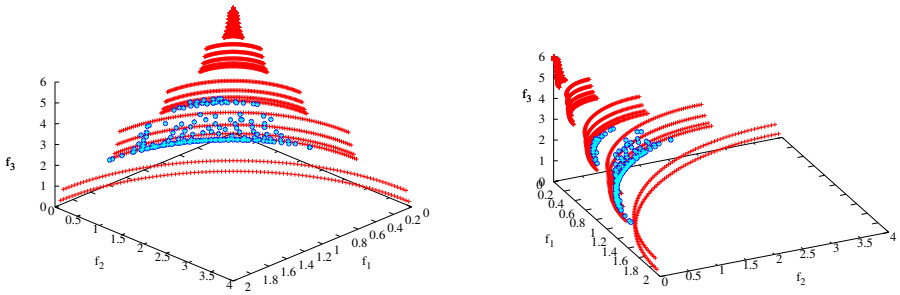


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG1 test problem

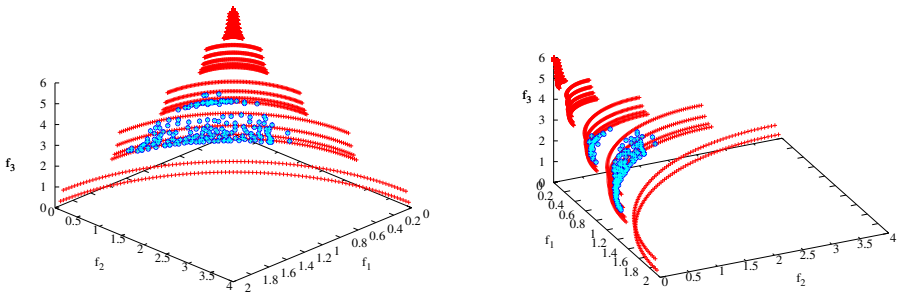
Figure E.1.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG1 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG2 test problem

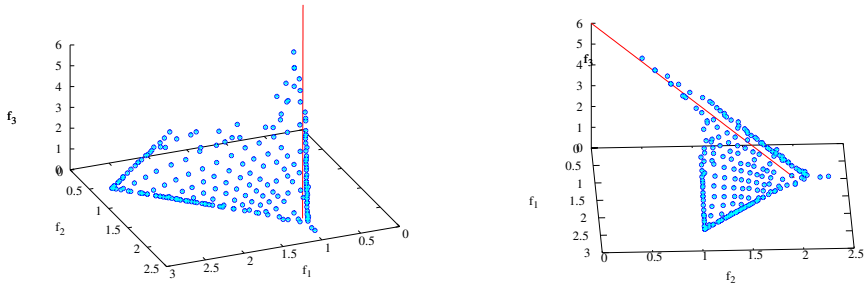


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG2 test problem

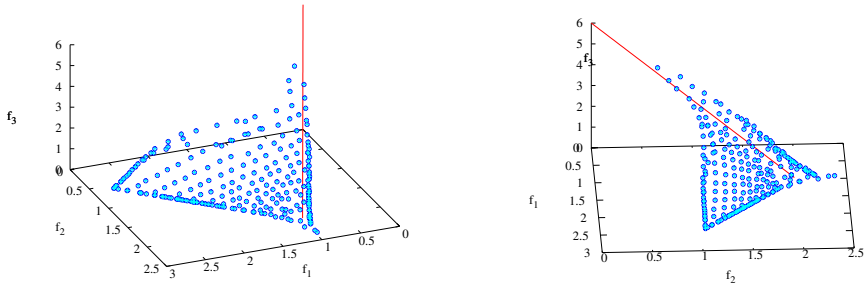


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG2 test problem

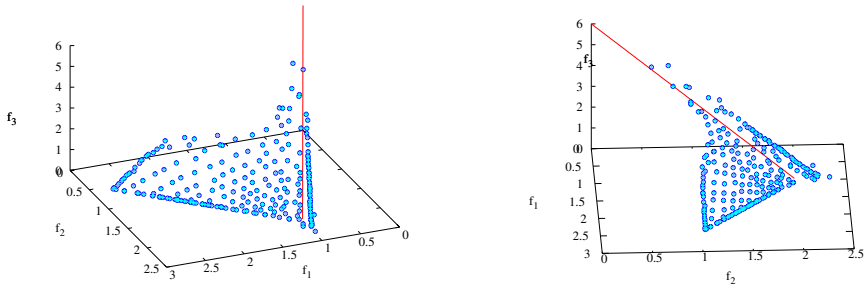
Figure E.2.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG2 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG3 test problem

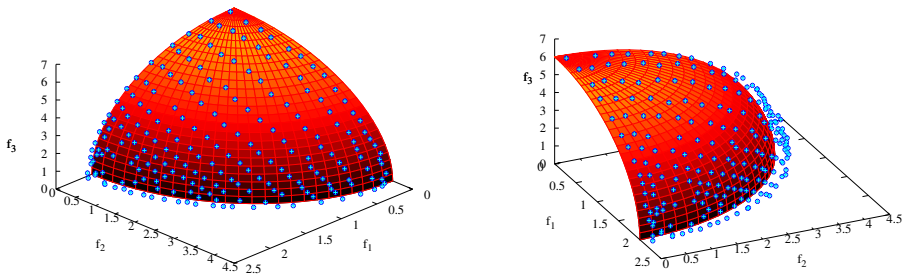


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG3 test problem

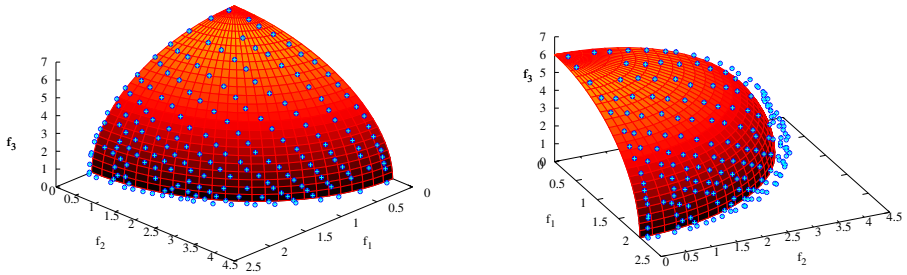


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG3 test problem

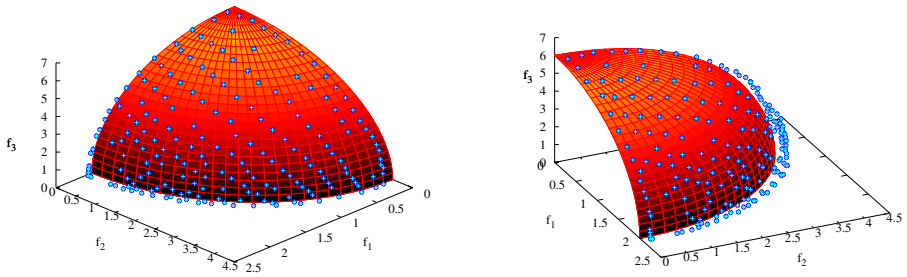
Figure E.3.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG3 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG4 test problem

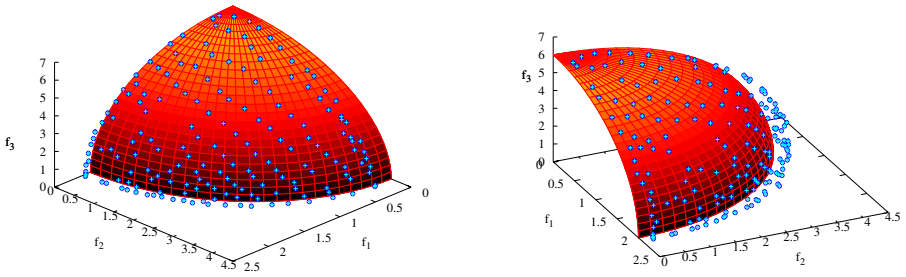


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG4 test problem

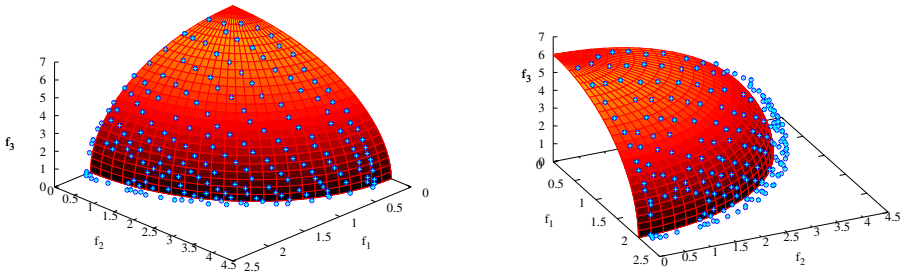


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG4 test problem

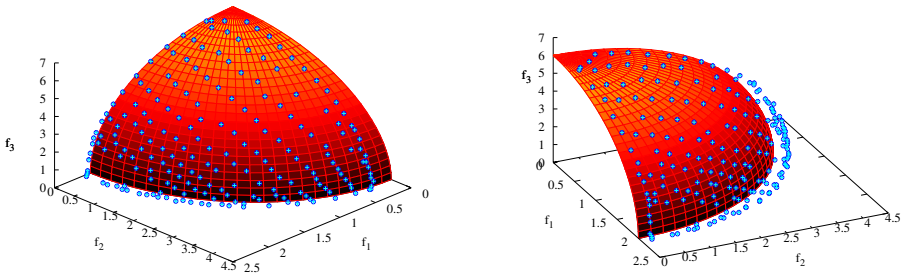
Figure E.4.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG4 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG5 test problem

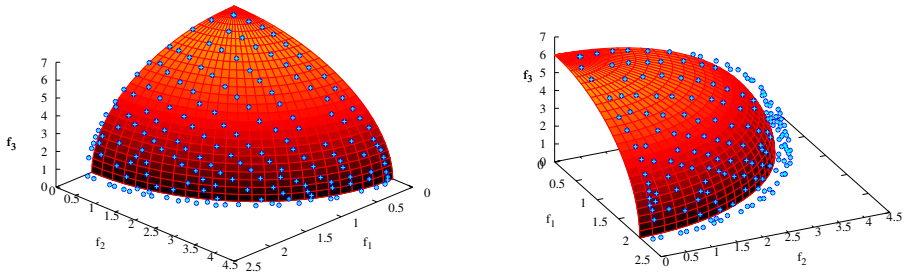


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG5 test problem

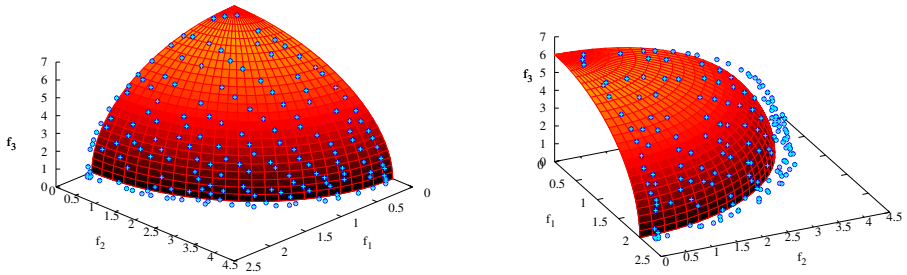


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG5 test problem

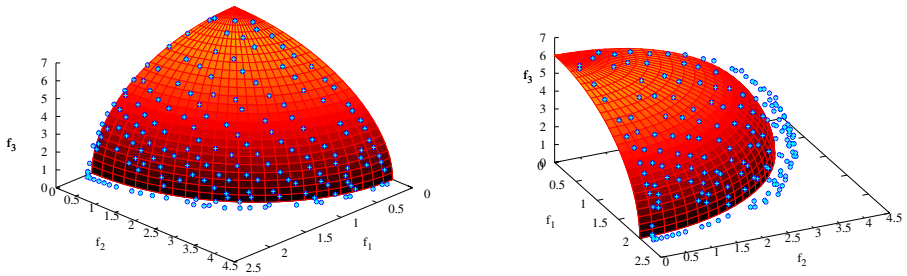
Figure E.5.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG5 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG6 test problem

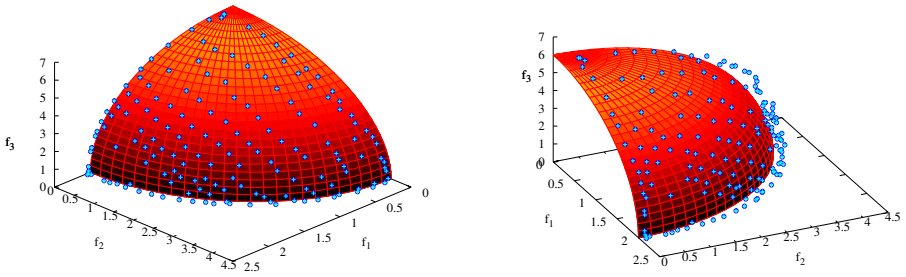


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG6 test problem

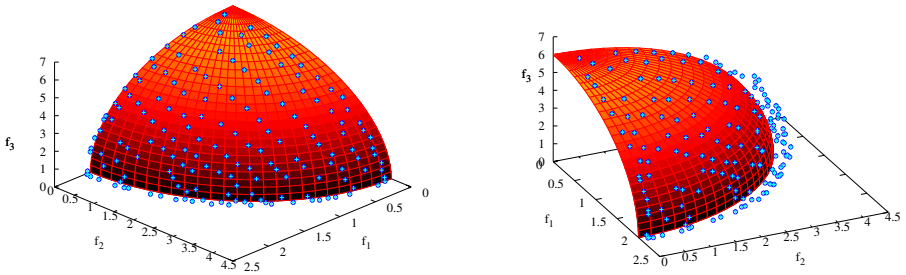


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG6 test problem

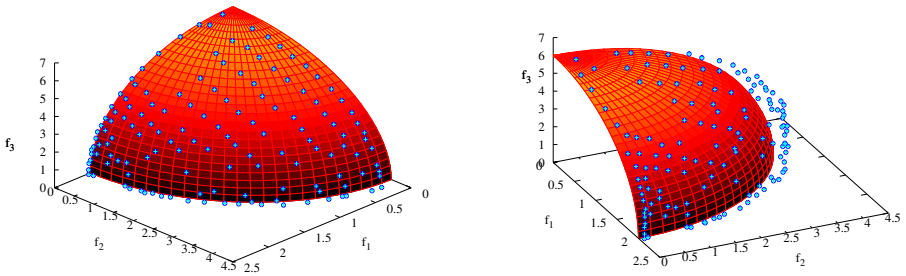
Figure E.6.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG6 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG7 test problem

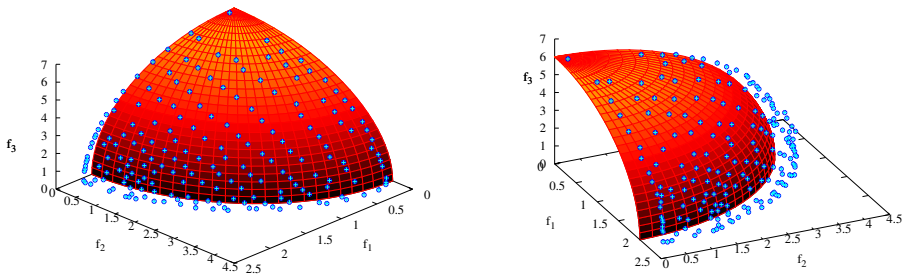


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG7 test problem

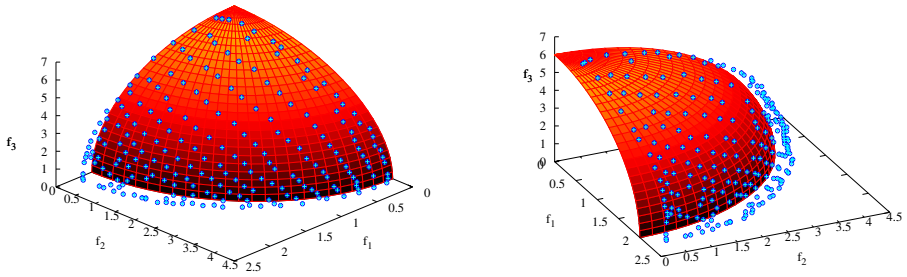


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG7 test problem

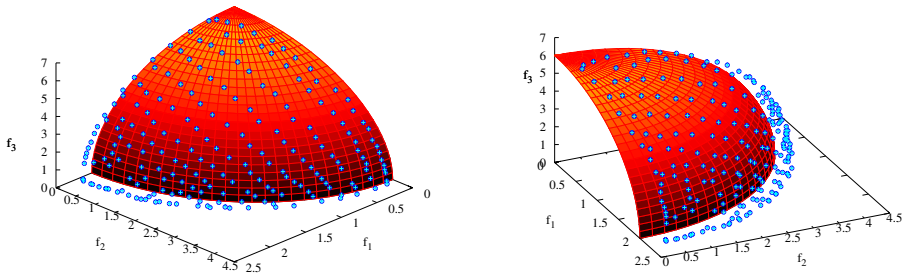
Figure E.7.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG7 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG8 test problem

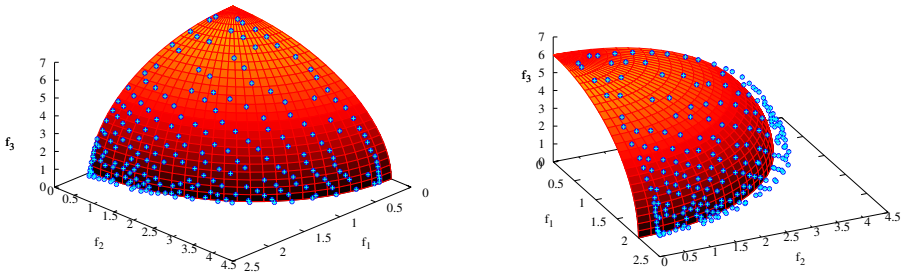


B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG8 test problem

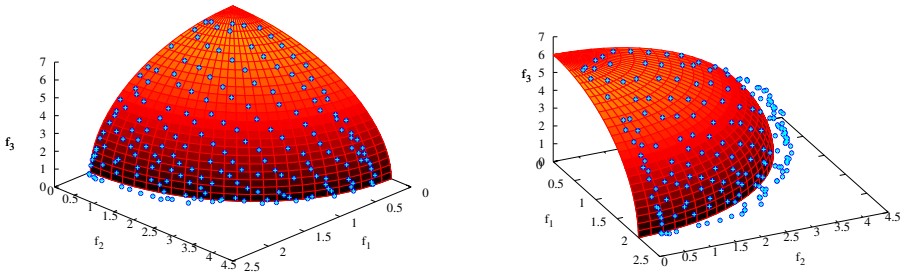


C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG8 test problem

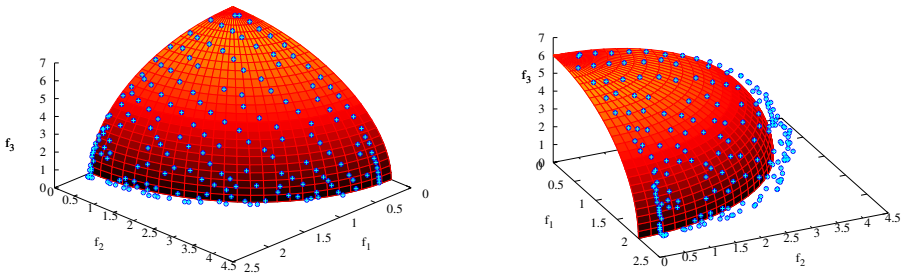
Figure E.8.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG8 test problem.



A) $\mathcal{PF}$ approximation obtained by MOEA/D+LS-II for the WFG9 test problem



B) $\mathcal{PF}$ approximation obtained by MOEA/D+LS for the WFG9 test problem



C) $\mathcal{PF}$ approximation obtained by MOEA/D for the WFG9 test problem

Figure E.9.: Comparison of the $\mathcal{PF}$ approximations obtained by MOEA/D+LS-II, MOEA/D+LS and MOEA/D for the WFG9 test problem.

Bibliography

[1] Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.

[2] Rakesh Angira and B. V. Babu. Non-dominated Sorting Differential Evolution (NSDE): An Extension of Differential Evolution for Multi-objective Optimization. In Bhanu Prasad, editor, *Proceedings of the 2nd Indian International Conference on Artificial Intelligence (IICAI)*, pages 1428–1443, 2005.

[3] Andreas Antoniou and Wu-Sheng Lu. *Practical Optimization: Algorithms and Engineering Applications*. Springer, 2007.

[4] Árpád Bűrmen, Janez Puhan, and Tadej Tuma. Grid Restrained Nelder-Mead Algorithm. *Computational Optimization and Applications*, 34:359–375, July 2006.

[5] Thomas Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Institute of Physics Publishing and Oxford University Press, 1997.

[6] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[7] Martin Brown and R. E. Smith. Directed multi-objective optimization. *International Journal of Computers, Systems and Signals*, 6(1):3–17, 2005.

[8] David Byatt. A Convergent Variants of the Nelder-Mead Algorithm. Master's thesis, University of Canterbury, 2000.

[9] A. Caponio and F. Neri. Integrating cross-dominance adaption in multi-objective memetic algorithms. In C.-K. Goh, Y.-S. Ong, and K. C. Tan, editors, *Multi-Objective Memetic Algorithms*, pages 325–351. Springer, Studies in Computational Intelligence , Vol. 171, 2009.

[10] Augustin-Louis Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Compte Rendu des S'eances de L'Acad'emie des Sciences XXV*, S'erie A(25):536–538, October 1847.

[11] Abraham Charnes and William Wager Cooper. *Management Models and Industrial Applications of Linear Programming*, volume 1. John Wiley & Sons Inc, New York, December 1961.

[12] Rachid Chelouah and Patrick Siarry. Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148(2):335–348, July 2003.

[13] Carlos A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, Agosto 1999.

[14] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[15] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[16] J. L. Cohon and D. H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resources Research*, 11(2):208–220, 1975.

[17] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[18] I. Das. *Nonlinear Multicriteria Optimization and Robust Optimality*. PhD thesis, Rice University, Houston, Texas, 1997.

[19] I. Das and J. E. Dennis. Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[20] Lawrence Davis. Adapting operator probabilities in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[21] Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1990.

[22] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[23] Kalyanmoy Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135–161. John Wiley & Sons, Ltd, Chichester, Reino Unido, 1999.

[24] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

[25] Kalyanmoy Deb. *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall of India Pvt. Ltd, 2002.

[26] Kalyanmoy Deb and Tushar Goel. A hybrid multi-objective evolutionary approach to engineering shape design. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, EMO '01, pages 385–399, London, UK, UK, 2001. Springer-Verlag.

[27] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[28] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[29] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.

[30] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In *New ideas in optimization*, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[31] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.

[32] Mark Drela. XFOIL: An Analysis and Design System for Low Reynolds Number Aerodynamics. In *Conference on Low Reynolds Number Aerodynamics*, University Of Notre Dame, IN, June 1989.

[33] Lucien Duckstein. Multiobjective optimization in structural design: The model choice problem. In K. M. Ragsdell E. Atrek, R. H. Gallagher and O. C. Zienkiewicz, editors, *New Directions in Optimum Structural Design*, pages 459–481. John Wiley & Sons, Inc., 1984.

[34] Francis Ysidro Edgeworth. *Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences*. C. Kegan Paul and Co., London, 1881.

[35] Matthias Ehrgott. *Multicriteria Optimization*. Springer, Berlin, 2nd edition edition, June 2005.

[36] Michael T. M. Emmerich, Kyriakos Giannakoglou, and Boris Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[37] Andreas Fischer and Pradyumn Kumar Shukla. A levenberg-marquardt algorithm for unconstrained multicriteria optimization. *Oper. Res. Lett.*, 36(5):643–646, 2008.

[38] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, February 1964.

[39] J. Fliege, L. M. Graña Drummond, and B. F. Svaiter. Newton's method for multiobjective optimization. *SIAM J. on Optimization*, 20(2):602–626, May 2009.

[40] J. Fliege and B. Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[41] Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., Nueva York, 1966.

[42] Lawrence J. Fogel. *Intelligence through simulated evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[43] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[44] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction. In *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 42–52, Sheffield, UK, September 1995. IEE.

[45] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.

[46] A. M. Geoffrion, J. S. Dyer, and A. Feinberg. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Managment Science*, 19:357–368, December 1972.

[47] Chariklia A. Georgopoulou and Kyriakos C. Giannakoglou. A multi-objective metamodel-assisted memetic algorithm with strengthbased local refinement. *Engineering Optimization*, 41(10):909–923, 2009.

[48] Fred Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer Academic Publishers, 1996.

[49] Fred Glover, Miguel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[50] T. Goel and K. Deb. Hybrid Methods for Multi-Objective Evolutionary Algorithms. In Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim, and Xin Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 1, pages 188–192, Orchid Country Club, Singapore, November 2002. Nanyang Technical University.

[51] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh. An Investigation on Evolutionary Gradient Search for Multi-Objective Optimization. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 3742–3747, Hong Kong, June 2008. IEEE Service Center.

[52] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[53] John J. Grefenstette. Genesis: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165, 1984.

[54] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, December 1960.

[55] J. M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86:844–874, 1960.

[56] Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of IEEE First Annual International Conference on Neural Networks*, volume 3, pages 11–14, 1987.

[57] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, Redwood City, CA, 1990.

[58] Claus Hillermeier. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*. Birkhäuser Basel, 2000.

[59] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[60] Robert Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, 1961.

[61] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, EE. UU., 1993.

[62] Xiaolin Hu, Zhangcan Huang, and Zhongfan Wang. Hybridization of the Multi-Objective Evolutionary Algorithms and the Gradient-based Algorithms. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 870–877, Canberra, Australia, December 2003. IEEE Press.

[63] Simon Huband, Phil Hingston, Luigi Barone, and Lyndon While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, October 2006.

[64] Amitay Isaacs, Tapabrata Ray, and Warren Smith. An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization. In *ACAL*, pages 257–268, 2007.

[65] Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[66] Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. *IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews*, 28(3):392–403, August 1998.

[67] A. Jaszkiewicz. Do Multiple-Objective Metaheuristics Deliver on Their Promises? a Computational Experiment on the Set-Covering Problem. *IEEE Transactions on Evolutionary Computation*, 7(2):133–143, April 2003.

[68] Andrzej Jaszkiewicz and Roman Slowinski. The 'Light Beam Search' approach -an overview of methodology and applications. *European Journal of Operational Research*, 113(2):300–314, 1999.

[69] Dervis Karaboga. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[70] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[71] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[72] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[73] J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[74] Joshua Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, January 2006.

[75] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

[76] Joshua D. Knowles and David W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., Julio 1999. IEEE Service Center.

[77] Patrick Koch, Oliver Kramer, Günter Rudolph, and Nicola Beume. On the hybridization of sms-emoa and local search for continuous multiobjective optimization. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 603–610, New York, NY, USA, 2009. ACM.

[78] Praveen Koduru, Sanjoy Das, Stephen Welch, and Judith L. Roe. Fuzzy Dominance Based Multi-objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 356–367, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[79] Praveen Koduru, Sanjoy Das, and Stephen M. Welch. Multi-Objective Hybrid PSO Using ϵ-Fuzzy Dominance. In Dirk Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 853–860, London, UK, July 2007. ACM Press.

[80] A. K. Kolmogorov. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, 114:369–373, 1957.

[81] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium*, pages 481–492. University of California Press, 1951.

[82] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[83] Adriana Lara, Gustavo Sanchez, Carlos A. Coello Coello, and Oliver Schütze. HCS: A New Local Search Strategy for Memetic Multi-Objective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112–132, February 2010.

[84] Marco Laumanns, Günter Rudolph, and Hans-Paul Schwefel. A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study. In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature — PPSN V*, pages 241–249, Amsterdam, Holland, 1998. Springer-Verlag.

[85] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. J. Appl. Maths.*, II(2):164–168, 1944.

[86] R. P Lippmann. An introduction to computing with neural nets. *IEEE Magazine on Accoustics, Signal, and Speech Processing*, 4:4–22, April 1987.

[87] Joanna Lis and A. E. Eiben. A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 59–64, Nagoya, Japan, 1996. IEEE.

[88] Changtong Luo and Bo Yu. Low Dimensional Simplex Evolution–A Hybrid Heuristic for Global Optimization. In

Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, volume 2, pages 470–474, 2007.

[89] M. Luque, Jian-Bo Yang, and B. Wong. Project method for multiobjective optimization based on gradient projection and reference points. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(4):864 –879, july 2009.

[90] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[91] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[92] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[93] K. I. M. McKinnon. Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization*, 9(1):148–158, 1998.

[94] J.M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345 –377, mar 1995.

[95] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[96] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Dordrecht, 1999.

[97] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[98] Noura Al Moubayed, Andrei Petrovski, and John A. W. McCall. A novel smart multi-objective particle swarm optimisation using decomposition. In *PPSN (2)*, pages 1–10, 2010.

[99] H. Mukai. Algorithms for Multicriterion Optimization. *IEEE Transactions on Automatic Control*, 25(2):177–186, 1980.

[100] T. Murata, S. Kaige, and H. Ishibuchi. Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 1234–1245. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[101] Tadahiko Murata and Hisao Ishibuchi. MOGA: Multi-Objective Genetic Algorithms. In *Proceedings of the 2nd IEEE International Conference on Evolutionary Computing*, pages 289–294, Perth, Australia, November 1995.

[102] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7:308–313, 1965.

[103] I. Newton. *De analysi per aequationes numero terminorum infinitas*. 1669.

[104] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2000.

[105] Yew S. Ong, Prasanth B. Nair, and Andrew J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[106] Vilfredo Pareto. *Cours d'Economie Politique* . F. Rouge, Lausanne, 1896.

[107] Wei Peng and Qingfu Zhang. A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In *IEEE International Conference on Granular Computing, 2008. GrC 2008*, pages 534 –537, 2008.

[108] Michael J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7:155–162, 1964.

[109] Michael J. D. Powell. On search directions for minimization algorithms. *Mathematical Programming*, 4:193–201, 1973.

[110] M. K. Rahman. An intelligent moving object optimization algorithm for design problems with mixed variables, mixed constraints and multiple objectives. *Structural and Multidisciplinary Optimization*, 32(1):40–58, July 2006.

[111] Singiresu S. Rao. *Engineering Optimization*. John Wiley & Sons Inc., 3rd edition, 1996.

[112] A. Ravindran, K. M. Ragsdell, and G. V. Reklaitis. *Engineering Optimization: Methods and Applications*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.

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

[114] R. S. Rosenberg. *Simulation of genetic populations with biochemical properties*. PhD thesis, University of Michigan, Ann Arbor, Michigan, EE. UU., 1967.

[115] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[116] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, January 1994.

[117] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[118] J. David Schaffer and John J. Grefenstette. Multiobjective Learning via Genetic Algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 593–595, Los Angeles, California, 1985. AAAI.

[119] J. David Schaffer and Amy Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of*

the Second International Conference on Genetic Algorithms and their application, pages 36–40, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[120] Hans-Paul Schwefel. *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Stromungstechnik*. PhD thesis, Technical University of Berlin, 1965.

[121] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[122] Hamed Shah-Hosseini. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation*, 1:71–79, 2009.

[123] Karthik Sindhya, Kalyanmoy Deb, and Kaisa Miettinen. A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 815–824. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germ., September 2008.

[124] Helmut Sobieczky. Parametric Airfoils and Wings. In K. Fuji and G. S. Dulikravich, editors, *Notes on Numerical Fluid Mechanics, Vol.. 68*, pages 71–88, Wiesbaden, 1998. Vieweg Verlag.

[125] O. Soliman, L. T. Bui, and H. Abbass. A memetic coevolutionary multi-objective diffierential evolution algorithm. In C.-K. Goh, Y.-S. Ong, and K. C. Tan, editors, *Multi-Objective Memetic Algorithms*, pages 325–351. Springer, Studies in Computational Intelligence , Vol. 171, 2009.

[126] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential Application of Simplex Designs in Optimization and Evolutionary Operation. *Technometrics*, 4(4):441–461, November 1962.

[127] David A. Sprecher. A universal mapping for kolmogorov's superposition theorem. *Neural Netw.*, 6(8):1089–1094, January 1993.

[128] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[129] Rainer M. Storn and Kenneth V. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, Berkeley, CA, March 1995.

[130] B. Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering*, 28:1849–1871, 2004.

[131] András Szöllös, Miroslav Smíd, and Jaroslav Hájek. Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and epsilon-dominance. *Advances in Engineering Software*, 40(6):419–430, 2009.

[132] Virginia Joanne Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, Houston, Texas, USA, May 1989.

[133] Mohamed B. Trabia and Xiao Bin Lu. A Fuzzy Adaptive Simplex Search Optimization Algorithm. *Journal of Mechanical Design*, 123:216–225, 2001.

[134] Heike Trautmann, Uwe Ligges, Jörn Mehnen, and Mike Preuss. A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 825–836. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany, September 2008.

[135] Manuel Valenzuela-Rendón and Eduardo Uresti-Charre. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.

[136] Vladimir Vapnik, Steven E. Golowich, and Alex Smola. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press, 1997.

[137] Rémy Viennet, Christian Fontiex, and Ivan Marc. Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. *International Journal of Systems Science*, 27(2):255–260, 1996.

[138] Philippe Vincke. *Multicriteria Decision-Aid*. John Wiley & Sons, New York, 1992.

[139] L. Darrell Whitley, V. Scott Gordon, and Keith E. Mathias. Lamarckian Evolution, The Baldwin Effect and Function Optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 6–15, London, UK, 1994. Springer-Verlag.

[140] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[141] Yan yan Tana, Yong chang Jiaoa, Hong Lib, and Xin kuan Wanga. Moea/d + uniform design: A new version of moea/d for optimization problems with many objectives. *Computers & Operations Research*, 2012.

[142] Wen Ci Yu. The convergent property of the simplex evolutionary technique. *Scientia Sinica*, Zhongguo Kexue:69–77, 1979.

[143] Wen Ci Yu. Positive basis and a class of direct search techniques. *Scientia Sinica*, Zhongguo Kexue:53–68, 1979.

[144] W. I. Zangwill. Minimizing a Function Without Calculating Derivatives. *The Computer Journal*, 10(3):293–296, November 1967.

[145] Saúl Zapotecas Martínez, Alfredo Arias Montaño, and Carlos A. Coello Coello. A Nonlinear Simplex Search Approach for Multi-Objective Optimization. In *2011 IEEE Congress on Evolutionary*

Computation (CEC'2011), pages 2367–2374, New Orleans, USA, June 2011. IEEE Press.

[146] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Proposal to Hybridize Multi-Objective Evolutionary Algorithms with Non-Gradient Mathematical Programming Techniques. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, volume 5199, pages 837–846. Springer. Lecture Notes in Computer Science, Dortmund, Germany, September 2008.

[147] Saúl Zapotecas Martínez and Carlos A. Coello Coello. An Archiving Strategy Based on the Convex Hull of Individual Minima for MOEAs. In *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pages 912–919, Barcelona, España, July 2010. IEEE Press.

[148] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature–PPSN XI*, volume 6238, pages 576–585, Kraków, Poland, September 2010. Springer, Lecture Notes in Computer Science.

[149] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Multi-objective Particle Swarm Optimizer Based on Decomposition. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO'2011)*, pages 69–76, Dublin, Ireland, July 2011. ACM Press.

[150] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Direct Local Search Mechanism for Decomposition-based Multi-Objective Evolutionary Algorithms. In *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, pages 3431–3438, Brisbane, Australia, June 2012. IEEE Press.

[151] Saúl Zapotecas Martínez and Carlos A. Coello Coello. MOEA/D assisted by RBF Networks for Expensive Multi-Objective Optimization Problems. Technical Report EVOCINV-02-2013, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, México, February 2013.

[152] Saúl Zapotecas Martínez and Carlos A. Coello Coello. MONSS: A Multi-Objective Nonlinear Simplex Search Algorithm. Technical Report EVOCINV-01-2013, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, México, February 2013.

[153] Saúl Zapotecas Martínez, Edgar G. Yáñez Oropeza, and Carlos A. Coello Coello. Self-Adaptation Techniques Applied to Multi-Objective Evolutionary Algorithms. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683, pages 567–581, Rome, Italy, January 2011. Springer. Lecture Notes in Computer Science.

[154] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical Report CES-487, University of Essex and Nanyang Technological University, 2008.

[155] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.

[156] Qingfu Zhang, Wudong Liu, E. Tsang, and B. Virginas. Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model. *Evolutionary Computation, IEEE Transactions on*, 14(3):456 –474, june 2010.

[157] Xiang Zhong, Wenhui Fan, Jinbiao Lin, and Zuozhi Zhao. Hybrid non-dominated sorting differential evolutionary algorithm with nelder-mead. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 1, pages 306 –311, December 2010.

[158] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[159] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In

K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001.

[160] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.

[161] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Noviembre 1999.

[162] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.