Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Hiper-Heurísticas Paralelas para Optimización Multi-Objetivo

Tesis que presenta

Raquel Hernández Gómez

Para obtener el Grado de

Doctora en Ciencias en Computación

Director de la Tesis

Dr. Carlos Artemio Coello Coello

México, Ciudad de México Marzo, 2018

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Zacatenco Campus

Computer Science Department

Parallel Hyper-Heuristics for Multi-Objective Optimization

Submitted by

Raquel Hernández Gómez

as the fulfillment of the requirement for the degree of

Ph.D. in Computer Science

Advisor

Dr. Carlos Artemio Coello Coello

Mexico City March, 2018

Resumen

Una amplia variedad de problemas complejos del mundo real son de naturaleza multi-objetivo y su solución implica encontrar un conjunto de variables de decisión que representen los mejores compromisos entre todos sus objetivos. Los Algoritmos Evolutivos Multi-Objetivo (AEMOs) son técnicas poderosas de búsqueda, inspiradas en ideas derivadas de procesos biológicos, que son adecuadas para encontrar soluciones casi óptimas para este tipo de problemas. Usualmente, los AEMOs dirigen su búsqueda por medio de una única heurística, que puede ser una regla, componente o proceso. Un fenómeno común es que los AEMOs son exitosos en la resolución de un tipo particular de problema. Sin embargo, al aplicarlos a nuevos problemas o instancias ligeramente diferentes, los AEMOs pueden ver degradado su desempeño. Por esta razón, una nueva tendencia es que los AEMOs seleccionen la heurística más apropiada de un conjunto de heurísticas disponibles. Este paradigma se conoce como hiper-heurística y se ha promovido con el objetivo de proporcionar metodologías de búsqueda de aplicación general. Sin embargo, al usar hiper-heurísticas surge la problemática de que éstas pueden requerir mucho tiempo de ejecución para ciertas aplicaciones y por lo tanto surge la necesidad de utilizar paralelismo. En esta tesis, proponemos diferentes hiper-heurísticas paralelas multi-objectivo, así como un marco de software para su desarrollo. Los enfoques propuestos se comparan con respecto a los AEMOs de última generación que adoptan varios problemas de referencia y medidas de rendimiento que normalmente se utilizan en la literatura especializada. Nuestros resultados experimentales indican que nuestras propuestas obtienen mejores resultados en la mayoría de los casos, siendo aplicables para una amplia gama de problemas complicados y también para problemas que tienen cuatro o más objetivos (es decir, los denominados problemas de optimización con muchos objetivos).

Abstract

A wide variety of complex real-world problems are multi-objective in nature, and their solution involves finding a set of decision variables that represent the best possible trade-offs among all their objectives. Multi-Objective Evolutionary Algorithms (MOEAs) are powerful search techniques, which are inspired by ideas derived from biological science, being suitable for finding near-optimal solutions for such types of problems. Usually, MOEAs rely on the guidance of a single heuristic, which can be a rule, component or process. A common phenomenon is that MOEAs are successful in solving a particular kind of problem. However, when applying them to new problems or slightly modified instances, MOEAs may face difficulties in their performance. For this reason, a new trend is that MOEAs select the most appropriate heuristic from a pool of available heuristics. This paradigm is known as hyper-heuristic and has been promoted with the aim to provide generally applicable search methodologies. Another issue is that using hyper-heuristics can be a time-consuming task for certain applications and therefore the need to use parallelism. In this thesis, we propose different parallel hyper-heuristics for multi-objective optimization, and a software framework for their development. The proposed approaches are compared with respect to state-of-the-art MOEAs adopting several benchmark problems and performance measures normally used in the specialized literature. Our experimental results indicate that our proposals obtain better results in most cases, being applicable for a wide range of complicated problems, and also for problems having four or more objectives (i.e., the so-called many-objective optimization problems).

Agradecimientos

En primer lugar deseo expresar mis más sinceros y reiterados agradecimientos al Dr. Carlos A. Coello Coello, por el apoyo, esmero y paciencia otorgados.

Agradezco a los sinodales, Dr. Luis Gerardo de La Fraga, Dr. Amilcar Meneses Viveros, Dr. Gregorio Toscano Pulido y al Dr. Antonin Sebastien Ponsich por sus enriquecedores comentarios.

Agradezco a las doctoras Ana María A. Martínez Enríquez, Sonia G. Mendoza Chapa y Dolores Lara Cuevas; así como a los doctores Guillermo B. Morales Luna y José G. Rodríguez García por sus enseñanzas y atinados consejos.

Infinitas gracias al personal del departamento: Sofía Reza Cruz, Felipa Rosas López y Erika B. Ríos Hernández, Arcadio A. Morales Vázquez, José U. Cruz Cedillo, Santiago Domínguez Domínguez y José Luis Flores Garcilazo. Sin su apoyo este trabajo de tesis no se hubiera concretado.

Agradezco a mis colegas: Miriam Pescador Rojas, Adriana Menchaca Mendez, Jesús G. Falcón Cardona y Saúl Zapotecas Martínez por su solidaridad.

Muchas gracias a mi familia por su apoyo incondicional.

Agradezco también al CONACyT y CINVESTAV, por la beca otorgada durante este tiempo. Sin su financiamiento muchos estudiantes nos quedarıamos a la mitad del camino.

Este trabajo de tesis se derivó del proyecto CONACyT titulado "Nuevos Paradigmas Algorítmicos en Optimización Evolutiva Multi-Objetivo" (Ref. 221551), cuyo Responsable Técnico es el Dr. Carlos A. Coello Coello.

Se agradece el apoyo a la Coordinación General de Servicios de Tecnologías de la Información y Comunicaciones (CGSTIC) del CINVESTAV por proveer recursos de cómputo de alto desempeño con el Cluster Hibrído "Xiuhcoatl"[1] del nodo LANCAD CINVESTAV.

Y sobre todo se agradece a ABACUS-CINVESTAV por facilitar recursos de supercómputo con el proyecto CONACYT EDOMEX-2011-C01-165873.

[1] `http://clusterhibrido.cinvestav.mx`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most real-world problems are multi-objective in nature, requiring the simultaneous optimization of several (often conflicting) objective functions, whose solution involves finding a set of decision variables that represent the best possible trade-offs among all their objectives. This set of decision variables is called Pareto optimal set, and their image is named the Pareto optimal front. Multi-Objective Evolutionary Algorithms (MOEAs), as well as other bio-inspired meta-heuristics, are powerful search techniques that are suitable to solve multi-objective optimization problems. They can find discrete approximations to the Pareto optimal set in a single run without requiring particular assumptions, such as continuity or differentiability. In fact, MOEAs perform random search strategies that operate under Darwin's principle of natural selection, where the fittest individuals must achieve: 1) convergence to the Pareto optimal front and 2) uniform distribution along the objective space (better known as diversity). In the last few decades, several MOEAs have been proposed, with the vast majority relying on two concepts: Pareto dominance as their primary selection mechanism, followed by a density estimator. The former favors non-inferior or non-dominated solutions over dominated ones, whereas the latter induces a total order of incomparable solutions, preserving diversity at the same time.

1.1 Motivation

One of the main concerns is that Pareto-based MOEAs face difficulties to reach the Pareto optimal front when dealing with multi-objective optimization problems with four or more objectives (also known as many-objective optimization problems) [74, 131, 92]. This is due to the fact that most or all solutions in the population quickly become non-dominated with respect to the rest, and the best individuals are identified only by the density estimator. Thus, in some cases, good locally non-dominated solutions in terms of convergence might be discarded at the expense of keeping good solutions in terms of diversity, in spite of the fact that they may be distant from the Pareto optimal front [1]. To address this issue, a new trend is the incorporation of performance indicators into the selection mechanism of a MOEA [7, 34, 154]. The

hypervolume indicator [152] is, with no doubt, a natural choice, (see for example [34, 154]) since it is the only unary indicator that is known to be Pareto compliant. Also, it has been shown that maximizing the hypervolume is equivalent to reaching the Pareto optimal set [35]. However, the main drawback of this sort of approach is its computational cost, which increases exponentially with the number of objectives [13], making it prohibitive for many-objective optimization problems. Moreover, another issue is that MOEAs require a significant number of function evaluations. Thus, their applicability becomes unaffordable for certain applications that demand an intensive use of CPU or memory.

The parallelization of MOEAs (pMOEAs) arises as an attractive option to address these factors, where the basic idea is to divide somehow the MOEA into several tasks. Each of these tasks is solved simultaneously on a different processing unit and, once all of them are completed, the results are combined to provide a solution to the problem [3, p.2]. Moreover, processing units can be in the same machine, or distributed in a collection of machines interconnected by a network [22]. The wide acceptance of pMOEAs is mainly because they can produce substantial gains in performance, and in some cases, they can also improve the accuracy of the results with respect to their serial counterparts. Moreover, when MOEAs combine different low-level heuristics, the quality of solutions is highly improved. In general, these methods are known as hyper-heuristics, and can be seen as high-level methodologies, which automatically produce an adequate combination of single heuristics for solving a broad set of problems. In this thesis, we attempt to propose new algorithms based on hyper-heuristics and pMOEAs.

1.2 Hypotheses

In this work, we consider the two following hypotheses:

1. MOEAs combining different search techniques perform much better than algorithms based on a single search strategy.

2. Parallel MOEAs can reduce the execution time of their serial counterparts, with the possibility of also improving the quality of solutions.

1.3 Contributions

Next, we describe the main contributions of this thesis.

- In collaboration with Miriam Pescador and a research group from the Federico Santa María Technical University in Chile, we performed a review of scalarizing functions, which transform the original multi-objective optimization problem into several single-objective problems with the aid of different target directions. Moreover, we performed a novel experimental study for determining the behavior of such scalarizing functions in many-objective optimization problems [106].

- We develop, for the first time, a Hyper-Heuristic of scalarizing functions, called Many-Objective Metaheuristic Based on the $R2$ Indicator III (MOMBI-III), for solving continuous multi-objective optimization problems (see Chapter 3 in page 39). The pool of heuristics consists of seven scalarizing functions, which are compatible with some form of Pareto dominance. The basic idea is that MOMBI-III grants more presence to those scalarizing functions that help to improve diversity, which is measured using the s-energy indicator. Experimental results showed that MOMBI-III significantly outperformed single heuristics as well as other state-of-the-art algorithms on standard test problems. Furthermore, our proposal showed promise in solving inverted Pareto fronts, as well as many-objective optimization problems.

- We provide a counterexample showing that for a certain parameter model ($\theta = 5$), the scalarizing function, Penalty Boundary Intersection (PBI), may be incompatible with any form of the Pareto dominance relation (see Figure 3.3 in page 42). This is an important result because PBI has been widely used within the search mechanism of optimizers based on decomposition, and this finding may help us to understand its poor behavior in disconnected and degenerated problems.

- We present a novel algorithm, named Multi-objective Optimizer based on Value Path (MOVAP), which incorporates a density estimator based on a visualization technique called Parallel Coordinates (see Chapter 4 in page 61). This technique has frequently been used for visualizing results in multi-objective optimization, specifically in high-dimensional spaces. To the best of our knowledge, MOVAP is the first algorithm that incorporates automatic image analysis in its search mechanism in order to extract knowledge from the Parallel-Coordinates plot. Experimental results indicate that MOVAP significantly outperforms state-of-the-art algorithms in more than 35% of the test instances, producing a much better diversity of solutions, and exploring more regions of the search space in high dimensional spaces than the MOEAs with respect to which it was compared. Moreover, this density estimator is extended to an archiving technique for an island-based MOEA (see Chapter 5 in page 75).

- In Chapter 5 of page 75, we proposed, in collaboration with Dr. Enrique Alba from Málaga University, a parallel island version of the $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA). Our proposal, called PArallel MICRo Optimizer based on the $\mathcal{S}$ metric ($\mathcal{S}$-PAMICRO), is a steady state genetic algorithm that ranks individuals according to Pareto dominance and uses the hypervolume as its density estimator. In this proposal, we observed that regardless of the number of objectives, CPU time was almost negligible when using small populations. Therefore, $\mathcal{S}$-PAMICRO splits the overall population into subpopulations (islands), containing less than 12 individuals. Each island evolves independently a serial SMS-EMOA with an external archive that is pruned using the density estimator of MOVAP. We observed that $\mathcal{S}$-PAMICRO

could achieve much better results than SMS-EMOA and HypE (another MOEA based on the hypervolume), spending much less computational time.

- In Chapter 6 of page 89, we developed a free open-source software framework, named EMO Project, intended to solve multi-objective optimization problems. This software consists of a set of useful command-line programs, which are implemented in ANSI C, GNU Make,[1] MPI[2] and Gnuplot.[3] MPI is used for the parallelization of MOEAs and concurrent execution of commands over several processors, whereas Gnuplot is used for visualization purposes. Moreover, EMO Project provides a set of predefined libraries for the developing of new algorithms.

- In Appendix A of page 141, we presented a corrected version of the incremental hypervolume algorithm of the Walking Fish Group. This algorithm was adopted in SMS-EMOA and $\mathcal{S}$-PAMICRO to efficiently determine the solution that contributes the least to the hypervolume of a non-dominated set.

1.4 Thesis Structure

Including this introduction, the thesis consists of 8 chapters and two appendices.

Chapter 2 provides some basic concepts and definitions about multi-objective optimization and evolutionary algorithms, which are required as a background for the following chapters of this document. This chapter also reviews the related work.

Chapter 3 presents a novel hyper-heuristic of scalarizing functions. We first provide the motivation, discussing some limitations and the need for combining different heuristics. Then, we introduce our proposal, which is an extension of a genetic algorithm that ranks the population using the $R2$ indicator. After that, the computational complexity is derived. Next, we present the experimental results using standard test problems. We divide the experiments in single heuristics, state-of-the-art algorithms, inverted test instances, and many-objective problems.

Chapter 4 is devoted to the description of a density estimator based on the Parallel-Coordinates graph. We first provide the motivation, making some observations about the knowledge that can be extracted from these graphs. Then, we present the algorithm of the density estimator and couple it to a genetic algorithm. Its computational complexity is also provided. Then, the experiments and discussion of the results are presented, comparing our proposal with other popular MOEAs on standard test problems. The proposed density estimator can be considered as a hyper-heuristic of a set of rules derived from empirical knowledge.

Chapter 5 introduces a parallel version of SMS-EMOA, which draws some ideas from the island model. First, we provide the motivation, noticing that the calculation

[1] https://www.gnu.org/software/make
[2] https://www.mpich.org
[3] http://www.gnuplot.info

of the hypervolume in many-objective problems is achievable when using relative small populations. Then, we present our proposal, incorporating the density estimator of the previous chapter. In the experimental results, we include an analysis of the effect of the migration parameters using the hypervolume and IGD^+ indicators. We also perform tests with hypervolume-based MOEAs on artificial problems. A discussion of the results is provided.

Chapter 6 focuses on the description of our software framework EMO Project[4], which supports parallelization of MOEAs over TCP/IP. We present its architecture, command-line programs, and libraries. In addition, we illustrate how to implement new problems and algorithms.

Chapter 7 evidences the applicability of MOMBI-III in solving real-world problems. Since their problem definition involves constraints, first we describe the adopted technique to handle them. Then, we validate our approach on three well-known constrained problems. Finally, we investigate the performance of our hyper-heuristic on three engineering design optimization problems having up to eight objectives and thirty constraints. In all tests, we compare our approach with state-of-the-art algorithms using the ONVG, hypervolume, IGD^+ and two-set coverage indicators.

The conclusion of the thesis, as well as some possible paths for future research are presented in Chapter 8.

Appendix A describes the incremental hypervolume algorithm of the Walking Fish Group, which is used to determine the solution that contributes the least to the hypervolume of a non-dominated set. We describe a bug in its original implementation, and provide a solution to fix it keeping a low computational cost.

Finally, Appendix B presents the standard test problems used in the comparative studies.

1.5 Publications

The following book chapter, conference, workshop papers and journals were produced during the thesis preparation:

- Raquel Hernández Gómez, and Carlos A. Coello Coello. Improved Metaheuristic Based on the $R2$ Indicator for Many-Objective Optimization. In *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*, pages 679-686, Madrid, Spain, July 11-15 2015. ACM Press. ISBN 978-1-4503-3472-3.

  (Derived from Chapter 3)

- Raquel Hernández Gómez, Carlos A. Coello Coello, and Enrique Alba Torres. A Multi-Objective Evolutionary Algorithm based on Parallel Coordinates. In *2016 Genetic and Evolutionary Computation Conference (GECCO'2016)*, pages 565-572, Denver, Colorado, USA, 20-24 July 2016. ACM Press. ISBN 978-1-4503-4206-3. **(Best Paper Award Track EMO)**

[4] Available for download at the link http://computacion.cs.cinvestav.mx/ rhernandez

(Derived from Chapter 4)

- Raquel Hernández Gómez, and Carlos A. Coello Coello. Parallel SMS-EMOA for Many-Objective Optimization Problems. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, pages 1011–1014, New York, NY, USA, 2016. ACM. **(Student Workshop Best Paper)**

  (Derived from Chapter 5)

- Raquel Hernández Gómez, Carlos A. Coello Coello, and Enrique Alba. A Parallel Version of SMS-EMOA for Many-Objective Optimization Problems. In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference*, pages 568-577. Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, September 17-21 2016. ISBN 978-3-319-45822-9.

  (Derived from Chapter 5)

- Miriam Pescador-Rojas, Raquel Hernández Gómez, Elizabeth Montero, Nicolás Rojas-Morales, María-Cristina Riff, and Carlos A. Coello Coello. An Overview of Weighted and Unconstrained Scalarizing Functions. In Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme, editors, *Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*, pages 499–513. Springer International Publishing, Cham 2017.

  (Derived from Chapter 3)

- Raquel Hernández Gómez, and Carlos A. Coello Coello. A Hyper-Heuristic of Scalarizing Functions. In *2017 Genetic and Evolutionary Computation Conference (GECCO'2017)*, pages 577-584, Berlin, Germany, July 15-19, 2017. ACM Press. ISBN 978-1-4503-4920-8.

  (Derived from Chapter 3)

- Carlos A. Coello Coello, Luis Miguel Antonio, and Raquel Hernández Gómez. Fundamentals and Practice of Evolutionary Optimization. John Wiley & Sons, Inc., 2018 (submitted).

  (Derived from Chapter 2)

- Raquel Hernández Gómez, Carlos A. Coello Coello, and Enrique Alba. A Parallel Island Model for Hypervolume-Based Many-Objective Optimization. In Thomas Bartz-Beielsten, Bogdan Filipic, Peter Korosec, El-Ghazali Talbi, editors, High-Performance Simulation Based Optimization. Springer. Studies in Computational Intelligence 2018 (submitted).

(Derived from Chapter 5)

- Raquel Hernández Gómez, and Carlos A. Coello Coello. Considerations in the Incremental Hypervolume Algorithm of the Walking Fish Group. Soft Computing, Springer 2018 (submitted).

  (Derived from Appendix A)

- Luis G. De La Fraga, Esteban Tlelo-Cuautle, Raquel Hernández Gómez, and Carlos A. Coello Coello. Many-objective Optimization of an Analog Circuit. Swarm and Evolutionary Computation (to be submitted).

  (Derived from Chapter 7)

- Raquel Hernández Gómez, and Carlos A. Coello Coello. A Survey of Parallel Multi-Objective Evolutionary Algorithms. IEEE Transactions on Cybernetics (to be submitted).

  (Derived from Chapter 2)

- Elizabeth Montero, Nicolás Rojas Morales, María Cristina Riff, Miriam Pescador Rojas, Raquel Hernández Gómez, and Carlos A. Coello Coello. On the Effects of Scalarizing Functions. IEEE Transactions on Cybernetics (to be submitted).

Chapter 2

Background

This chapter provides some fundamental concepts and formal notation that are adopted throughout this document. Section 2.1 covers definitions of multi-objective optimization, such as optimality notions, reference points, pursued goals, performance indicators, and scalarizing functions. Section 2.2 includes an introduction to multi-objective evolutionary algorithms, which includes their main components, their selection strategies, the most representative algorithms, and their parallel models. We also differentiate three commonly confused words: heuristic, meta-heuristic, and hyper-heuristic. Section 2.3 discusses the most relevant related work. We close this chapter with a summary in Section 2.4.

2.1 Multi-Objective Optimization

Multi-objective optimization deals with solving mathematical problems involving the simultaneous optimization of two or more competing objective functions, instead of having only one. Consequently, there is no single solution, but several of them, which represent the best possible trade-offs among the objectives. Thus, it is rarely the case that there is a single point that simultaneously optimizes all the objectives.[1]

2.1.1 Problem definition

A Multi-Objective Optimization Problem (MOP) is defined as follows:

$$\begin{aligned}
\text{Minimize} \quad & \mathbf{f}(\mathbf{x}) := \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\big) && && (2.1)\\
\text{subject to} \quad & g_i(\mathbf{x}) \geq 0 && \forall i \in \{1, 2, \ldots, o\}, && (2.2)\\
& h_j(\mathbf{x}) = 0 && \forall j \in \{1, 2, \ldots, r\}, && (2.3)\\
& x_k \in [x_k^l, x_k^u] && \forall k \in \{1, 2, \ldots, n\}. && (2.4)
\end{aligned}$$

$\mathcal{X} \subset \mathbb{R}^n$ is the decision space (or search space) and $\mathcal{Z} \subset \mathbb{R}^m$ is the objective space. Each decision vector $\mathbf{x} \in \mathcal{X}$ is related to an objective vector $\mathbf{f}$, where $\mathcal{X}$ is restricted

[1] This would be possible only if there was no conflict among the objectives.

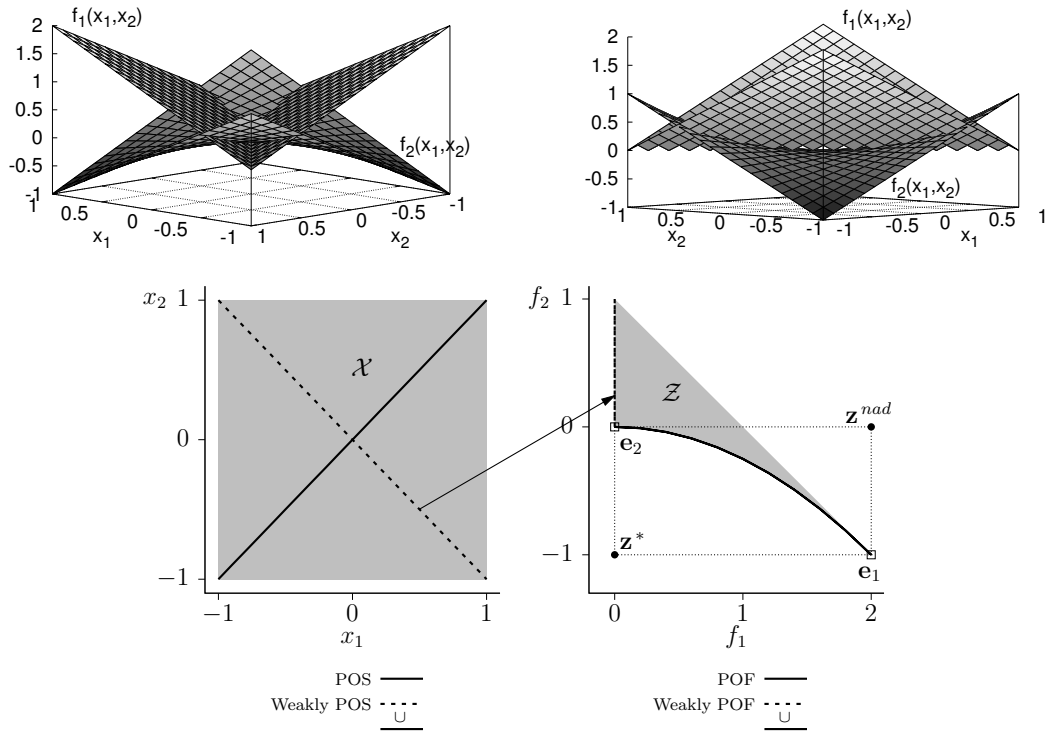


Figure 2.1: Illustration of a minimization problem $\mathbb{R}^2 \to \mathbb{R}^2$. On top, we show different views of the surfaces. The variable space is on the bottom left where $x_1, x_2 \in [-1, 1]$. The objective space is on the bottom right, defined by the functions $f_1(x_1, x_2) = |x_1 + x_2|$ and $f_2(x_1, x_2) = -x_1 x_2$.

by inequality constraints (2.2), equality constraints (2.3), and bounds on the decision variables (2.4). An example of a MOP is provided in Figure 2.1.

MOPs are classified and studied according to their features. For instance, constrained MOPs [100] include problems with inequality and equality constraints; large-scale MOPs [19] have hundreds, even thousands of decision variables; whereas Many-objective Optimization Problems are those having four or more objectives. Figure 2.2 presents this classification. In this thesis, we mainly focus on continuous nonlinear MOPs having up to 10 objectives with box constraints. Nevertheless in Chapter 7, we tackle some real-world applications having equality and inequality constraints.

2.1.2 Optimality notion

In multi-objective optimization, it is not possible to compare directly two solutions $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ as in single-objective optimization. As an alternative, the *Pareto dominance relation* must be applied:

> A solution $\mathbf{x}$ is said to *dominate* a solution $\mathbf{y}$, denoted by $\mathbf{x} \prec \mathbf{y}$ or $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{y})$, if and only if $\mathbf{x}$ is at least as good as $\mathbf{y}$ in all objectives ($\forall i \in \{1, \ldots, k\}$ $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$) and better in at least one objective ($\exists j \in \{1, \ldots, k\}$, $f_j(\mathbf{x}) < f_j(\mathbf{y})$).

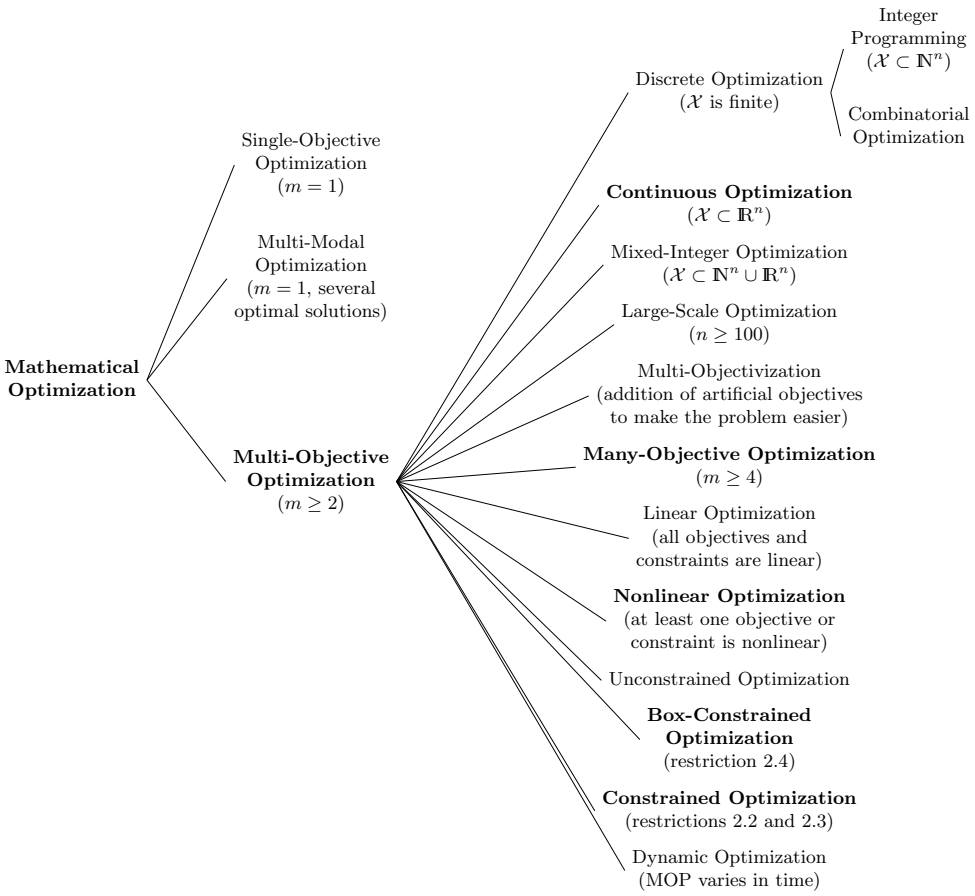


Figure 2.2: Location of multi-objective optimization and its major subfields.

This binary relation induces a *strict partial order*[2] on $\mathcal{X}$, where three scenarios may occur, either $\mathbf{x} \prec \mathbf{y}$, $\mathbf{y} \prec \mathbf{x}$ or neither of both, i.e., they are *incomparable*. The collection of all incomparable solutions relative to a set $A \subseteq \mathcal{X}$ is known as the Non-Dominated Set (NDS), given by:

$$\text{NDS}(A) := \{\mathbf{a} \in A : \nexists \mathbf{a}' \in A, \mathbf{a}' \prec \mathbf{a}\}. \tag{2.5}$$

Therefore, the solution to a MOP consists of finding the optimal set of non-dominated decision vectors in all $\mathcal{X}$, which cannot be improved in any objective without worsening at least another objective. This set is known as the *Pareto Optimal Set* (POS), and is formally defined as:

$$\text{POS} := \{\mathbf{x}^* \in \mathcal{X} : \nexists \mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n, \mathbf{x} \prec \mathbf{x}^*\}, \tag{2.6}$$

and its image is named the *Pareto Optimal Front* (POF):

$$\text{POF} := \{\mathbf{f}(\mathbf{x}^*) \in \mathcal{Z} \subset \mathbb{R}^m : \mathbf{x}^* \in \text{POS}\}. \tag{2.7}$$

[2] A strict partial order is a binary relation that is irreflexive ($\mathbf{a} \not\prec \mathbf{a}$), transitive (if $\mathbf{a} \prec \mathbf{b}$ and $\mathbf{b} \prec \mathbf{c}$ then $\mathbf{a} \prec \mathbf{c}$) and asymmetric (if $\mathbf{a} \prec \mathbf{b}$ then $\mathbf{b} \not\prec \mathbf{a}$).

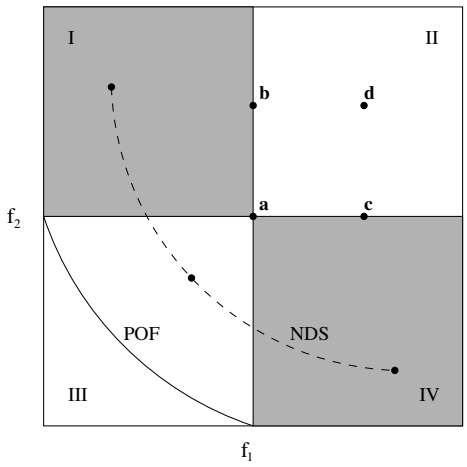


Figure 2.3: Examples of Pareto dominance relations. It holds that $\mathbf{a} \preceq \mathbf{a}$, $\mathbf{a} \preceq \mathbf{b}$, $\mathbf{a} \preceq \mathbf{c}$, $\mathbf{a} \preceq \mathbf{d}$, $\mathbf{a} \prec \mathbf{b}$, $\mathbf{a} \prec \mathbf{c}$, and $\mathbf{a} \prec \mathbf{d}$. Solutions that are inside the quadrants I and IV are incomparable to each other.

Another commonly way to compare solutions is by using the *weak Pareto dominance relation*:

> A solution $\mathbf{x}$ is said to *weakly dominate* a solution $\mathbf{y}$, denoted by $\mathbf{x} \preceq \mathbf{y}$ or $\mathbf{f}(\mathbf{x}) \preceq \mathbf{f}(\mathbf{y})$, if and only if $\mathbf{x}$ is at least as good as $\mathbf{y}$ in all objectives ($\forall i \in \{1, \ldots, k\}$. $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$).

This relaxed form of the Pareto dominance induces a *partial order*[3] on $\mathcal{X}$, where the following optimality notion is derived:

> A vector of decision variables $\mathbf{x}^* \in \mathcal{X}$ is *weakly Pareto optimal* if there does not exist another vector $\mathbf{x} \in \mathcal{X}$ such that $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, m$.

The set of all weakly Pareto optimal solutions in $\mathcal{X}$ constitutes the *weakly Pareto optimal set*, which is the superset of the Pareto optimal set.

For our example in Figure 2.1, the POS is $\{(x_1, x_2) | x_1 = x_2\}$ and the weakly POS is $(x_1, x_2) | x_1 = x_2, x_1 = -x_2$. Moreover, Figure 2.3 shows the difference between the two Pareto dominance relations.

2.1.3 Reference points

The Pareto optimal front of a MOP is bounded by two special points: the ideal and nadir points. Next, we provide their definitions.

> The *ideal point* $\mathbf{z}^* \in \mathbb{R}^m$ minimizes all the objective functions, being its i^{th}-component $z_i^* := \min \{ f_i(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X} \}$ $\forall i \in \{1, \ldots, m\}$.

[3] A partial order is a binary relation that is reflexive ($\mathbf{a} \prec \mathbf{a}$), antisymmetric (if $\mathbf{a} \prec \mathbf{b}$ and $\mathbf{b} \prec \mathbf{a}$ then $\mathbf{a} = \mathbf{b}$) and transitive (if $\mathbf{a} \prec \mathbf{b}$ and $\mathbf{b} \prec \mathbf{c}$ then $\mathbf{a} \prec \mathbf{c}$) .

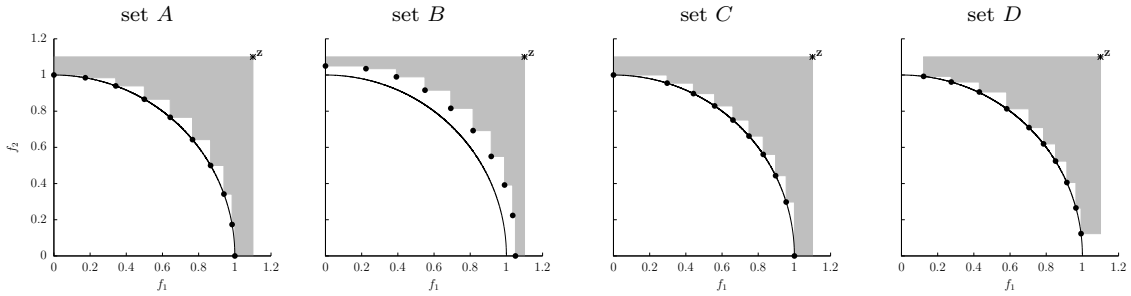


Figure 2.4: Desired features of approximation sets to the Pareto optimal front. From left to right the following goals are fulfilled: G1-3, G2,3, G1,3, and G1,2.

> The *nadir point* $\mathbf{z}^{nad} \in \mathbb{R}^m$ is constructed using the worst values of the Pareto optimal front. Its i^{th}-component is defined as $z_i^{nad} := \max\{f_i(\mathbf{x}) \mid \mathbf{x} \in \text{POS}\}$ $\forall i \in \{1, \ldots, m\}$.

In most multi-objective evolutionary algorithms (see Subsection 2.2), the approximation to the ideal point considers all individuals created so far. In contrast, the nadir point is updated using only information of the current Pareto front.

Another important concept are the *extreme points*, which is the set of points $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$ living in $\mathbb{R}^m$ that yield the best value of one objective. Extreme points are characterized by being enclosed by the nadir point.

These reference points appear in the bottom-right side of Figure 2.1 on page 10.

2.1.4 Pursued goals

Computing the whole Pareto optimal set of a given MOP in an enumerative manner is not possible in most cases, since the cardinality of this set might be huge, or even infinite. Even if its storage is achievable on a computer with limited resources, the decision maker might be unable to check all the possible solutions that can be generated. For this reason, a discrete approximation that better represents the optimal set is a much more practical choice. Such approximation must fulfill the following goals:

G1. Convergence to the Pareto optimal front,

G2. A uniform distribution of solutions along the Pareto front, and

G3. A good spread of solutions, such that all of the Pareto front is covered.

The last two points are closely related, and are known as *diversity*. These goals are depicted in Figure 2.4.

2.1.5 Performance indicators

A *performance indicator* evaluates the quality of an approximation to the Pareto optimal front, regarding convergence or diversity, formally defined as follows [157]:

A k-ary *performance indicator* I is a function $I : \mathcal{Z}^k \to \mathbb{R}$, which assigns each vector $(A_1, A_2, \ldots, A_k)$ of k approximation sets a real value $I(A_1, \ldots, A_k)$.

Of special interest are the *unary performance indicators*, i.e. when $k = 1$. Comparisons between approximation sets $A, B \subset \mathcal{Z}$ are possible using the weak Pareto dominance relation, where A weakly dominates B if it occurs:

$$A \preceq B \Leftrightarrow \forall \mathbf{b} \in B, \exists \mathbf{a} \in A : \mathbf{a} \preceq \mathbf{b}. \tag{2.8}$$

Using this binary relation, a unary performance indicator I is said to be *weakly Pareto compliant*, if it fulfills:

$$\forall A, B \subset \mathcal{Z} : A \preceq B \Rightarrow I(A) \geq I(B), \tag{2.9}$$

and is *Pareto compliant*, if it satisfies:

$$\forall A, B \subset Z : (A \preceq B \wedge A \neq B) \Rightarrow I(A) > I(B). \tag{2.10}$$

Performance indicators have been used mainly to compare the effectiveness of optimizers, and recently, an important trend is their incorporation into the search engine of optimizers. In this section, we briefly review the most important performance indicators reported in the specialized literature. Unless otherwise stated, let $A, B \subset \mathcal{Z}$ be the approximations to be evaluated and $R \subset \mathcal{Z}$ be a reference set.

Hypervolume

The only unary performance indicator that is known to be Pareto compliant is the *hypervolume* [152], also known as the $\mathcal{S}$ metric. This indicator determines the size of the portion of objective space that is dominated by the solutions of a set A, collectively and bounded by a reference point $\mathbf{z} \in \mathbb{R}^m$:

$$HV(A; \mathbf{z}) = \Lambda \left(\bigcup_{\mathbf{a} \in A} \{\mathbf{x} \mid \mathbf{a} \prec \mathbf{x} \prec \mathbf{z}\} \right), \tag{2.11}$$

where Λ denotes the Lebesgue measure and $\mathbf{z}$ should be dominated by all elements on A. Higher values of this indicator are preferred. The hypervolume indicator considers all the pursued goals, being the order of importance G1, G3 and G2. Therefore, it has some bias for favoring non-linear Pareto fronts with clusters near the middle point (*knee region*). A nice mathematical property of the hypervolume is that its maximization is equivalent to reaching the Pareto optimal set [35]. This has been experimentally validated [80, 34]. For these reasons, the hypervolume is one of the most preferred performance indicators for comparing multi-objective optimizers. Nevertheless, the main drawback of using the hypervolume is its high computational cost, which grows exponentially with the number of objectives [143]. The shaded area in Figure 2.4 (page 13) corresponds to the hypervolume indicator using the reference point $\mathbf{z} = (1.1, 1.1)$.

$R2$ Indicator

Given a set U of utility functions, which have an associated set of weight vectors W, the unary version of the $R2$ indicator [14] is given by:

$$R2(A;U) = -\frac{1}{|U|}\sum_{u\in U}\max_{\mathbf{a}\in A}\{u(\mathbf{a};\mathbf{w})\}, \tag{2.12}$$

This indicator simultaneously evaluates all the desired aspects of a Pareto front approximation, where lower values are preferred. Moreover, the computational cost of the $R2$ indicator is much lower than that of the the hypervolume, being weakly Pareto compliant. Regarding the choice of the utility functions u, there are several possibilities (see Table 2.5 on page 19), such as: WS, CHE, ASF, PBI, etc. The main drawbacks of this indicator are that the weight vectors should be supplied by the user, and since we are adding values of different utility functions, the set A should be normalized.

Generational Distance

The Generational Distance (GD) [116], indicates how "far" the approximation set A is from the discretized Pareto optimal front R. In other words, it measures the average distance from each $\mathbf{a} \in A$ to its closest reference point $\mathbf{r} \in R$, and is given by:

$$GD(A;R) = \left(\frac{1}{|A|}\sum_{\mathbf{a}\in A} d(\mathbf{a},R)^p\right)^{1/p}. \tag{2.13}$$

where $p \in \mathbb{R}$ (usually set to $p = 2$) and d is the Euclidean distance from $\mathbf{a} \in A$ to its nearest member of R:

$$d(\mathbf{a},R) = \min_{\mathbf{r}\in R}\sqrt{\sum_{i=1}^{m}(a_i - r_i)^2}. \tag{2.14}$$

A result of $GD(A,R) = 0$ indicates that $A = R$; any other value indicates a deviation. This performance indicator assesses convergence. However, it is not Pareto compliant.

Inverted Generational Distance

The Inverted Generational Distance (IGD) [116] indicates how "far" the discretized Pareto optimal front R is from the approximation set A, i.e., it is the average distance from each reference point to its nearest solution in A. The formal definition of IGD is the following:

$$IGD(A;R) = \left(\frac{1}{|R|}\sum_{\mathbf{r}\in R}\tilde{d}(\mathbf{r},A)^p\right)^{1/p}, \tag{2.15}$$

where $p \in \mathbb{R}$ (usually set to $p = 2$) and $\tilde{d}$ is the Euclidean distance from $\mathbf{r} \in R$ to its nearest member of A:

$$\tilde{d}(\mathbf{r}, A) = \min_{\mathbf{a} \in A} \sqrt{\sum_{i=1}^{m} (r_i - a_i)^2}. \tag{2.16}$$

Similarly to the GD indicator, the interpretation of the IGD values is as follows: when $IGD(A, R) = 0$, then $A = R$; any other value $IGD(A, R) > 0$ represents a deviation. This performance indicator measures both, convergence and diversity. Unfortunately, it is not Pareto compliant.

Δ_p Indicator

Δ_p [116] can be seen as an "averaged Hausdorff distance" between the approximation set and the discretized Pareto optimal front, defined as:

$$\Delta_p(A; R) = \max(GD(A, R), IGD(A, R)) \tag{2.17}$$

where GD is the generational distance (2.13) and IGD is the inverted generational distance (2.15). Δ_p simultaneously evaluates proximity to the Pareto optimal front and spread of solutions along it. Although Δ_p is not Pareto compliant, its computation has a much lower computational cost than that of the hypervolume.

IGD$^+$

The modified Inverted Generational Distance (IGD$^+$) [69] is an improved version of IGD ($p = 1$), now being weakly Pareto compliant (note that the inequality in expression (2.9) on page 14 should be reversed). Its definition is given by:

$$IGD^+(A; R) = \frac{1}{|R|} \sum_{\mathbf{r} \in R} d^+(\mathbf{r}, A). \tag{2.18}$$

The underlying idea of this performance indicator is that the distance d^+ considers the Pareto dominance relation:

$$d^+(\mathbf{r}, A) = \min_{\mathbf{a} \in A} \sqrt{\sum_{i=1}^{m} \left(\max\{a_i - r_i, 0\}\right)^2}. \tag{2.19}$$

This indicator is to be minimized, having an optimum value of zero. IGD$^+$ may achieve all the pursued goals, without having a specific order of importance, since its value depends on the distribution of the reference set. Another remarkable aspect of IGD$^+$ is its low computational cost, which allows to assess high-dimensional Pareto front approximations.

s-energy

The s-energy indicator [54] is given by:

$$E_s(A) := \sum_{i \neq j} \left\| \mathbf{a}_i - \mathbf{a}_j \right\|^{-s}, \tag{2.20}$$

where $A = \left\{ \mathbf{a}_1, \ldots, \mathbf{a}_{|A|} \right\}$, and $s > 0$ is a fixed parameter. This performance indicator has been used to discretize high-dimensional manifolds since its minimization leads to a uniform distribution of the points in A, if $s \geq m-1$ [54, 40]. Therefore, s-energy has been used to assess diversity of approximation sets, though it is not Pareto compliant.

Two Set Coverage

Zitzler et al. [153] proposed a binary performance indicator that compares two sets $A, B \subseteq A$ in terms of their relative coverage:

$$C(A, B) := \frac{\left| \left\{ \mathbf{b} \in B; \exists \mathbf{a} \in A : \mathbf{a} \prec \mathbf{b} \vee \mathbf{a} = \mathbf{b} \right\} \right|}{|B|}. \tag{2.21}$$

This Pareto compliant performance indicator is defined as the mapping of the order pair (A, B) to the interval $[0, 1]$. If all points in A dominate or are equal to all points in B, then by definition $C = 1$. $C = 0$ implies the opposite. In general, $C(A, B)$ and $C(B, A)$ both have to be considered due to set intersecions not being empty.

Overall Non-dominated Vector Generation

The Overall Non-dominated Vector Generation (ONVG) measures the total number of non-dominated solutions found during an optimizer's execution. This Pareto non-compliant indicator is given by:

$$ONVG(A) := |\mathrm{NDS}(A)|. \tag{2.22}$$

Tables 2.1 and 2.2 provide some examples of performance indicator values corresponding to the sets shown in Figure 2.4 (page 13). The reference point for the hypervolume was set to $(1.1, 1.1)$. For the $R2$ indicator, the ASF function was chosen, and the weight vectors were generated by the Simplex-Lattice Design method [114] with $H = 10$ (see page 23). In the case of GD, IGD and Δ_p, the parameter p was set to 2, and the reference set, also for IGD$^+$, was the leftmost approximation of Figure 2.4.

Table 2.2: Two set coverage of the approximation sets shown in Figure 2.4 (page 13). The best results are in **boldface.**

Set	**A**	**B**	**C**	**D**
A	-	**1.00**	**0.20**	0.00
B	0.00	-	0.00	0.00
C	0.00	0.80	-	0.00
D	0.00	0.80	0.00	-

Table 2.1: Performance indicators of the approximation sets shown in Figure 2.4 (page 13). The best results are in **boldface**.

Set	**Hypervolume**	$R2$	**GD**	**IGD**	Δ_p	**IGD$^+$**	s**-energy**	**ONVG**
A	0.3848	1.2534	**0.0000**	**0.0000**	**0.0000**	**0.0000**	223.5093	10
B	0.2823	1.3212	0.0671	0.0671	0.0671	0.0655	**215.2461**	10
C	**0.3890**	**1.2499**	0.0492	0.0682	0.0682	0.0159	256.0155	10
D	0.3653	22316.5414	0.0653	0.0770	0.0770	0.0437	269.3581	10

2.1.6 Scalarizing functions

A scalarizing function, also known as utility function or aggregation function, transforms the original MOP (2.1), defined on page 9, into a single-objective problem using a predefined target direction or *weight vector* $\mathbf{w} \in \mathbb{R}^m$. Such transformation is performed as follows:

$$\text{minimize} \quad u(\mathbf{f}'(\mathbf{x}); \mathbf{w}) \tag{2.23}$$

$$\text{subject to} \quad \mathbf{x} \in \mathcal{X}, \tag{2.24}$$

where $\mathbf{f}'(\mathbf{x}) := \mathbf{f}(\mathbf{x}) - \mathbf{z}$ and $\mathbf{z} \in \mathbb{R}^m$ is a reference point (usually the ideal point is adopted). Each component of $\mathbf{w}$ must satisfy $w_i > 0$, and although there is no particular reason beyond achieving uniformity among solutions, we also assume that $\sum_i w_i = 1$ [113].

A scalarizing function u is *Pareto compliant*, if it satisfies:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{X} : \mathbf{x} \prec \mathbf{y} \;\Rightarrow\; u(\mathbf{f}'(\mathbf{x}); \mathbf{w}) < u(\mathbf{f}'(\mathbf{y}); \mathbf{w}), \tag{2.25}$$

and *weakly Pareto-compliant* if it holds:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{X} : \mathbf{x} \preceq \mathbf{y} \;\Rightarrow\; u(\mathbf{f}'(\mathbf{x}); \mathbf{w}) \leq u(\mathbf{f}'(\mathbf{y}); \mathbf{w}). \tag{2.26}$$

Otherwise, a scalarizing function is non-Pareto compliant.

Let $p \in \mathbb{N}_+$, $\alpha \in \mathbb{R}_+$ and $\theta \in \mathbb{R}$ be the model parameters. The dot product is symbolized as $\bullet$, the absolute value of a real number is denoted by $|\cdot|$, and the magnitude of a vector is represented by $\|\cdot\|$. In Figure 2.5, we present the contour lines of some aggregation functions, whereas their corresponding definition and features are shown in Table 2.3. They differ in that they minimize some sort of a distance metric to the reference point, others combine two distance metrics, and a minority of them also consider the deviation to the weight vector. In all cases, their computational complexity is $O(m)$. Several of these scalarizing functions can generate (weakly) Pareto optimal solutions.

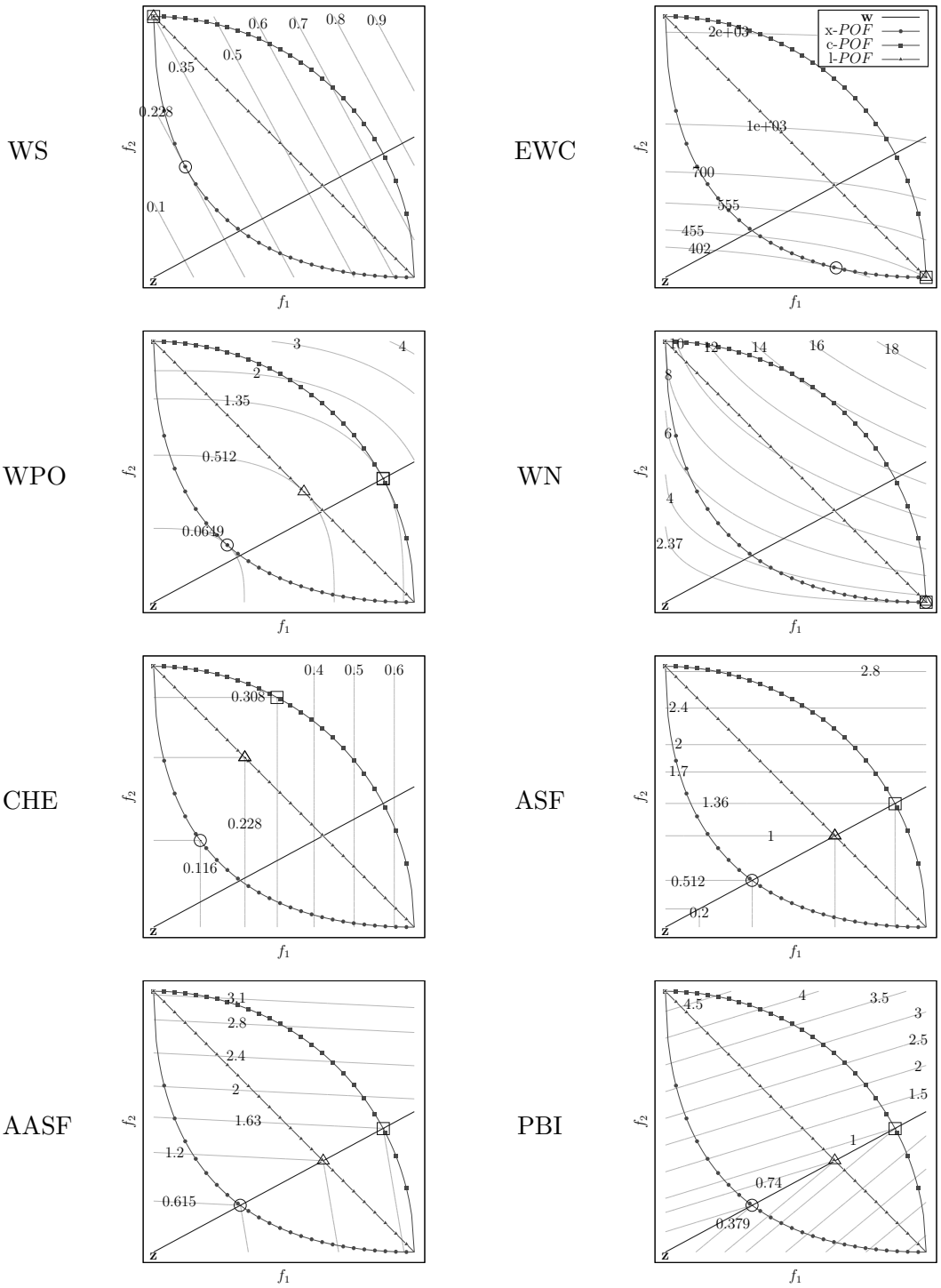


Figure 2.5: Contour lines of some scalarizing functions for the weight vector $\mathbf{w} = (0.65, 0.35)$. The big non-filled shapes denote optimal solutions for each of the Pareto front shapes.

Table 2.3: Some scalarizing functions and their features. Pareto front shapes are abbreviated to x (convex), c (concave) or l (linear). $\prec$ ($\preceq$) denotes compatibility with (weak) Pareto optimality. $||$ means that the optimal objective vector $\mathbf{y}^*$ is nearly parallel to the weight vector $\mathbf{w}$.

Acronym	Full Name	Minimize $u(\mathbf{y};\mathbf{w}) :=$	Support	Model Parameter
WS	weighted sum	$\sum_i w_i y_i$	x $\prec$	-
EWC	exponential weighted criteria	$\sum_i \left(e^{p\,w_i} - 1\right) e^{p\,y_i}$	x, c, l $\prec$	$p = 100$
WPO	weighted power	$\sum_i \frac{(y_i)^p}{w_i}$	x, c, l $\prec$ $\|\|$	$p = 3$
WN	weighted norm	$\left(\sum_i \frac{\lvert y_i\rvert^p}{w_i}\right)^{\frac{1}{p}}$	x, c, l $\prec$ $\|\|$	$p = 0.5$
CHE	chebyshev function	$\max_i \left\{w_i \lvert y_i\rvert\right\}$	x, c, l $\preceq$	-
ASF	achievement scalarizing function	$\max\left\{\frac{y_i}{w_i}\right\}$	x, c, l $\preceq$ $\|\|$	-
AASF	augmented achievement scalarizing function	$\max\left\{\frac{y_i}{w_i}\right\} + \alpha \sum_i \frac{y_i}{w_i}$	x,c,l $\prec$ $\|\|$	$\alpha = 1e^{-4}$
PBI	penalty boundary intersection	$d_1 + \theta d_2$, where $d_1 := \left\lvert \mathbf{y} \cdot \frac{\mathbf{w}}{\lVert\mathbf{w}\rVert} \right\rvert$ and $d_2 := \left\lVert \mathbf{y} - d_1 \frac{\mathbf{w}}{\lVert\mathbf{w}\rVert} \right\rVert$	x, c, l $\|\|$	$\theta = 5$

2.2 Multi-Objective Evolutionary Algorithms

Throughout the years, Multi-Objective Evolutionary Algorithms (MOEAs) have successfully shown their effectiveness in solving MOPs [123, 21, 79, 26, 20]. This is mainly because these methods can find discrete approximations to the Pareto optimal set in one single run without requiring particular assumptions, such as continuity or differentiability. Instead, MOEAs perform random search strategies that mimic Darwin's principle of natural selection, where the fittest individuals must achieve the pursued goals of Subsection 2.1.4 (page 13). The main elements of a MOEA are:

1. The evolutionary operators, parent selection (π), variation (ν), and survival selection (σ).

2. A population of individuals (or solutions) at generation t, $P_t^\mu = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_\mu\}$, where $\mathbf{a}_i \in \mathcal{X}$ represents an individual.

3. The iterative rule $P_{t+1} = \sigma\left\{P_t \cup \nu[\pi(P_t)]\right\}$.

A few MOEAs may also handle a secondary population, better known as an external archive, which keeps the non-dominated solutions found so far. At the end, the external archive, or in its absence, the final population, contains the approximation set.

Moreover, some MOEAs score the population by assigning a fitness number to each individual, measuring its performance regarding convergence, diversity, or a combination of the two. Those individuals with a better fitness have a high probability of being selected, either for reproduction or to constitute the next generation.

In the remainder of this section, we turn our attention to the evolutionary operators (Subsection 2.2.1) and how the fitness is assigned (Subsection 2.2.2). Interested readers may consult [21] for more details.

2.2.1 Operators

The *parent selection* $\pi : \mathcal{X}^\mu \to (Q \subseteq \mathcal{X}^\mu)$ decides which individuals will contribute to the creation of new ones. The common strategies are random sampling and tournaments where the fittest member from an arbitrary candidate subset becomes a parent.

The *variation operator* $\nu : (Q \subseteq \mathcal{X}^\mu) \to \mathcal{X}^\lambda$ produces λ new individuals from a subset of μ parents. It is worth noticing that a wide range of variation operators has been proposed for different *encoding representations* of the decision vector. Examples of variation operators for binary encoding are: single-point crossover, two-point crossover, and binary mutation; whereas for real encoding the most common choices are: simulated binary crossover (SBX) and polynomial-based mutation [27].

Finally, the *survival selection* determines the parent population of the next generation. There are two schemes: $\sigma_{(\mu,\lambda)} : \mathcal{X}^\lambda \to \mathcal{X}^\mu$ in which the best μ ($\mu \leq \lambda$) offspring replace parents, and $\sigma_{(\mu+\lambda)} : (\mathcal{X}^\mu \cup \mathcal{X}^\lambda) \to \mathcal{X}^\mu$ in which the best individuals from the union of parents and offspring are chosen. The most simple is the steady-state selection, which refers to $\sigma_{(\mu+1)}$. In general, the scheme $\sigma_{(\mu+\lambda)}$ is preferred to incorporate *elitism*, a milestone in MOEAs, since it guarantees convergence to the Pareto optimal front [111]. An additional option to introduce elitism is by using an external archive.

2.2.2 Selection strategies

MOEAs adopt mainly three strategies to select solutions: Pareto dominance, aggregation, and performance indicators. *Pareto based* MOEAs favor individuals that are

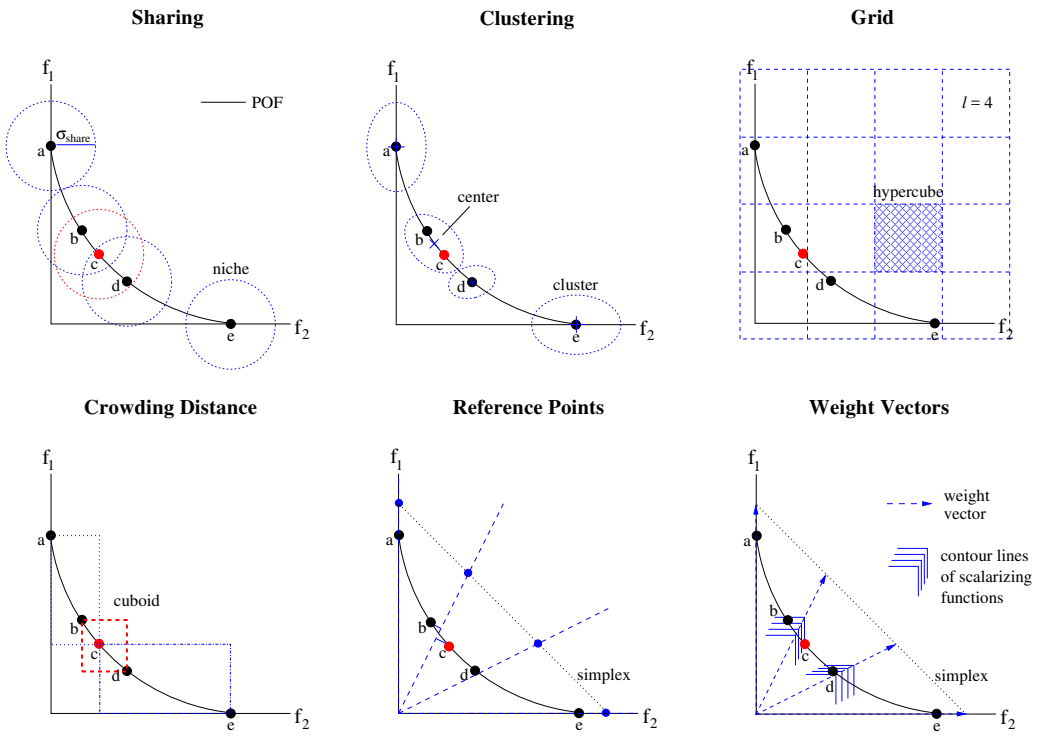


Figure 2.6: Common diversity mechanisms in multi-objective optimization. In all cases, solution c is the candidate for removal.

less dominated by other members in the population. For example, in Pareto ranking [37], a solution is scored according to the number of solutions that dominate it. Whereas in the non-dominated sorting algorithm [30], the population is classified into fronts; each one includes solutions that are non-dominated by inferior fronts. In other approaches, the fitness of an individual considers both dominating and dominated solutions [156, 155]. However, when it is not possible to discern among solutions using Pareto dominance, a secondary selection criterion is applied oriented to improve diversity, such as fitness sharing, clustering, grids, crowding distance, or reference points. The application of these methods is usually in objective space. Next, we describe each of them (see Figure 2.6 for examples).

Sharing methods [42] degrade an individual's fitness by dividing it by a *niche count*, which estimates how crowded is the individual's neighborhood. The *niche* or neighborhood is represented by a hypersphere of radius σ_{share} surrounding the current position of such individual. Diversity population depends on the proper setting of this parameter. In *clustering methods* [145], the overall population is grouped into subsets, called *clusters* within which the individuals are closer in terms of their Euclidean distance. The μ representative solutions (or cluster centers) constitute the next generation. The most popular algorithms are average linkage [156] and k-nearest-neighbor (kNN) [155] as they do not require extra parameters. In *grid methods* [78], objective space is partitioned into l^d hypercubes, where l is the number of subdivisions of the space and d is the dimension. The number of solutions in a hypercube is then used as

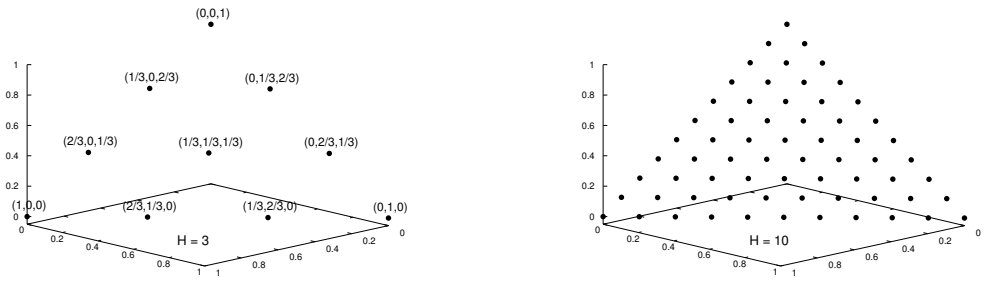


Figure 2.7: Examples of the Simplex-Lattice Design method.

an approximate form of niche count. Another approach is the *crowding distance* [30], given by the average distance in each dimension of the two nearest neighbors that surround a solution. Whereas in the *reference-point methods*, a set of well-dispersed points is required. In this case, the population is associated with the lines passing through the origin and these points. Those solutions having the closest perpendicular distance to segregated lines are retained.

In contrast, *aggregation-based MOEAs* [148] decompose a MOP into several single-objective subproblems through a scalarizing function (see Section 2.1.6 on page 18). Subproblems are associated with different weight vectors, which should be well distributed in $(0, 1]^m$.

It is worth mentioning that, in algorithms based on reference points or weight vectors, the m-tuple is usually generated by the *Simplex-Lattice Design method* [114]. In this case, each component takes a value from $\{10^{-2}, 1/H, 2/H, \ldots, H/H\}$ where $H \in \mathbb{N}$. The quantity of m-tuples is represented by the combinatorial number C_{m-1}^{H+m-1}. This set is equally spaced over a simplex as shown in Figure 2.7.

Indicator-based MOEAs favor solutions that highly contribute to a performance indicator (see Section 2.1.5 on page 13). Because of their compatibility with a relaxed form of Pareto dominance, the most relevant are the unary indicators: hypervolume, IGD^+ and $R2$, along with the binary ϵ indicator [157]. The fitness or individual's contribution to a unary indicator is defined as: $\Delta_I(a, A) := I(A) - I(A \setminus \{a\})$, where I is one of the above mentioned indicators. Appendix A is dedicated to the computation of the hypervolume contribution.

Finally, Tables 2.4 and 2.5 summarize the features of the most representative MOEAs developed over the years, describing the adopted strategies for each evolutionary operator.

Table 2.4: Features of the most representative MOEAs.

Year	**Optimizer/ Reference**	**Full Name**	**Parent Selection** (π)	**Variation** (ν)	**Survival Selection** (σ)	**Observations**
1993	MOGA [37]	Multi-Objective Genetic Algorithm	Similar individuals are recombined	Two-point crossover and binary mutation	Non-elitist (μ, μ). Fitness based on Pareto ranking and niching in objective space	Parameter σ_{share} is estimated by solving a polynomial equation
1993	NPGA [61]	Niched Pareto Genetic Algorithm	Pareto domination tournaments, candidates are compared to a sample and ties are solved by niching in objective space	Single-point crossover and binary mutation	Non-elitist approach: (μ, μ)	Additional parameters: σ_{share} and sample size. The latter is critical to the selection pressure and premature convergence
1994	NSGA [118]	Nondominated Sorting Genetic Algorithm	Stochastic remainder selection. Dummy fitness based on non-dominated fronts and sharing in decision variable space	Binary operators	Non-elitist approach: (μ, μ)	Computational complexity of $O(\|P\|^3 m)$. Performance sensitive to σ_{share} parameter
1998	SPEA [156]	Strength Pareto Evolutionary Algorithm	Tournaments (including external solutions). Fitness based on Pareto dominance	Single-point crossover and binary mutation	Elitist scheme with selection (μ, μ)	External archive pruned to a limit size using a clustering method (average linkage)
1999	PAES [78]	Pareto Archived Evolution Strategy	Not applicable, except for the variant $(\mu + \lambda)$, whith $\mu > 1$ or $\lambda > 1$. In this case, selection is performed via tournaments	Random mutation (similar to a hill-climber)	Elitist $(1 + 1)$, comparisons are made using Pareto dominance and a degree of crowding, which is estimated from an external archive	External archive truncated to a fixed size by an adaptive grid, which recursively splits the objective space in $O(2^{lm})$ subdivisions, where l is a user defined parameter
2000	NSGA-II [30]	Nondominated Sorting Genetic Algorithm II	Tournaments based on a preference relation ($\prec_n$), which considers non-dominated fronts and crowding distance	Simulated binary crossover (SBX) and Polynomial-based mutation	Elitist scheme $(\mu + \mu)$, based on $\prec_n$	Computational complexity of $O(\|P\|^2 m)$. Constraint handling by modifying Pareto dominance
2001	SPEA2 [155]	Strength Pareto Evolutionary Algorithm 2	Tournaments. Fitness based on Pareto dominance and density information (kNN method)	Depending on the problem, same as SPEA or NSGA-II	Elitist scheme $(\mu + \lambda)$, truncation considers distances among individuals	Boundary points are preserved. Computational complexity of $O(\|P\|^2(\log \|P\| + m))$

Table 2.5: Features of the most representative MOEAs (cont'd).

Year	**Optimizer/ Reference**	**Full Name**	**Parent Selection** (π)	**Variation** (ν)	**Survival Selection** (σ)	**Observations**
2004	IBEA [154]	Indicator-Based Evolutionary Algorithm	Binary tournaments. Fitness is given by an exponential function, which calculates the performance indicator of each distinct pair of individuals	Depending on the problem, same as SPEA or NSGA-II	Elitist scheme ($\mu + \mu$), which iteratively removes the worst individual, updating fitness values	Framework of binary indicators, compliant with Pareto dominance. Required scaling factor $\kappa > 0$, dependent on the problem and the indicator used
2005	SMS-EMOA [34, 9]	S-metric Selection Evolutionary Multi-Objective Optimization	Random sampling	Simulated binary crossover (SBX) and Polynomial-based mutation	Elitist ($\mu + 1$). The solution with the worst hypervolume contribution of the last non-dominated front is removed	Expensive computational complexity of $O(\|P\|^{m-1})$. Hypervolume is maximized through generations
2007	MO-CMA-ES [66]	Multi-Objective Covariance Matrix Adaptation Evolution Strategy	In the scheme $\lambda(\geq 1)$ every individual in the population is mutated	Sampling from a multi-variate normal distribution with the parent as the mean vector and an adaptive covariance matrix	Elitist scheme $\mu(1 + \lambda)$. Population is ranked according to the non-dominated fronts and the contributing hypervolume (or crowding distance)	Invariant against rotation, scaling and translation of the search space. Parameter adaptation of the mutation distribution per individual. Computationally expensive for problems with many variables or objectives
2007	MOEA/D [148]	Multi-Objective Evolutionary Algorithm based on Decomposition	Random sampling from the neighborhood T_i of each i-th individual, associated with a scalar optimization problem	Only one offspring is created. Same as SPEA2, including differential evolution	Elitist scheme ($\mu + \mu$). A child i replaces solutions from T_i if its corresponding scalarizing function is equivalent or better	Framework of scalarizing functions with a complexity of $O(\|P\|\|T\|m)$. Required set of weight vectors and neighborhood size ($\|T\| \ll \|P\|$)
2011	HypE [7]	Hypervolume Estimation Algorithm for Multi-Objective Optimization	Binary tournaments using the hypervolume contributions of the population	Depending on the problem, SBX and Polynomial-based mutation	Elitist scheme ($\mu + \mu$). Solutions with the worst hypervolume contributions of the last non-dominated front are discarded	If $m \leq 3$ the exact hypervolume is calculated; otherwise, it is estimated using Monte Carlo simulation, where a large number of sampling points is required
2014	NSGA-III [28]	Nondominated Sorting Genetic Algorithm III	Binary tournaments for constrained MOPs; otherwise, random sampling takes place	SBX and Polynomial based mutation	Elitist ($\mu + \mu$), crowding distance is replaced by a niching strategy, which requires a set of well spread reference points	Computational complexity of $O(\|P\|^2 m)$. Algorithm designed for solving MOPs with up to 15 objectives

2.2.3 Parallelization

At each generation of a MOEA, the computational cost depends on two factors: 1) the computational complexity of evaluating the MOP, and 2) the scalability of the input parameters, such as the number of decision variables, objectives and population size. In the first case, MOEAs require a large number of function evaluations to approach Pareto optimal fronts. However, a vast majority of MOPs cannot be expressed in algebraic form. Thus, their evaluation is conducted by time-consuming simulators. Even though some approaches have been proposed to reduce the execution time by exploiting knowledge acquired during the search [112, 77], the quality of the final solutions often worsens. Examples of the second case, include applications that deal with high dimensional spaces, such as large-scale MOPs or many-objective problems. These types of MOPs make MOEAs to slow down considerably [5, 96, 56]. Some other applications need large population sizes for improving accuracy [41] or covering more regions of the search space [2]. Though most MOEAs run in polynomial time with respect to their population size, the storage limit may also becomes an issue.

The parallelization of MOEAs (pMOEAs) arises as an attractive option to address these factors, where the basic idea is to divide somehow the MOEA into several tasks. Each task is solved simultaneously on a different processor and, once all of them have been completed, the results are combined to provide a solution to the MOP [3, p.2]. Processors can be in the same machine, or distributed in a collection of machines interconnected by a network [22]. The wide acceptance of pMOEAs is mainly because they can produce substantial gains in performance, and in some cases, they can also improve the accuracy of the results with respect to their serial counterparts.

In spite of the fact that the iterative rule of MOEAs is a purely sequential process, at least two parallelization strategies can be applied [94]: 1) parallelization of the computations, in which the operations commonly applied to each individual are performed in parallel, and 2) parallelization of the population, in which the population is split into different parts, each one evolving in semi-isolation. Both strategies are considered in the four major *parallel models*[4] of MOEAs, which are the master-slave, island, diffusion and hybrid models. In the following, we describe each of them. This thesis focuses on the paradigms master-slave (Chapter 7) and island (Chapter 5), which were implemented in EMO Project (Chapter 6). For more information, readers are referred to the works of Coello et al. [21, p.443], Luna and Alba [94], Van Veldhuizen et al. [127], and Talbi et al. [122].

Master-Slave Model

The master-slave model (Figure 2.8a) parallelizes the operations of a MOEA that do not require information about all individuals at the same time. Therefore, the population (P) is managed by a central or *master* processor, which applies the parent (π)

[4] A parallel model is an abstraction of a computer system architecture that defines the logic in which MOEAs are executed in parallel. Therefore, the implementation of a parallel model is not tied to any parallel architecture.

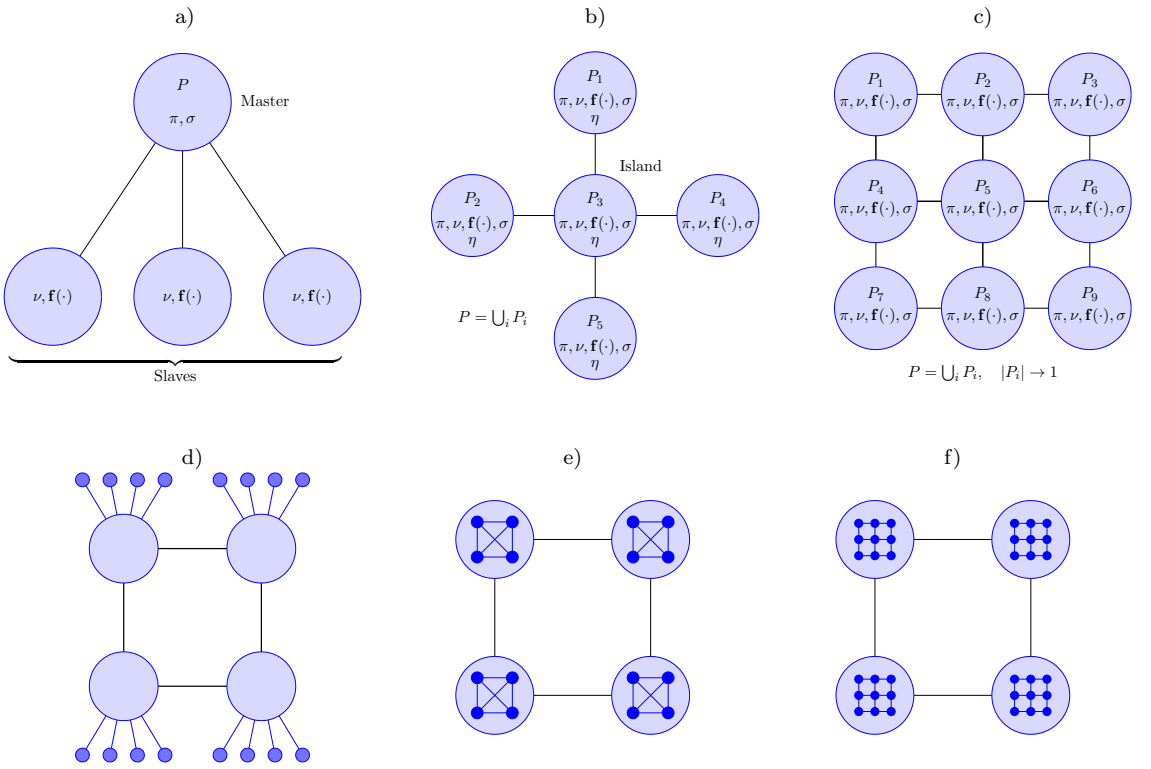


Figure 2.8: Parallel models for MOEAs: a) master-slave, b) island using a star topology, c) diffusion, and examples of hybrid models: d) island/master-slave, e) island/island, f) island/difussion.

and survival (σ) selections. On the other hand, *slave* processors simultaneously apply the variation operations (ν) and the evaluation of the objective functions ($\mathbf{f}$).

There are two variants of this model. One is the *synchronous master-slave*, where the central processor waits to receive the results of all the population before proceeding into the next generation. In this case, the search behavior is conceptually identical to a serial MOEA, with its execution time being the only difference. The other variant, suitable for heterogeneous architectures, is the *asynchronous master-slave*, where the central processor does not stop to wait for any slow slave and thus, the search space exploration is different from a serial MOEA.

In general, the master-slave model is recommended for large-scale MOPs or time-consuming objective functions.

Island Model

The island model (Figure 2.8b) is inspired by the natural phenomenon of populations evolving in relative isolation, such as it might happen within some archipelago. In this case, the overall population is divided into a number of independent subpopulations (P_i) that live in islands (processors) and evolve through the execution of a serial MOEA. Islands are connected in a physical or logical topology, such as line,

tree, torus, mesh or ring. Occasionally, neighboring islands interact with each other, exchanging some individuals. This operation is known as *migration* (η), and its goal is to introduce diversity into subpopulations, avoiding to get stuck in local optima. However, the new operation introduces extra parameters: the number of solutions to migrate, how often migration occurs, the migration policy that determines which individuals migrate and which are replaced in the destination island. These parameters along with the topology are important factors in the performance of an island pMOEA because they determine how fast (or how slow) a good solution spreads to other subpopulations.

The migration operation can be implemented using synchronous or asynchronous communication [95]. In the *synchronous* case, each island evolves a serial MOEA, selects emigrants, sends copies of them to destination islands, and waits to receive immigrants from source islands. In the *asynchronous* case, the last step is different; the island does not wait, and if immigrants are available, they are incorporated into its subpopulation, but otherwise, it computes the next generation using the latest immigrants received from the neighborhood.

Regarding the migration and replacement schemes, there are many of them available in the literature [21, p. 494]. Due to predictable communication costs and easy implementation, in this thesis, we focus on those that migrate a constant number of individuals ($nmig$) at each event, such as:

Random Migration/Replacement (R): This scheme selects $nmig$ random solutions for migration or replacement.

Elitist Random Migration (ER): Migrate $nmig$ random individuals from the non-dominated set. If there are not enough members, the rest is selected at random from the remainder of the population.

Elitist Ranking Migration (EK): Migrate $nmig$ random individuals from the non-dominated set. If there are not enough members, the rest is selected from the successively ranked Pareto fronts.

Elitist Random Replacement (ER): $nmig$ random solutions, which are dominated by the immigrants, are replaced in the population. If the number of solutions is smaller than $nmig$, random immigrants are discarded. This method does not guarantee that the worst solutions will be replaced but ensures the retention of the non-dominated set.

Elitist Ranking Replacement (EK): Rank all Pareto fronts and replace individuals from the worst ranked front(s) with the immigrants. Solutions may be randomly selected from some front(s) if the number of individuals to be replaced does not match the number of individuals represented by the front being replaced.

Elitist Replacement (E): Combine immigrants with the current population, rank all Pareto fronts and remove individuals from the worst ranked front(s). This method replaces the solutions of the worst front of the current population with the best immigrants. When there is a tie between these two sets, individuals are randomly replaced.

The behavior of island pMOEAs differs from their serial counterpart, allowing the formation of niches, which can improve the search. Algorithms based on this model

can considerably reduce the execution time, and its use is recommended for clusters of computers with limited communication among them. They are also suitable for problems with large search spaces where a large population size is required.

Diffusion Model

In the diffusion model (Figure 2.8c), the population is dispersed among several subpopulations (P_i) living on different processors and having one or few individuals. Unlike the island model, there is no migration per se. However, subpopulations are interconnected in a neighborhood structure, where the variation operators (ν) are applied only within these overlapping neighborhoods. The neighborhood geometry could be a square, a rectangle, a cube, or any other shape depending upon the number of dimensions associated with the diffusion algorithm's topological design. Each geometry reflects some associated number and arrangement of neighbors within a multidimensional grid. As fittest individuals arise in different areas of the local topology, the aim is that they spread or diffuse slowly throughout the entire population. The diffusion model distinguishes itself for requiring high data-rate communications being suitable for massively parallel architectures, such as Graphics Processing Units (GPUs).

Hybrid Model

Hybrid models (Figure 2.8d-f) combine different paradigms to parallelize MOEAs. In most approaches, the fusion follows a two-level hierarchy, where at the upper level is the island model, and at the lower level is either the master-slave, diffusion or even the same island model.

2.2.4 Heuristics and its derivatives

The word *heuristic* derives from Greek "heuriskein," and means to find or discover. It can be a rule, a choice, a component, or a process. According to Merriam-Webster dictionary[5], heuristics involve or serve as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods. Heuristics have been one of the crucial elements in Artificial Intelligence to solve complex, uncertain and ambiguous problems, where there is no guarantee to find a correct solution. Heuristics can be seen as rules of thumb or empirical knowledge that help to guide an algorithm to find acceptable solutions in a reasonable time, narrowing the search space considerably [4]. Examples of heuristics are local search techniques, selection operators, variation operators, meta-heuristics, and Rechenberg's 1/5th rule for adapting the step size in a Gaussian mutation [110]. In the following, we define two derived concepts from the word heuristic: meta- and hyper-heuristic.

The Greek etymology *meta* means "after" or "beyond," and it is an abstraction behind another concept used to complete the latter. A *meta-heuristic* is an iterative

[5] `www.merriam-webster.com/dictionary/heuristic` (accessed November 21, 2017)

process, which guides a subordinate heuristic using ideas derived from artificial intelligence, biological, mathematical, and physical sciences to find efficiently near-optimal solutions [104]. Meta-heuristics can be considered as approximate methods that work as general frameworks to attack hard optimization problems, where classical heuristics have failed [104]. Examples of meta-heuristics are evolutionary algorithms (genetic algorithms, genetic programming, evolution strategies, differential evolution), particle swarm optimization, simulated annealing, and tabu search [104, 121, 39].

On the other hand, the Greek prefix *hyper* means "over" or "above," and it denotes something above the standard. A *hyper-heuristic* is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems [15]. Hyper-heuristics are high-level approaches that operate on a search space of heuristics or *heuristic pool* instead of the search space of the problem at hand. Hyper-heuristics have been promoted with the aim to provide generally applicable search methodologies. A common phenomenon is that simple heuristics are successful in solving a particular kind of problems. However, when applying them to new problems or slightly modified instances, heuristics may face difficulties in their performance. Moreover, to identify which heuristic works efficiently in a given problem is an arduous task, sometimes prohibitive for computationally expensive applications.

As shown in Figure 2.9, Burke et al. [16] proposed a taxonomy of hyper-heuristics considering two dimensions: 1) the nature of the heuristics' search space, and 2) the different sources of feedback information. Regarding the nature of the search space, there are two options: a) *heuristic selection*, which are methodologies for choosing existing heuristics, and b) *heuristic generation*, which are methodologies for generating new heuristics from the components of existing ones. Regarding the source of feedback information obtained during the search process, there are three options: c) *no-learning*, in which there is no learning mechanism and the heuristic selection is based on either a random or an exhaustive process. d) *offline learning*, in which knowledge is gathered in the form of rules from a set of training instances, that will hopefully generalize to solve unseen instances, and e) *online learning*, in which the learning takes place while the algorithm is solving an instance of a problem.

2.3 Related Work

In this section, we examine the most relevant previous related work. The use of collaborative approaches working as hyper-heuristics can be found across Operational Research, Computer Science and Artificial Intelligence. Although the ideas behind hyper-heuristics can be traced back to early 1960s in single-objective optimization, little attention has been paid in the field of multi-objective optimization. Early approaches date back to 2005, where hyper-heuristics have been used to solve combinatorial applications, such as space allocation and timetabling [17], decision-tree induction algorithms [8], bin packing and cutting stock problems [43], integration and test order problems [50, 51, 97], spanning trees [83], and job shop scheduling [128]. In integer programming, hyper-heuristics have been applied to solve the knapsack problem [84],

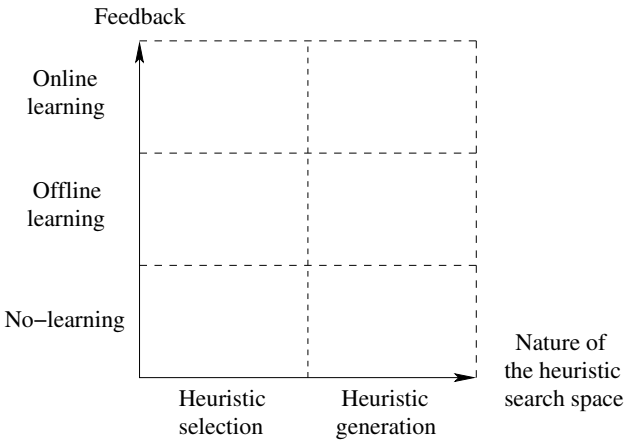


Figure 2.9: A simplified classification of hyper-heuristic approaches.

as well as software module clustering [86, 85]. In this branch, a theoretical work presented by Qian et al. [107] supported the effectiveness of hyper-heuristics, proving that they can speed up the optimization via running time analysis. Regarding continuous optimization, there are a few works which rely on Pareto dominance, and therefore limiting their use in many-objective optimization. Besides, a large number of efforts combine strategies using a pool of variation operators. However, currently, there is no single work handling hyper-heuristic within the parent or survival selection mechanism of a MOEA. In the following, we review in detail multi-objective hyper-heuristics for continuous search spaces.

A Multi-ALgorithm Genetically Adaptive Method (AMALGAM), proposed by Vrugt et al. [133], is an on-line selection hyper-heuristic that operates as NSGA-II. However, the offspring are also created using the variation operators of other stochastic methods, such as Differential Evolution [120], Particle Swarm Optimization [76] and Adaptive Metropolis Search [53]. Even though all these methods participate during the optimization process, those that exhibit the highest reproductive success are favored. Consequently, the number of individuals to be created by a method $i \in \{1, \ldots, k\}$ at the next generation is calculated as:

$$N_{t+1}^{i} = \max \left\{ N_{\min}, \frac{N\left(\frac{P_{t+1}^{i}}{N_{t}^{i}}\right)}{\sum_{j=1}^{k} \frac{P_{t+1}^{j}}{N_{t}^{j}}} \right\}, \tag{2.27}$$

where $N_{\min}$ is the minimum number of individuals, N is the population size, and P_{t+1}^{i} is the number of surviving individuals produced by a method i at generation $t+1$.

Another distinction in this approach is the initialization of the population, which is performed using Latin hypercubes sampling. AMALGAM has been used for the calibration of hydrologic models [108, 144, 151, 134, 150] and weather prediction models [132]. The experimental results in [133, 144] shown that AMALGAM significantly improved NSGA-II, requiring much lower function evaluations to approach the Pareto

optimal front. In [134], Vrugt et al. presented a parallel version of AMALGAM, where the synchronous master-slave model was implemented on a distributed computer network of 10 nodes using standard MPI toolbox in Octave.[6] Each node equipped with a dual-core Intel Xeon 3.4-GHz processor with 4GB of memory. The algorithmic part of AMALGAM, including the generation of the offspring using different meta-heuristics, and the collection of all results is controlled by the master; whereas the evaluation of the costly objective functions is distributed across the slaves. This parallel version drastically reduced time, achieving an almost linear speedup. Later, Zhang et al. [150] extended this work to a computing software, called Python-based Parallel Soil and Water Assessment Tool (PP-SWAT). This software employs Python 2.6, mpi4py and OpenMPI to parallelize AMALGAM. The software was tested on a cluster of 290 nodes. Each node equipped with a Intel Xeon 5560 (quad-core) or 5660 (hex-core) microprocessor running at 2.8 GHz and having 48 GB of memory. The test results in [150] showed that PP-SWAT can achieve speedups of 45-109 depending on the complexity of the watershed model. The major drawback of AMALGAM is that the involvement of multiple methods leads to the definition of several parameters.

The **Multi-Strategy Ensemble Dynamic Multi-Objective Evolutionary Algorithm (MS-MOEA)**, proposed by Wang and Li [140], is an offline selection hyper-heuristic that adopts the fundamental principle of AMALGAM of combining different variation operators. This approach works as ϵ-MOEA [29] with an external archive that is pruned to a limit size using the hypervolume indicator. The heuristics for generating new individuals are two re-initialization techniques, which are based on random sampling and Gaussian distribution with mean around previous optimal solutions; the genetic operators SBX and polynomial based mutation; the Differential Evolution strategies DE/rand/1 and DE/current to best/1; as well as Gaussian mutation. MS-MOEA was designed to solve dynamic multi-objective optimization problems, i.e., problems in which the Pareto optimal set or the Pareto optimal front change in time. The heuristic selection is performed by a set of rules derived from knowledge. If there is an environmental change, then one re-initialization technique is applied under certain probability. Genetic operators are used in early stages of evolution. Once convergence has been achieved, Differentia Evolution is employed to improve diversity. Here, each strategy creates an offspring. After a fixed number of solutions were created, Gaussian mutation is launched for escaping from local optima. It is worth noticing that convergence is detected when the external archive has been full during a certain number of generations. Experimental results in [140] demonstrated that MS-MOEA accelerated convergence speed on bi-objective test problems. This behavior is attributed to the combination of multiple variation operators and a hypervolume-based archive.

The **Markov Chain Hyper-Heuristic (MCHH)**, proposed by McClymont et al. [98], is a selection heuristic with online learning working as a $(\mu + \lambda)$-Evolution Strategy with an unlimited external archive. The pool of heuristics is composed of three variation operators: mutation, replication, transposition, and clone. All of them

[6] https://www.gnu.org/software/octave

operate on the decision variables of a given solution. The heuristic selection uses a Markov chain, which can be seen as a directed graph where every vertice is connected to each other vertice and to itself. A vertice represents a state (heuristic), and the weight of an edge out represent the probability of moving from the current state to the destination state. All edges out of a state must sum one. The Markov chain stochastically selects the next heuristic biased by these probabilities. The selected heuristic is employed during ϵ generations. Then, its performance is measured by counting the number of parents that were dominated by each offspring produced by such heuristic. This performance is used to update the corresponding probability in the Markov chain using a reinforcement learning. In this case, if the resulting score is greater than some threshold γ, the weight is increased by α. Otherwise, the weight is degraded by β. Next, the sum of outflow edges is normalized. In [98], MCHH was compared with single heuristics, a random hyper-heuristic, and a hyper-heuristic from the literature [17] using the DTLZ test problems with three objectives. Experimental results showed that MCHH was helpful in identifying good sequences of heuristics and discarding the worthless ones, such as clone, which was intentionally introduced. Some disadvantages of this approach are that there is no mechanism to encourage diversity, and the proper setting of the learning parameters $(\gamma, \alpha, \beta, \epsilon)$.

The Metaheuristics-based Extensible Tool for Cooperative Optimization (METCO) [88] is based on the island model and the cooperation of a set of MOEAs, which grants more computational resources to those algorithms that show a more promising behavior. A coordinator node is in charge of maintaining the global solution and selecting the configurations that are executed on the islands. A *configuration* consists of a MOEA plus the variation operators and the set of parameters which define them (population size, mutation and crossover rates, etc.). These parameters are defined by the user. The global solution set is obtained by merging local results achieved by each of the islands and its size is limited using the crowding distance operator. Besides the global stopping criterion, a local stopping criterion is defined for the execution of the MOEAs on the islands. When the local stopping criterion is reached, the configuration is scored using a performance indicator. Then, the coordinator applies the hyperheuristic, selecting the configuration that will continue executing on the idle island. If the configuration has changed, the subpopulation is replaced by a random subset of the currently global solution. As scoring strategies, the contribution metric [88] and the hypervolume [87, 90] have been incorporated into the scheme, being the latter more suitable when the number of configurations increases. The scheme was tested on the ZDT and WFG test problems for instances with two objectives [87], using SPEA, SPEA2, NSGA-II, IBEA and four variation operators. In total, sixteen configurations were considered for four islands using an all-to-all topology, asynchronous communication and elitist migration. The proposal was among the best pMOEAs when compared with an algorithm that randomly changes the configuration on islands and the homogeneous islands of each of the sixteen configurations. In another study [117], different metaheuristics were contemplated: NSGA-II, Evolution Strategies and Differential Evolution versions of

NSGA-II, SPEA2, MOPSO, MOCell and IBEA. Authors reported promising results, being IBEA the most used configuration, while MOPSO and MOCell had the worst performance. Furthermore, the scheme has been extended to a framework [89, 91], which is implemented in C++ and MPI. In order to obtain an almost linear speedup, this scheme is suggested for problems with costly objective function evaluations [89]. One disadvantage of this approach is that is not scalable for problems with more than three objectives.

The **MOEA/D Hyper-Heuristic (MOEA/D-HH)**, proposed by Gonçalves et al. [46], is an online selection hyper-heuristic that is coupled to a MOEA/D variant [149]. In this approach, an adaptive choice function is used to determine the Differential Evolution (DE) strategy that should be applied to generate individuals at each iteration. Its authors proposed the following choice function to compute the score of a given heuristic h:

$$CF(h) = \alpha\phi\left(f_1(h) + f_2(g,h)\right) + \delta f_3(h), \tag{2.28}$$

where g is the current heuristic, f_1 and f_2 are the mean rewards of applying h alone and g followed by h, respectively. The *reward* is calculated as the difference between the Chebyshev scalarizing function value of the parent and the child. f_3 corresponds to the time elapsed since h was last selected. α is a scale factor, which is problem dependent and needs to be calibrated a priori. ϕ and δ are parameters that control the intensification and diversification of the selection of the best heuristics, respectively. Here, δ is set to $1-\phi$ and MOEA/D-HH automatically updates ϕ through generations. The heuristic with a higher CF value is chosen to create offspring, which is later perturbed by polynomial mutation. The pool of heuristics consists of five DE strategies: 1) *DE/rand/1/bin* of slow convergence speed and good exploration capability, suitable for solving multimodal problems; 2) *DE/current-to-rand/1/bin* for enabling the algorithm to solve rotated problems more effectively; 3) *DE/nonlinear*, which includes a non-linear part of the DE mutation operator; 4) *DE/rand/2/bin* and 5) *DE/current-to-rand/2/bin*, which may provide better perturbations than the two first strategies. In [46], MOEA/D-HH was tested on ten unconstrained instances of the CEC'09 benchmark, improving the performance of MOEA/D when using a single heuristic. In [45], the authors proposed a slight modification, where the values of f_1 and f_2 in (2.28) only considered accumulated normalized rewards of recent subset heuristics. In [44], this later version was applied to solve an environmental/economic dispatch problem.

Another work destined to solve many-objective optimization problems is **the Multi-Objective Sequence-based Selection Hyper-Heuristic (MOSSHH)**, which was proposed by Walker and Keedwell [136]. This online selection hyper-heuristic is based on a hidden Markov model to determine the mutation strategy to be applied for generating a single child from the current parent. Thus, this approach works as a $(1+1)$-Evolution Strategy complemented with an external archive, which keeps all the non-dominated solutions discovered so far. The pool of seven mutation heuristics consists primarily of 1) adding noise to the current solution using three

different continuous probability distributions, and 2) replacing the parent (or only a variable) with another one, whether randomly created or taken from the archive. At each iteration, the child replaces the parent if the former dominates the second. However, in another work [137], this comparison rule was changed by strategies based on the hypervolume indicator[7], the favor relation[8] and the average rank[9]. Moreover, the hidden Markov model is updated if the child is added to the archive and if it was better than the parent. In [137], the three strategies were independently applied to solve the DTLZ test problems having up to 6 objectives. The best results were obtained using either the hypervolume or the favor relation. Although its computational cost is low (even when using the hypervolume in high dimensionality), this hyper-heuristic is reported to face difficulties in problems with disconnected Pareto optimal regions in the search space (like DTLZ6). In [137], authors indicated that this might be due to the lack of crossover. Other disadvantages are that solutions are not uniformly distributed and the external archive may grow too much.

Table 2.6 summarizes the main features of above mentioned hyper-heuristics. On the other hand, the idea of using the simultaneous collaboration of different scalarizing functions within MOEAs dates back to 2003, when there were a few attempts to combine, at most, two scalarizing functions.

Hughes [65] scored the population using the CHE and the vector angle distance scaling (also introduced in that work) for solving continuous MOPs with two and three objectives. Although this method, called Multiple Single Objective Pareto Sampling (MSOPS), can deal with any Pareto front geometry, it may generate dominated solutions in disconnected regions, since the second scalarizing function is not compatible with any form of Pareto optimality.

Ishibuchi et al. [71] modified MOEA/D for automatically choosing between CHE and WS. The former was applied only for concave parts of the Pareto front, where local concavity was detected as long as an individual was identical to a certain number of neighbors. However, when frequent changes occurred, this approach did not perform well in combinatorial MOPs having up to 6 objectives. The reason was probably that both scalarizing functions drove individuals to entirely different regions when considering the same weight vector [71]. Shortly afterwards, this issue was further analyzed in [72], exploring two alternatives. In one approach an individual optimized a previously assigned scalarizing function, whereas in the second strategy the population increased linearly at the rate of $|P|$ individuals per scalarizing function. Each different subpopulation focused on a particular scalarizing function.

[7] A solution $\mathbf{x} \in \mathcal{X}$ is better than a solution $\mathbf{y} \in \mathcal{X}$, iff $\prod_i^m z_i - f_i(\mathbf{x}) > \prod_i^m z_i - f_i(\mathbf{y})$, where $\mathbf{z} \in \mathbb{R}^m$ is a reference point.

[8] A solution $\mathbf{x} \in \mathcal{X}$ favors a solution $\mathbf{y} \in \mathcal{X}$ ($\mathbf{x} \prec_f \mathbf{y}$), iff $|\{i : f_i(\mathbf{x}) < f_i(\mathbf{y}), 1 \leq i \leq m\}| > |\{j : f_j(\mathbf{x}) > f_j(\mathbf{y}), 1 \leq j \leq m\}|$.

[9] A solution $\mathbf{x} \in \mathcal{X}$ is better than a solution $\mathbf{y} \in \mathcal{X}$, iff $\sum_i^m r_i(\mathbf{x}) < \sum_i^m r_i(\mathbf{y})$, where $r_i \in \mathbb{N}$ is the rank of the solution according to the i^{th} objective. The ranking process considers the external archive.

Table 2.6: Summary of multi-objective hyper-heuristics

Year	Name	Class	Scope	Heuristic Pool	Heuristic selection	Parallel	Suport to many-objective	Diversity mechanism	External archive	Extra parameters	Meta-heuristic
2007	AMALGAM [133]	heuristic selection, online learning	variation operators from different optimizers	NSGA-II, Particle Swarm Optimization, Differential Evolution, and Adaptive Metropolis Search	highest reproductive success	sync. master slave [134, 150]	no	crowding distance	no	$N_{\min}$ + the heuristic parameters	NSGA-II
2008	METCO [88]	heuristic selection, online learning	search mechanism of diffferent MOEAs	SPEA, SPEA2, NSGA-II, IBEA, MOPSO, MOCell with 4 variation operators	hypervolume or contribution metric	async. island model	no	crowding distance	no	yes, given by the heuristic pool	NSGA-II
2010	MS-MOEA [140]	heuristic selection, offline learning	variation operators from different optimizers	NSGA-II, Differential Evolution, and Gaussian Mutation	rules derived from knowledge	no	no	ϵ and hypervolume indicators	yes	3 new + the heuristic parameters	ϵ-MOEA
2011	MCHH [98]	heuristic selection, online learning	variation operator	mutation, replication, transposition, and clone	Markov chain with a reinforcement learning	no	no	no	yes, unlimited	4	$(\mu + \lambda)$-ES
2015	MOEA/D-HH [46]	heuristic selection, online learning	variation operator	5 differential evolution strategies	adaptive choice function	no	yes	weight vectors	no	1	MOEA/D variant with CHE function
2016	MOSSHH [136]	heuristic selection, online learning	variation operator	7 mutation strategies	hidden Markov model	no	yes	no	yes, unlimited	no	$(1 + 1)$-ES

2.4 Summary

This chapter introduced the basic concepts of multi-objective optimization, which is widely used in several disciplines. We started by defining a MOP, the concept of optimality, and the Pareto dominance relations: strict and weak. We defined essential reference solutions in objective space: the ideal, nadir, and extreme points. Moreover, we denote the pursued goals in multi-objective optimization, which are convergence to the Pareto optimal front, and good diversity in objective space. Then, we defined performance indicator and a scalarizing function. In both cases, we provided examples. We summarized the features of the most representative MOEAs. Additionally, we presented the paradigms to parallelize MOEAs: master-slave, island, diffusion, and hybrid models. We clarified the difference among heuristic, meta-heuristic, and hyper-heuristic. Finally, we gave a general overview of related work, providing details of implementation and main issues.

Chapter 3

Hyper-heuristic of Scalarizing Functions

Scalarizing functions have been successfully used by MOEAs for the fitness assignment process. In the literature, we can find algorithms that incorporate one or two fixed scalarizing functions, such as MOGLS [70] and MSOPS [65], respectively. Other approaches operate as frameworks, where the decision maker must select the scalarizing function that best suits his (her) needs, such as MOEA/D [148]. The popularity of scalarizing functions has to do with their low computational cost, their capability to generate (weakly) Pareto optimal solutions, and their effectiveness in solving many-objective optimization problems. Moreover, some scalarizing functions can capture the whole Pareto optimal front by varying either the weight vectors [33] or their parameter models, which control the curvature of their contour lines [99]. Nevertheless, recent studies indicate that the search behavior of MOEAs strongly depends on the choice of the scalarizing function [71, 72, 68]. In this chapter, we present a novel hyper-heuristic for continuous search spaces, which combines the strengths and compensates the weaknesses of different scalarizing functions. These heuristics have been proposed within the evolutionary multi-objective optimization and mathematical programming communities. Furthermore, the selection of heuristics is conducted through the s-energy indicator [54] (see page 17), which measures the even distribution of a set of points in k-dimensional manifolds.

3.1 Motivation

Despite the evident advantages of MOEAs relying on scalarizing functions, three crucial issues should be considered for a problem at hand: 1) the setting of the weight vectors, 2) the choice of an appropriate scalarizing function, and, if required, 3) the setting of the model parameters. About the first issue, it has been observed that a uniform distribution of the weight vectors in $[0, 1]^m$ does not necessarily imply that approximation sets will exhibit good diversity [24]. For this reason, several efforts have focused on the adaptation of the weights (see for example [33, 40]). Regarding

the second issue, some studies have exhaustively researched only the scalarizing functions WS, CHE, and PBI (see Table 2.3 on page 20), which present some limitations. For example, WS can only capture convex Pareto fronts [99], CHE loses diversity for more than two objectives [148], and, although PBI can find evenly distributed solutions from three objectives onwards [148, 28], its behavior depends on the parameter θ [147]. Moreover, recent experiments have shown that, for a given MOP, the search behavior of MOEAs strongly depends on the choice of these scalarizing functions [71, 72, 68]. Finally, concerning the third issue, certain scalarizing functions require the specification of some model parameters, which may be sensitive to the Pareto front shape. Thus, some attempts have emerged to adjust them dynamically [139, 147].

In this chapter, we focus on the second issue (the choice of an appropriate scalarizing function) and propose a hyper-heuristic [15], which combines the strengths and compensates the weaknesses of seven scalarizing functions. Our aims are to provide an algorithm that is more generally applicable than current implementations and also to release the decision maker from the difficult task of selecting an appropriate scalarizing function. The proposed hyper-heuristic can be seen as a search method for selecting low-level heuristics (i.e., scalarizing functions) to solve continuous MOPs.

3.2 Proposed Approach

Our proposal relies on an elitist genetic algorithm, which ranks the population by means of the $R2$ indicator. The original version was presented in 2013 under the name of Many-Objective Metaheuristic Based on the $R2$ Indicator (MOMBI) [57]. In Section 3.2.1, we describe, in order of importance, the main limitations of the original MOMBI, and propose an improved version, named MOMBI-II. Later, in Section 3.2.2 we introduce an extension that works as a hyper-heuristic, called MOMBI-III.

3.2.1 MOMBI framework

MOMBI does not require Pareto dominance. Instead, it uses for its selection mechanism the $R2$ indicator, which requires a set of utility functions, each one mapping an objective vector into a scalar value using a weight vector. The original MOMBI used CHE as a utility function, and for generating the weight vectors, the Simplex-Lattice Design method was adopted (see page 23). However, in some test instances for many-objective optimization, MOMBI experimented loss of diversity.

On the other hand, we observed that ASF is very similar to CHE with respect to their mathematical expression (see Table 2.3 on page 20). The main difference is that in CHE the weight component is being multiplied, whereas in ASF is being divided. Their contour lines are shown in Figure 3.1. It is interesting to observe that for CHE the contour lines are incongruent with respect to the weight vector. As a result the optimal point (represented as a black dot) is far away from the intersection between the weight vector and the Pareto optimal front.

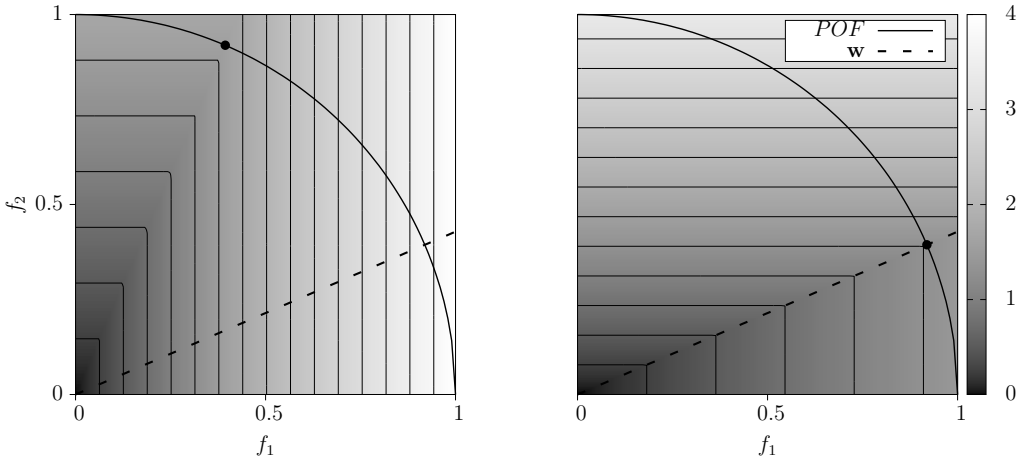


Figure 3.1: Contour lines of the CHE (left) and ASF (right) for the weight $\mathbf{w} = (0.7, 0.3)$.

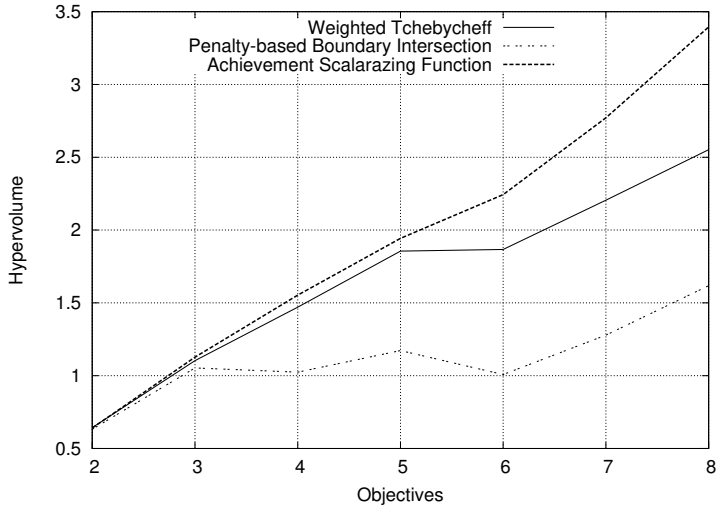


Figure 3.2: Average hypervolume for MOEA/D using three different scalarizing functions from 2 to 8 objectives on the DTLZ3 test problem.

When using the Simplex-Lattice Design method, the distribution of the vectors is equally spaced over a simplex, producing more points at the boundary than at the center. Therefore, this could explain the distinctive distribution generated by MOEAs based on CHE, such as MOEA/D [148] or MOMBI, where solutions are biased from the center of the Pareto front to a few boundary points, since this scalarizing function optimizes the opposite weight vectors.

In the following, we examine the scalability of CHE, ASF and PBI applied to the well-known MOEA/D on the DTLZ3 test problem. We performed 30 independent runs of each instance from 2 to 8 objectives, adopting the same parameter settings used in [57]. To assess performance, we employed the hypervolume indicator. In Figure 3.2, we present our experimental results. Here, for two and three objectives all the scalarizing functions gave very similar results. However, as the number of

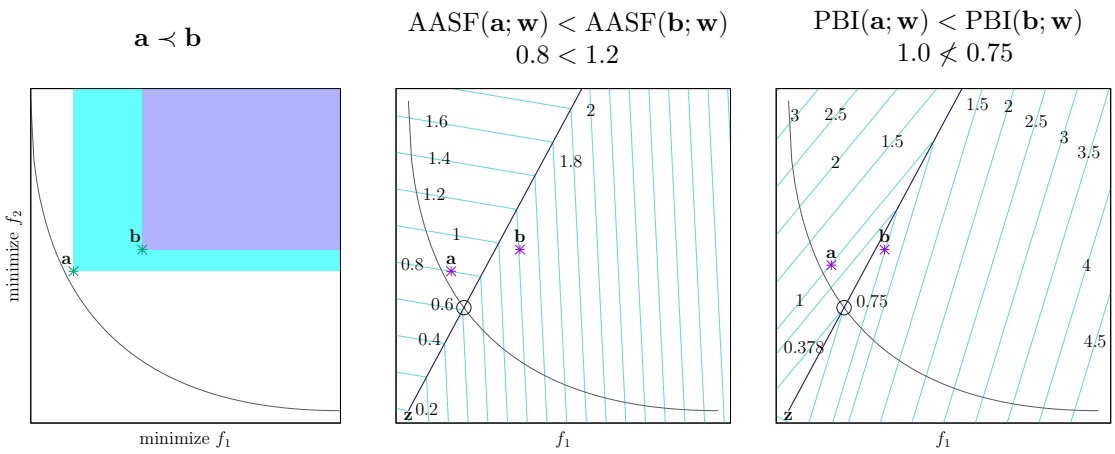


Figure 3.3: Illustration of a Pareto compliant scalarizing function (AASF), and a non-Pareto compliant one (PBI), considering the solutions $\mathbf{a}(0.138, 0.45)$, $\mathbf{b}(0.36, 0.52)$, and the weight vector $\mathbf{w}(0.35, 0.65)$.

objectives increases, the performance of PBI and CHE degrades. In contrast, the ASF metric obtained the maximum hypervolume for many-objective problems.

For this reason, in MOMBI-II we integrate ASF instead of CHE. Moreover, the reference points used during normalization are cleverly updated using statistical information about the population's proximity to the true Pareto optimal front, involving three additional parameters.

3.2.2 Extension to multiple scalarizing functions

The proposed hyper-heuristic is an extension of MOMBI-II (now named MOMBI-III), which allows the inclusion of more than one scalarizing function. The core idea is that each individual in the population minimizes a distinct scalarizing function, having its own weight vector. Thus, it can be considered as a method that encourages convergence through the optimization of the scalarizing function, while diversity is accomplished by two strategies: the weights and the heuristic selection.

The pool of heuristics consists of seven scalarizing functions $H = \{$WS, EWC, WPO, WN, CHE, ASF, AASF$\}$. Their mathematical expressions, features and model parameters appear in Table 2.3 (page 20), whereas their corresponding contour lines are shown in Figure 2.5 (page 19). The model parameters were established according to the values recommended in the literature. We selected this set of scalarizing function due to their compatibility with some form of Pareto dominance, a requirement that PBI does not meet. For a graphical counterexample see Figure 3.3. We also chose these scalarizing functions because their contour lines are very different, endowing our algorithm with an increased capacity to handle different Pareto front geometries. It is worth mentioning that in the original versions of WPO and WN, the component w_i is being multiplied by y_i [106]. We did not adopt this form since the search is driven to different regions of the objective space, as it occurred in [71, 58].

Heuristic selection is achieved through the use of s-energy indicator [54] (see page 17), which has been employed only for comparing MOEAs. As suggested by Giagkiozis et al. [40], s is set to $m-1$ since the Pareto front is at most an $(m-1)$-manifold. In order to know the individual contribution to the s-energy indicator, we define:

$$\Delta E_s(\mathbf{a}, A) := \frac{E_s(A) - E_s(A \setminus \{\mathbf{a}\})}{2}, \tag{3.1}$$

fulfilling that $E_s(A) = \sum_i \Delta E_s(\mathbf{a}_i, A)$.

The main loop of MOMBI-III is presented in Algorithm 1. First, the population is initialized uniformly at random. Thereafter, it is evaluated according to the MOP definition. At each iteration, new individuals are created from parents selected uniformly at random (lines 4 and 5). These individuals are evaluated and added to the overall population in lines 6 and 7. Next, the reference points are updated. Here, we used the ideal point for $\mathbf{z}^{\min}$ and the nadir point for $\mathbf{z}^{\max}$. The latter was calculated in the following way. First, we look for those individuals that minimized WPO and AASF, using as weights the unitary vectors parallel to the axes. The maximum values of each of the objective components corresponding to these individuals constitute the point $\mathbf{z}^{\max}$. We ensure that these individuals are different to each other and that $\mathbf{z}^{\max}$ encloses at least $|P|$ members of the population. If these requirements are not satisfied, we update $\mathbf{z}^{\max}$ with the worst objective values of the whole population. In line 9, the objective function values are normalized using the reference points. Here, we adopt the notation $q.\mathbf{y}$ to refer to the objective vector of an individual q. In the following steps, the population is ranked and reduced to the desired size.

Algorithm 2 scores the population according to the the pool of heuristics H. In line 1, the ranks are initialized and in lines 2 to 10 those individuals having the best value for each scalarizing function and weight vector obtain the first rank.

In Algorithm 3, the population is first sorted and partitioned in layers using the ranks. A layer contains individuals with the same rank. We notice that ties are broken using Pareto dominance. With this relation we can avoid weakly Pareto solutions in an effective manner. In lines 2 to 10, the individuals belonging to the worst ranks are removed from the population, either by discarding an entire layer or one individual at a time. In case that more than two individuals have the same rank, we remove the one with the highest contribution to the s-energy indicator. It is important to mention that for one set of high-fidelity objective-function evaluations, MOMBI-III can derive seven pieces of information with very low computational effort albeit these pieces may not be entirely independent.

3.3 Computational Complexity

The computational cost of MOMBI-II is polynomial on the population size and linear on the objectives. With a careful implementation of the s-energy indicator, the complexity of MOMBI-III is $\mathcal{O}(|P|^2 m + |H||W||P|(\log|P| + m))$. It is assumed that $|H| << |P|$ and $|W| = |P|$. Thus, the complexity is reduced to $\mathcal{O}(|P|^2(\log|P| + m))$.

Algorithm 1 Main loop of MOMBI-III

Input: MOP, stopping criterion, set of weight vectors W, set of heuristics H
Output: Final population P

Initialize population P at random
Evaluate MOP for each element $p \in P$
while the stopping criterion is not satisfied **do**
  Select random parents from P
  $P' \leftarrow$ Generate offspring using variation operators
  Evaluate MOP for each $p' \in P'$
  $Q \leftarrow P \bigcup P'$
  Update the reference points $\mathbf{z}^{\min}$ and $\mathbf{z}^{\max}$
  Normalize objective functions by setting
  $q.\mathbf{y} \leftarrow \frac{q.\mathbf{y}-\mathbf{z}^{\min}}{\mathbf{z}^{\max}-\mathbf{z}^{\min}}$, $\forall q \in Q$, where $q.\mathbf{y} \in \mathbb{R}^m$
  $R \leftarrow R2$ Ranking (Q, W, H)
  $P \leftarrow$ Reduce $(Q, R, |P|)$
return P

Algorithm 2 $R2$ Ranking

Input: Population set Q, set of weight vectors W, set of heuristics H
Output: Ranks R

$R[q] \leftarrow \infty \quad \forall q \in Q$
for all $h \in H$ **do**
  for all $\mathbf{w} \in W$ **do**
    for all $q \in Q$ **do**
      $q.\mu \leftarrow h(q.\mathbf{y}; \mathbf{w})$
    Sort Q w.r.t. the field μ in increasing order
    $rank \leftarrow 1$
    for all $q \in Q$ **do**
      $R[q] \leftarrow \min\{R[q], rank\}$
      $rank \leftarrow rank + 1$

3.4 Experimental Results

In this section, we investigate the effectiveness of our proposed hyper-heuristic, considering five instances of the ZDT test suite, seven instances of the DTLZ test problems, and all nine problems of the WFG test suite. In addition, for the last two benchmarks, we consider their inverted versions. All the mathematical definitions of these test problems can be found in Appendix B (page 151). The experiments were divided in three parts: comparison with single heuristic versions (Subsection 3.4.1), contrast with some state-of-the-art MOEAs (Subsection 3.4.2), behavior on inverted test problems (Subsection 3.4.3), and a case study for many-objectve optimization (Subsection 3.4.4). The first two experiments consider 2 and 3 objectives for the ZDT and DTLZ/WFG test problems, respectively; the third experiment contemplates 3 to

Algorithm 3 Reduce

Input: Population set Q, rank set R, desired size n
Output: Reduced population Q

$\{L_1, \ldots, L_k\} \leftarrow$ Sort the population in layers with respect to R
while $|Q| > n$ **do**
  if $|L_k| \leq |Q| - n$ **then** {Remove members of the k^{th} layer}
    $r \leftarrow L_k$
    $k \leftarrow k - 1$
  else
    Compute the contribution to the s-energy indicator
    $E[q.\mathbf{y}] \leftarrow \Delta E_{m-1}(q.\mathbf{y}, Q)$ for all $q \in \{L_1 \cup \ldots \cup L_k\}$
    $r \leftarrow \arg\max_{q \in L_k} E[q.\mathbf{y}]$
    $L_k \leftarrow L_k \setminus \{r\}$
  $Q \leftarrow Q \setminus \{r\}$
return Q

5 objectives; whereas the last experiment covers 4 to 10 objectives. In the following, we provide further details about these experiments.

The parameters for the WFG benchmark are provided in Table 3.1. For comparison purposes we selected the following algorithms: MOEA/D [148] (based on decomposition), NSGA-III [28] (based on reference points) and MOMBI-II [58] (based on the $R2$ indicator). All adopted the same parameter values of Table 3.1. We selected these optimizers since they have been reported to perform well in a wide range of problems. In addition, they share common features, such as the requirement of the set of weight vectors, which were generated using the Simplex-Lattice Design method, where we adopted a cardinality similar to the population size[1].

In the case of MOEA/D, its scalarizing function was CHE for two objectives and PBI with $\theta = 5$ for the remaining objectives [148, 28]. The population of MOEA/D was normalized during its evolution in the problems with different scale, such as the WFG and DTLZ7 instances, as suggested by its authors in the original paper. For MOMBI-II the scalarizing function was ASF, and its parameters were: record= 5, tolerance threshold= 1×10^{-3} and 0.5 for the variance threshold. The parameter models adopted for MOMBI-III and all its variants with single heuristics are established in Table 2.3 (page 20).

The variation operators were Polynomial-based mutation and Simulated Binary Crossover (SBX). For the mutation operator, its probability and distribution index were set to $1/n$ and 20, respectively. For the crossover operator, these parameters varied according to the number of objectives, for two objectives, they were set to 0.9 and 20, whereas for higher dimensionality were set to 1.0 and 30, respectively. The stopping criterion consisted of reaching a maximum number of MOP evaluations.

[1] For 5 and 6 objectives the set was pruned using a clustering technique, whereas for 8 and 9 objectives two layers were employed, discarding duplicated points.

Table 3.1: Parameters adopted in our experiments

Objectives (m)		2	3	4	5	6	7	8	9	10
Population size		100	136	166	180	200	210	230	250	266
Objective function evaluations ($\times 10^3$)		40	60	70	80	80	90	100	100	110
WFG	variables (n)	24	26	28	30	32	34	36	38	40
WFG	position-related parameters	2	2	3	4	5	6	7	8	9
Weight-vector partitions (H)		99	15	8	6	5	4	3,4	3,4	2,3
MOEA/D niche		20	27	33	36	40	42	46	50	53

Finally, for the performance assessment of the algorithms, we relied on the hypervolume indicator [152], which measures convergence and spread at the same time. However, this indicator favors non-linear Pareto fronts with clusters near the middle point (knee region). To compensate this situation, we also adopted the $(m-1)$-energy indicator (see expression 2.20 on page 17), which rewards even distributions. The reference points for the hypervolume indicator were $(4, 4, \ldots)$ for DTLZ3, $(2, 2, \ldots, 2, 8)$ for DTLZ7, $(3, 5, 7, \ldots)$ for WFG, $(2, 2, \ldots)$ for ZDT, DTLZ1,2, and DTLZ4-6; whereas for the inverted problems were $(0.1, 0.1, \ldots, -10)$ for DTLZ7, $(0.1, 0.1, \ldots)$ for DTLZ1-6^{-1} and WFG^{-1}. These points are slightly worse in all objectives than the nadir point.

In the following experiments, we always performed 30 independent runs for each MOEA and test problem. We applied the Wilcoxon rank sum test (one-tailed) to the mean of these indicators, in order to determine outperformance among the algorithms at the confidence interval of 99%. Moreover, due to multiple comparisons, this value was adjusted by the Bonferroni correction.

3.4.1 Single heuristics

Hyper-heuristics are meant to perform better than their constitutive heuristics in a wide range of problems [15]. Thus, motivated by this requirement, MOMBI-III was compared with the pool of heuristics contained within it. In this experiment, each scalarizing function was coupled independently to MOMBI-III. The results are presented in Tables 3.2, 3.3, 3.4, and 3.4. Regarding the hypervolume indicator, there is no doubt that MOMBI-III showed a clear advantage over these single heuristics, except for the problems ZDT1,6, DTLZ2,4, and WFG4,9, where our proposed method was outperformed by EWC, AASF and WPO. In spite of this, our MOMBI-III achieved the first and second places in 16 problems, out of 21. On the other hand, when evaluating with respect to the s-energy, our method outperformed the other variants in 20 cases. Only for ZDT2, WN ranked first, and in this case only the extreme points were found. This suggests that MOMBI-III effectively minimized this indicator.

Table 3.2: Median and standard deviation of the hypervolume and s-energy for single heuristics and MOMBI-III on the ZDT test problems. The two best values are shown in gray scale, where a darker tone corresponds to the best value. The outperformance relation among algorithms is presented, using a confidence level of 99% (for instance, WPO performs significantly better than WN on ZDT4).

Indicator	Problem	WS (1)	EWC (2)	WPO (3)	WN (4)	CHE (5)	ASF (6)	AASF (7)	MOMBI-III (8)
hypervolume	ZDT1	3.6539e+00 7.26e-4 3,4	3.6620e+00 8.08e-5 1,3,4,5,6,7,8	3.6333e+00 1.60e-2 4	3.0000e+00 1.78e-4 –	3.6614e+00 4.55e-5 1,3,4	3.6614e+00 4.55e-5 1,3,4	3.6614e+00 9.04e-5 1,3,4	3.6616e+00 7.40e-5 1,3,4,5,6,7
	ZDT2	3.0000e+00 1.71e-7 –	3.3286e+00 1.02e-4 1,3,4,5,6,7,8	3.3276e+00 1.20e-3 1,4	3.0000e+00 4.02e-7 –	3.3281e+00 1.60e-4 1,3,4	3.3281e+00 1.60e-4 1,3,4	3.3281e+00 1.65e-4 1,3,4	3.3283e+00 1.51e-4 1,3,4,5,6,7
	ZDT3	4.7070e+00 8.32e-2 4	4.8148e+00 5.20e-5 1,3,4,5,6,7	4.7813e+00 2.41e-1 4	4.0361e+00 2.52e-4 –	4.8140e+00 1.67e-4 1,3,4	4.8140e+00 1.67e-4 1,3,4	4.8141e+00 1.40e-4 1,3,4	4.8152e+00 4.91e-5 1,2,3,4,5,6,7
	ZDT4	3.6521e+00 1.72e-3 3,4	3.6590e+00 1.26e-2 1,3,4	3.5424e+00 1.10e-1 4	2.9967e+00 2.40e-3 –	3.6584e+00 1.93e-3 1,3,4	3.6584e+00 1.93e-3 1,3,4	3.6580e+00 3.6573e+00 1,3,4	3.42e-3 2.47e-3 1,3,4
	ZDT6	2.7741e+00 4.13e-4 4	3.0348e+00 1.25e-3 1,4,5,6,7,8	3.0347e+00 1.18e-3 1,4,5,6,7,8	2.7728e+00 1.14e-3 –	3.0312e+00 2.59e-3 1,4	3.0312e+00 2.59e-3 1,4	3.0309e+00 2.56e-3 1,4	3.0316e+00 2.20e-3 1,4
s-energy	ZDT1	1.366e+05 3.34e+04 –	5.741e+04 1.29e+02 1,3,5,6,7	1.275e+05 4.58e+03 –	4.472e+07 1.34e+08 –	6.486e+04 1.03e+01 1,3	6.486e+04 1.03e+01 1,3	6.486e+04 5.02e+01 1,3	5.639e+04 1.34e+02 1,2,3,5,6,7
	ZDT2	1.684e+08 9.93e+08 –	5.702e+04 1.13e+04 1,3	6.501e+04 1.05e+02 1	1.414e+00 3.24e+07 1	5.606e+04 2.22e+02 1,2,3	5.606e+04 2.22e+02 1,2,3	5.605e+04 3.14e+03 1,2,3	5.609e+04 1.10e+02 1,2,3
	ZDT3	1.931e+06 8.62e+05 –	6.505e+04 1.04e+04 1,3	1.203e+06 1.61e+05 1	5.737e+04 1.08e+08 –	4.938e+04 4.73e+03 1,2,3	4.938e+04 4.73e+03 1,2,3	4.884e+04 5.74e+03 1,2,3	4.404e+04 2.80e+02 1,2,3,5,6,7
	ZDT4	1.176e+05 2.11e+04 3,4	5.764e+04 2.25e+03 1,3,4,5,6,7	1.415e+05 6.14e+04 4	8.762e+06 3.16e+08 –	6.486e+04 7.17e+02 1,3,4	6.486e+04 7.17e+02 1,3,4	6.485e+04 5.80e+02 1,3,4	5.629e+04 2.69e+02 1,2,3,4,5,6,7
	ZDT6	2.930e+09 1.50e+16 –	7.231e+04 9.64e+03 1,3,4	8.987e+04 1.03e+04 1	9.295e+05 9.01e+10 1	7.012e+04 5.11e+02 1,2,3,4	7.012e+04 5.11e+02 1,2,3,4	7.007e+04 3.52e+03 1,2,3,4	6.936e+04 6.07e+02 1,2,3,4,5,6,7

Table 3.3: Median and standard deviation of the hypervolume for single heuristics and MOMBI-III on the DTLZ test problems.

Problem	WS (1)	EWC (2)	WPO (3)	WN (4)	CHE (5)	ASF (6)	AASF (7)	MOMBI-III (8)
DTLZ1	7.8809e+00 1.42e-3 4	7.9637e+00 2.81e-4 1,3,4,6,7	7.8612e+00 7.14e-2 –	7.8807e+00 2.17e-4 –	7.9691e+00 2.21e-2 1,2,3,4,6,7	7.9317e+00 2.50e-2 1,3,4	7.9364e+00 2.32e-2 1,3,4	7.9745e+00 1.14e-4 1,2,3,4,5,6,7
DTLZ2	7.0000e+00 3.43e-7 –	7.3928e+00 7.36e-4 1,4	7.4174e+00 1.37e-3 1,2,4,5	7.0000e+00 2.97e-7 –	7.3919e+00 4.29e-3 1,4	7.4190e+00 1.73e-3 1,2,3,4,5	7.4245e+00 3.68e-4 1,2,3,4,5,6,8	7.4236e+00 9.48e-4 1,2,3,4,5,6
DTLZ3	6.2978e+01 1.30e+0 –	6.3350e+01 1.44e-2 1,3,4	6.2969e+01 1.78e+0 –	6.2979e+01 2.30e-2 –	6.3384e+01 1.05e-1 1,2,3,4	6.3385e+01 2.47e-2 1,2,3,4	6.3400e+01 1.52e-2 1,2,3,4,5	6.3415e+01 1.36e-2 1,2,3,4,5,6
DTLZ4	7.0000e+00 2.10e-1 –	7.3931e+00 3.01e-1 1,4	7.4186e+00 2.55e-1 1,2,4,5	7.0000e+00 3.28e-1 –	7.3983e+00 3.00e-1 1,2,4	7.4234e+00 3.49e-1 1,2,3,4,5	7.4252e+00 3.08e-1 1,2,3,4,5,6,8	7.4247e+00 1.84e-1 1,2,3,4,5,6
DTLZ5	5.6716e+00 9.91e-3 –	5.8123e+00 4.24e-2 1,4	6.0001e+00 2.11e-2 1,2,4,5	5.6716e+00 1.77e-2 –	5.9656e+00 3.00e-2 1,2,4	6.0410e+00 5.87e-3 1,2,3,4,5	6.0412e+00 1.03e-2 1,2,3,4,5	6.1021e+00 1.19e-3 1,2,3,4,5,6,7
DTLZ6	5.4390e+00 9.07e-2 –	5.7566e+00 4.66e-2 1,4	5.7683e+00 8.14e-2 1,4	5.3947e+00 9.99e-2 –	5.7426e+00 7.22e-2 1,4	5.8099e+00 7.57e-2 1,4,5	5.8163e+00 8.00e-2 1,4,5	5.8509e+00 9.41e-2 1,2,4,5
DTLZ7	1.5994e+01 2.18e-3 4	1.7252e+01 1.08e-1 1,3,4	1.6218e+01 6.09e-2 1,4	1.5868e+01 3.30e-3 –	1.7230e+01 9.60e-2 1,3,4	1.7475e+01 2.81e-2 1,2,3,4,5	1.7482e+01 3.77e-2 1,2,3,4,5	1.7545e+01 1.02e-2 1,2,3,4,5,6,7

Table 3.4: Median and standard deviation of the s-energy indicator for single heuristics and MOMBI-III on the DTLZ test problems.

Problem	WS (1)	EWC (2)	WPO (3)	WN (4)	CHE (5)	ASF (6)	AASF (7)	MOMBI-III (8)
DTLZ1	1.103e+41 4.69e+60 –	6.064e+06 8.42e+10 1,4	3.319e+13 2.92e+17 1,4	3.302e+44 6.39e+61 –	1.526e+06 4.75e+10 1,2,3,4	8.164e+05 3.23e+14 1,3,4	8.153e+05 1.11e+12 1,2,3,4,5	7.981e+05 8.30e+03 1,2,3,4,5,6,7
DTLZ2	2.792e+23 2.48e+35 –	1.212e+06 2.57e+06 1,4	1.248e+05 1.04e+02 1,2,4,5	9.271e+22 5.04e+43 –	1.771e+07 2.05e+11 1,4	1.238e+05 2.83e+02 1,2,3,4,5	1.206e+05 7.42e+02 1,2,3,4,5,6	1.182e+05 6.26e+02 1,2,3,4,5,6,7
DTLZ3	1.526e+12 5.09e+34 –	6.366e+05 7.94e+08 1,3,4	1.873e+14 3.76e+23 –	2.113e+13 1.77e+33 –	3.949e+05 6.15e+09 1,3,4	1.364e+05 2.88e+04 1,2,3,4,5	1.429e+05 7.87e+04 1,2,3,4,5	1.167e+05 2.13e+03 1,2,3,4,5,6,7
DTLZ4	3.255e+23 8.42e+35 –	1.077e+06 2.91e+24 1,4	1.248e+05 3.72e+15 1,2,4,5	4.021e+25 3.53e+73 –	3.067e+05 1.49e+16 1,4	1.222e+05 3.07e+16 1,2,3,4,5	1.203e+05 1.36e+13 1,2,3,4,5,6	1.176e+05 1.66e+06 1,2,3,4,5,6,7
DTLZ5	5.059e+26 3.88e+62 –	1.433e+20 3.94e+23 1,4	1.037e+18 1.14e+20 1,2,4	1.086e+27 6.09e+57 –	5.090e+16 8.80e+19 1,2,4,7	2.547e+18 1.11e+22 1,2,4	6.107e+19 1.58e+21 1,4	7.102e+06 1.55e+14 1,2,3,4,5,6,7
DTLZ6	2.721e+39 1.45e+44 –	3.706e+07 1.80e+08 1,4	2.703e+07 1.19e+09 1,4	3.910e+27 4.31e+33 1	3.406e+06 9.04e+14 1,4,7	3.433e+07 1.52e+13 1,4	2.982e+09 1.01e+22 1,4	8.296e+05 1.09e+14 1,2,3,4,5,6,7
DTLZ7	2.209e+14 3.57e+18 –	8.873e+05 3.36e+08 1,3,4	7.196e+06 1.02e+08 1,4	1.148e+16 7.92e+17 –	6.833e+05 1.45e+10 1,3,4	1.111e+06 3.45e+06 1,3,4	1.257e+06 3.84e+06 1,3,4	9.824e+04 4.83e+03 1,2,3,4,5,6,7

Table 3.5: Median and standard deviation of the hypervolume indicator for single heuristics and MOMBI-III on the WFG test problems.

Problem	WS (1)	EWC (2)	WPO (3)	WN (4)	CHE (5)	ASF (6)	AASF (7)	MOMBI-III (8)
WFG1	5.4395e+01 1.32e+0 2,3,4	4.8913e+01 1.58e+0 3	4.4832e+01 1.80e+0 –	4.5190e+01 5.64e+0 –	5.2472e+01 1.63e+0 2,3,4	5.2447e+01 1.63e+0 2,3,4	5.2128e+01 1.75e+0 2,3,4	5.4921e+01 1.67e+0 2,3,4
WFG2	9.9789e+01 3.57e-1 3,4,5	1.0035e+02 1.40e-1 1,3,4,5	9.7306e+01 1.64e-1 4	7.1243e+01 1.21e+0 –	9.9489e+01 2.85e-1 3,4	1.0031e+02 1.22e-1 1,3,4,5	1.0034e+02 1.36e-1 1,3,4,5	1.0082e+02 1.04e-1 1,2,3,4,5,6,7
WFG3	5.4610e+01 4.01e-1 –	7.3941e+01 5.60e-1 1,3,4	7.2792e+01 8.79e-2 1,4	5.6352e+01 3.26e-1 1	7.4927e+01 2.44e-1 1,2,3,4	7.5219e+01 1.92e-1 1,2,3,4,5	7.5138e+01 2.00e-1 1,2,3,4	7.5220e+01 1.54e-1 1,2,3,4,5
WFG4	5.6991e+01 1.34e-2 –	7.5551e+01 9.92e-2 1,4	7.6980e+01 9.20e-2 1,2,4,5,6,7,8	5.6995e+01 9.27e-3 –	7.5590e+01 1.70e-1 1,4	7.6737e+01 8.43e-2 1,2,4,5,8	7.6743e+01 7.67e-2 1,2,4,5,8	7.6613e+01 8.90e-2 1,2,4,5
WFG5	5.3487e+01 5.78e-6 –	7.2536e+01 9.41e-2 1,4,5	7.3541e+01 4.35e-2 1,2,4,5	5.3487e+01 4.34e-6 –	7.2351e+01 1.53e-1 1,4	7.3688e+01 4.58e-2 1,2,3,4,5	7.3728e+01 6.67e-2 1,2,3,4,5	7.3823e+01 5.24e-2 1,2,3,4,5,6,7
WFG6	5.4326e+01 3.83e-1 –	7.2957e+01 2.84e-1 1,4	7.4509e+01 3.06e-1 1,2,4,5	5.4374e+01 3.99e-1 –	7.2872e+01 3.65e-1 1,4	7.4153e+01 3.93e-1 1,2,4,5	7.4305e+01 3.60e-1 1,2,4,5	7.4245e+01 2.87e-1 1,2,4,5
WFG7	5.7011e+01 4.57e-3 –	7.5688e+01 4.23e-1 1,4	7.6816e+01 5.35e-2 1,2,4,5	5.7012e+01 3.36e-3 –	7.5529e+01 2.59e-1 1,4	7.6852e+01 7.72e-2 1,2,4,5	7.6900e+01 6.40e-2 1,2,3,4,5	7.7048e+01 4.94e-2 1,2,3,4,5,6,7
WFG8	5.2754e+01 7.05e-1 –	7.1874e+01 3.21e-1 1,4	7.2693e+01 3.06e-1 1,2,4,5	5.3782e+01 6.93e-1 –	7.1874e+01 1.88e-1 1,4	7.3041e+01 1.80e-1 1,2,3,4,5	7.2915e+01 1.71e-1 1,2,3,4,5	7.2842e+01 2.28e-1 1,2,4,5
WFG9	5.4922e+01 1.76e+0 –	7.3757e+01 2.56e-1 1,4	7.6333e+01 1.34e+0 1,2,4,5,6,7,8	5.5106e+01 3.39e+0 –	7.3891e+01 1.28e+0 1,4	7.5062e+01 1.03e+0 1,2,4,5	7.5058e+01 1.12e+0 1,2,4,5	7.5127e+01 2.18e-1 1,2,4,5

Table 3.6: Median and standard deviation of the s-energy indicator for single heuristics and MOMBI-III on the WFG test problems.

Problem	WS (1)	EWC (2)	WPO (3)	WN (4)	CHE (5)	ASF (6)	AASF (7)	MOMBI-III (8)
WFG1	1.451e+09 1.35e+11 –	1.355e+05 4.40e+06 1,3,4	1.582e+07 8.35e+10 1,4	8.852e+09 6.25e+16 –	1.945e+05 3.17e+05 1,3,4	8.232e+04 1.99e+05 1,2,3,4,5	7.936e+04 1.58e+06 1,2,3,4,5	4.654e+04 6.16e+03 1,2,3,4,5,6,7
WFG2	4.249e+07 8.85e+09 4	6.491e+04 3.35e+05 1,3,4	2.765e+06 1.25e+07 1,4	2.854e+11 9.46e+12 –	1.191e+05 1.24e+07 1,3,4	2.845e+04 8.39e+05 1,2,3,4,5	2.934e+04 8.69e+04 1,2,3,4,5	2.064e+04 1.38e+03 1,2,3,4,5,6,7
WFG3	3.107e+13 2.45e+18 –	1.800e+06 3.20e+09 1,4	2.885e+06 5.79e+09 1,4	3.518e+13 8.16e+15 –	1.830e+05 5.31e+06 1,2,3,4,6,7	9.107e+05 6.35e+06 1,3,4	7.468e+05 8.65e+06 1,3,4	3.911e+04 2.52e+03 1,2,3,4,5,6,7
WFG4	1.312e+10 3.10e+11 –	1.500e+04 7.94e+04 1,4,5	9.368e+03 1.18e+02 1,2,4,5	4.125e+09 2.20e+13 –	3.636e+05 2.38e+06 1,4	9.360e+03 9.61e+02 1,2,4,5	9.438e+03 2.51e+03 1,2,4,5	8.353e+03 1.14e+02 1,2,3,4,5,6,7
WFG5	9.993e+16 2.50e+19 –	1.926e+04 3.78e+04 1,4,5	9.202e+03 3.00e+01 1,2,4,5	2.037e+16 1.47e+21 –	9.692e+06 4.80e+07 1,4	9.111e+03 1.13e+03 1,2,3,4,5	9.047e+03 1.32e+02 1,2,3,4,5	8.419e+03 9.45e+01 1,2,3,4,5,6,7
WFG6	2.161e+10 1.29e+12 –	1.562e+04 3.91e+05 4,5	9.186e+03 6.60e+01 2,4,5	2.731e+10 1.02e+13 –	2.838e+05 3.40e+07 4	9.216e+03 9.11e+02 2,4,5	9.245e+03 9.29e+02 2,4,5	8.354e+03 1.01e+02 2,3,4,5,6,7
WFG7	1.235e+11 2.32e+12 –	2.383e+04 1.06e+06 1,4,5	9.439e+03 4.35e+01 1,2,4,5	3.176e+11 8.53e+13 –	5.113e+05 9.90e+06 1,4	9.324e+03 6.27e+02 1,2,3,4,5	9.301e+03 2.57e+02 1,2,3,4,5	8.415e+03 1.17e+02 1,2,3,4,5,6,7
WFG8	3.879e+09 6.28e+12 –	1.928e+04 1.54e+05 1,3,4,5	2.482e+06 8.80e+06 1,4	9.632e+08 1.51e+11 –	3.408e+05 9.34e+05 1,3,4	3.435e+04 2.19e+05 1,3,4,5	5.806e+04 2.13e+05 1,3,4,5	8.314e+03 1.23e+02 1,2,3,4,5,6,7
WFG9	6.356e+08 5.32e+13 –	2.246e+04 4.22e+04 1,4,5	9.663e+03 2.72e+02 1,2,4,5,6,7	3.228e+09 8.34e+12 –	1.047e+05 1.43e+07 1,4	1.371e+04 5.40e+04 1,4,5	1.196e+04 6.04e+04 1,2,4,5	8.565e+03 9.60e+01 1,2,3,4,5,6,7

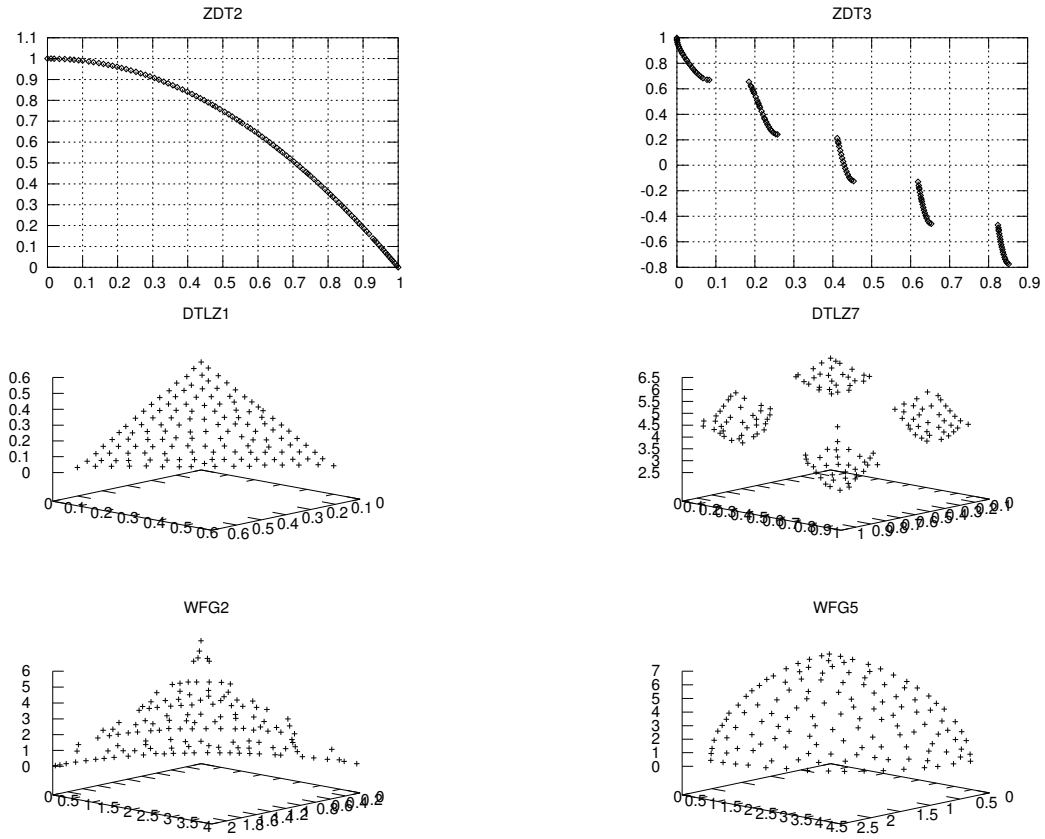


Figure 3.4: Pareto fronts produced by MOMBI-III.

In Figure 3.4, we show some approximations to the Pareto front produced by MOMBI-III, corresponding to the median of the hypervolume indicator.

3.4.2 State-of-the-art algorithms

In this section, we compared MOMBI-III with respect to the well-known MOEA/D and NSGA-III. Additionally, we considered MOMBI-II, since our proposed method improves it. Experimental results are shown in Tables 3.7 and 3.8. In the case of the hypervolume indicator, we can observe again an overwhelming outperformance of our proposed method, whose scores were in the top places on 17 instances out of 21. It was only outperformed by MOEA/D and NSGA-III on the concave problems ZDT6 and DTLZ2. MOMBI-III won over its predecessor MOMBI-II in 13 instances, providing solid evidence of its superiority. Concerning the s-energy indicator, MOMBI-III performed better in 14 instances, being outperformed on ZDT2, DTLZ1, and WFG4,6-8 by MOEA/D and NSGA-III. Lastly, our proposed approach was better than MOMBI-II in almost all the test problems adopted.

Table 3.7: Median and standard deviation of the hypervolume indicator for the compared MOEAs and MOMBI-III.

Problem	MOEA/D (1)	NSGA-III (2)	MOMBI-II (3)	MOMBI-III (4)
ZDT1	3.660e+00 1.58e-3 –	3.661e+00 1.78e-4 1	3.661e+00 8.04e-5 1	3.662e+00 7.59e-5 1,2,3
ZDT2	3.326e+00 1.28e-3 –	3.328e+00 2.27e-4 1	3.328e+00 1.22e-4 1	3.328e+00 1.14e-4 1,2,3
ZDT3	4.811e+00 2.95e-3 –	4.813e+00 3.36e-4 1	4.814e+00 8.81e-5 1,2	4.815e+00 6.16e-2 1,2,3
ZDT4	3.649e+00 5.11e-3 –	3.658e+00 7.45e-3 1	3.658e+00 4.11e-3 1	3.659e+00 1.64e-3 1
ZDT6	3.036e+00 1.26e-3 2,3,4	3.024e+00 4.58e-3 –	3.031e+00 2.28e-3 2	3.030e+00 2.55e-3 2
DTLZ1	7.975e+00 1.63e-4 3	7.975e+00 5.56e-4 3	7.937e+00 4.78e-3 –	7.975e+00 5.54e-5 3
DTLZ2	7.426e+00 2.34e-5 2,3,4	7.425e+00 4.06e-4 3,4	7.376e+00 7.14e-3 –	7.423e+00 9.85e-4 3
DTLZ3	6.339e+01 1.87e-2 3	6.340e+01 2.83e-2 3	6.336e+01 1.81e-2 –	6.341e+01 6.25e-3 1,2,3
DTLZ4	7.426e+00 1.05e+0 –	7.425e+00 4.37e-1 3	7.407e+00 4.91e-3 –	7.424e+00 1.84e-1 3
DTLZ5	6.050e+00 2.15e-4 2,3	5.954e+00 2.18e-1 –	6.015e+00 3.26e-3 –	6.103e+00 1.09e-4 1,2,3
DTLZ6	5.821e+00 7.95e-2 2,3	5.444e+00 1.15e-1 –	5.748e+00 6.70e-2 2	5.877e+00 7.63e-2 2,3
DTLZ7	9.729e+00 2.61e-2 –	1.739e+01 2.88e-2 1,3	1.736e+01 1.15e-2 1	1.754e+01 1.24e-2 1,2,3
WFG1	5.305e+01 1.45e+0 2	4.907e+01 1.59e+0 –	5.443e+01 1.79e+0 2	5.492e+01 1.67e+0 2
WFG2	9.666e+01 1.16e+0 –	1.003e+02 1.80e-1 1,3	1.001e+02 1.61e-1 1	1.008e+02 1.04e-1 1,2,3
WFG3	7.283e+01 6.74e-1 –	7.408e+01 1.53e-1 1	7.505e+01 1.47e-1 1,2	7.522e+01 1.54e-1 1,2,3
WFG4	7.382e+01 4.23e-1 –	7.656e+01 1.04e-1 1	7.668e+01 9.41e-2 1,2	7.661e+01 8.90e-2 1
WFG5	7.134e+01 5.35e-1 –	7.373e+01 8.87e-2 1,3	7.353e+01 8.09e-2 1	7.383e+01 4.10e-2 1,2,3
WFG6	7.153e+01 6.24e-1 –	7.412e+01 2.69e-1 1	7.401e+01 3.76e-1 1	7.422e+01 3.13e-1 1
WFG7	7.308e+01 8.38e-1 –	7.685e+01 7.76e-2 1	7.682e+01 8.26e-2 1	7.700e+01 5.64e-2 1,2,3
WFG8	6.945e+01 9.24e-1 –	7.285e+01 2.67e-1 1	7.266e+01 2.02e-1 1	7.293e+01 2.29e-1 1,3
WFG9	6.821e+01 1.79e+0 –	7.392e+01 9.19e-1 1	7.489e+01 1.10e+0 1,2	7.513e+01 2.70e-1 1,2

Table 3.8: Median and standard deviation of the s-energy indicator for the compared MOEAs and MOMBI-III.

Problem	MOEA/D (1)	NSGA-III (2)	MOMBI-II (3)	MOMBI-III (4)
ZDT1	6.481e+04 5.26e+01 2,3	6.487e+04 1.45e+02 –	6.486e+04 1.43e+02 –	5.693e+04 1.88e+02 1,2,3
ZDT2	5.606e+04 1.85e+02 4	5.606e+04 3.70e+03 4	5.606e+04 3.45e+02 4	5.612e+04 1.02e+02 –
ZDT3	1.691e+06 5.51e+07 3	1.315e+05 5.69e+04 1,3	2.080e+07 1.28e+07 –	4.401e+04 2.14e+03 1,2,3
ZDT4	6.450e+04 2.29e+02 2,3	6.484e+04 5.55e+03 –	6.489e+04 4.81e+03 –	5.645e+04 3.04e+02 1,2,3
ZDT6	7.073e+04 2.30e+02 –	7.001e+04 1.13e+04 –	7.070e+04 7.50e+05 –	6.936e+04 4.38e+02 1,2,3
DTLZ1	7.820e+05 2.25e+03 2,3,4	7.842e+05 1.12e+08 3,4	8.150e+05 1.43e+03 –	8.011e+05 4.57e+03 3
DTLZ2	1.191e+05 5.54e+00 2,3	1.192e+05 2.14e+02 3	1.250e+05 3.36e+02 –	1.182e+05 5.05e+02 1,2,3
DTLZ3	1.163e+05 3.65e+07 2,3	1.355e+05 9.25e+06 –	1.391e+05 3.62e+10 –	1.164e+05 1.51e+03 2,3
DTLZ4	1.191e+05 - –	1.192e+05 9.79e+12 3	1.236e+05 7.82e+02 –	1.176e+05 1.68e+06 1,2,3
DTLZ5	6.376e+15 8.00e+15 3	8.663e+14 1.89e+37 –	6.450e+16 1.41e+18 –	3.041e+06 1.52e+05 1,2,3
DTLZ6	1.214e+09 5.23e+10 2,3	3.282e+11 1.64e+21 –	4.892e+13 4.33e+15 –	4.489e+05 1.78e+05 1,2,3
DTLZ7	3.073e+10 1.73e+18 –	1.731e+06 1.01e+07 1,3	5.544e+11 1.30e+12 –	5.075e+04 2.22e+03 1,2,3
WFG1	8.868e+09 7.33e+11 –	1.566e+05 5.71e+06 1,3	3.479e+08 6.75e+10 1	4.654e+04 6.16e+03 1,2,3
WFG2	2.496e+04 1.21e+08 3	2.004e+04 8.16e+04 1,3	9.639e+08 3.12e+11 –	2.064e+04 1.38e+03 1,3
WFG3	2.202e+09 1.26e+10 –	2.662e+06 2.47e+08 1,3	8.204e+08 1.89e+11 –	3.911e+04 2.52e+03 1,2,3
WFG4	8.175e+03 1.28e+02 2,3,4	8.931e+03 4.38e+01 3	9.313e+03 1.00e+05 –	8.353e+03 1.14e+02 2,3
WFG5	1.659e+05 8.21e+07 –	8.852e+03 2.70e+01 1,3	9.156e+03 3.99e+07 1	8.398e+03 1.25e+02 1,2,3
WFG6	7.663e+03 1.41e+02 2,3,4	8.937e+03 7.02e+04 3	1.010e+04 6.12e+05 –	8.397e+03 1.29e+02 2,3
WFG7	7.665e+03 1.40e+02 2,3,4	8.858e+03 1.20e+03 3	9.173e+03 1.64e+06 –	8.447e+03 1.18e+02 2,3
WFG8	8.032e+03 2.16e+02 2,3,4	2.182e+04 5.33e+05 3	3.886e+08 2.05e+10 –	8.293e+03 1.42e+02 2,3
WFG9	2.931e+05 3.65e+15 –	1.083e+04 3.91e+04 1,3	5.015e+05 1.12e+07 –	4.314e+03 4.83e+01 1,2,3

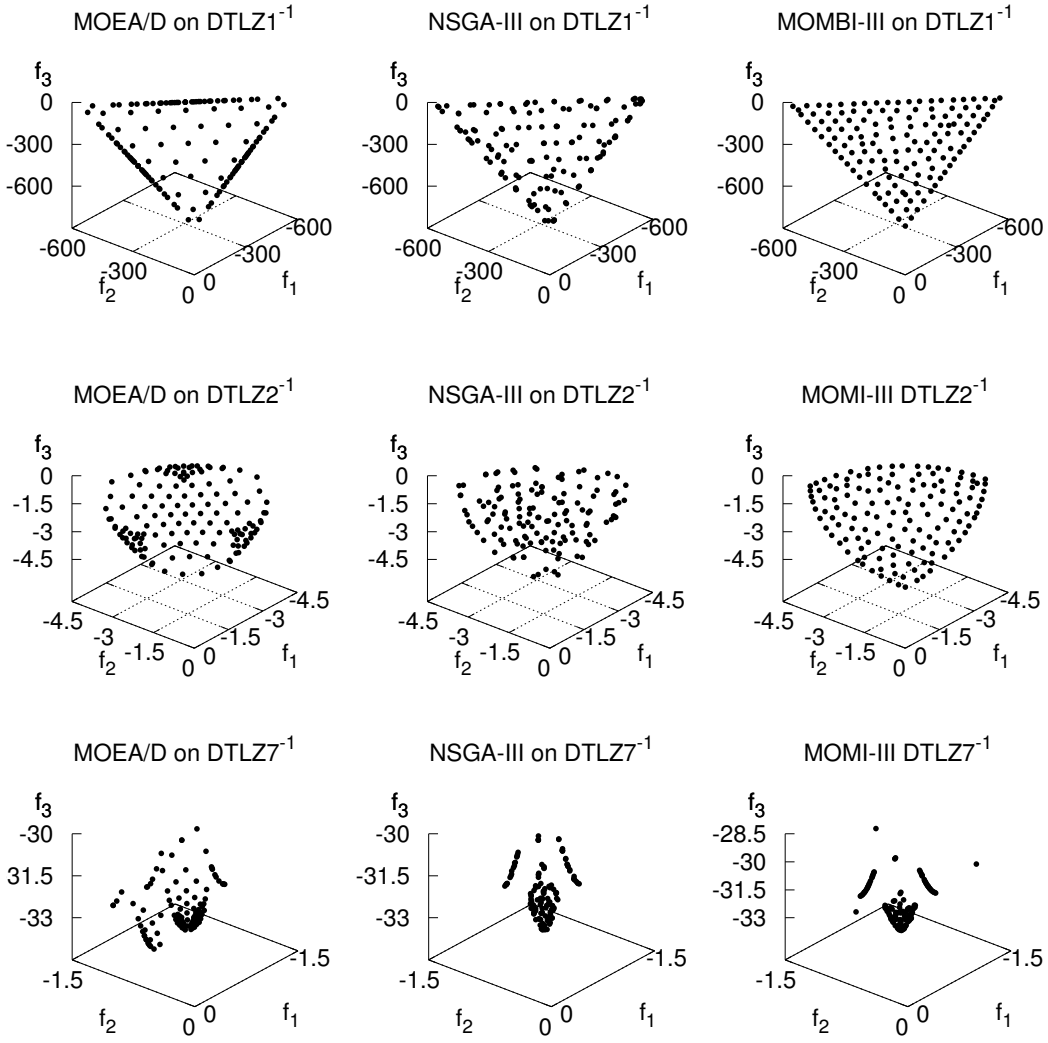


Figure 3.5: Pareto fronts produced by the MOEAs on DTLZ^{-1} test problems.

3.4.3 Inverted test problems

According to a recent study [73], optimizers that rely on weight vectors or reference points, such as MOEA/D or NSGA-III, deteriorate their performance on inverted benchmarks, such as DTLZ^{-1} and WFG^{-1}. Therefore, in this section, we investigate the behavior of MOMBI-III in such problems. The results are presented in Tables 3.9, 3.10, and 3.11. For three and four objectives, there is no doubt that MOMBI-III showed a clear advantage, significantly outperforming MOEA/D and NSGA-III 15 times, out of 16. Only in WFG1^{-1} there was a tie with MOEA/D. On the other hand, NSGA-III ranked second in most DTLZ^{-1} test problems, whereas MOEA/D ranked second in the WFG^{-1} benchmark. For five objectives, MOMBI-III ranked first in all problems, followed by MOEA/D. Finally, in Figures 3.5 and 3.6, we show some examples of Pareto fronts generated by MOMBI-III. As can be observed, our proposal produced more uniform distributions than the compared algorithms.

Table 3.9: Median and standard deviation of the hypervolume indicator on inverted test problems for **three objectives.**

Problem	MOEA/D (1)	NSGA-III (2)	MOMBI-III (3)
DTLZ1^{-1}	1.71e+07 5.72e+5 –	2.02e+07 1.62e+5 1	2.22e+07 5.17e+4 1,2
DTLZ2^{-1}	2.23e+01 7.25e-2 –	2.24e+01 1.03e-1 1	2.28e+01 2.64e-2 1,2
DTLZ3^{-1}	4.74e+09 6.03e+7 –	4.85e+09 3.71e+7 1	4.96e+09 9.24e+6 1,2
DTLZ4^{-1}	2.23e+01 3.82e+0 –	2.23e+01 9.40e-2 –	2.28e+01 3.37e-2 1,2
DTLZ5^{-1}	2.26e+01 4.90e-2 –	2.25e+01 6.40e-2 –	2.31e+01 2.60e-2 1,2
DTLZ6^{-1}	6.33e+02 1.73e+0 –	6.41e+02 1.93e+0 1	6.50e+02 5.99e-1 1,2
DTLZ7^{-1}	2.70e+01 1.92e-3 –	2.70e+01 4.72e-3 1	2.70e+01 1.57e-3 1,2
WFG1^{-1}	2.50e+01 4.92e+0 2	1.21e+01 4.69e+0 –	2.49e+01 4.34e+0 2
WFG2^{-1}	6.11e+01 7.86e-2 2	6.06e+01 9.56e-2 –	6.15e+01 4.66e-2 1,2
WFG3^{-1}	4.37e+01 1.30e-1 2	4.27e+01 3.58e-1 –	4.49e+01 8.84e-2 1,2
WFG4^{-1}	7.38e+01 1.40e-1 2	7.24e+01 5.00e-1 –	7.49e+01 5.06e-2 1,2
WFG5^{-1}	7.39e+01 8.55e-2 2	7.28e+01 2.27e-1 –	7.46e+01 5.43e-2 1,2
WFG6^{-1}	7.41e+01 7.43e-2 2	7.35e+01 2.32e-1 –	7.49e+01 7.18e-2 1,2
WFG7^{-1}	7.37e+01 1.64e-1 2	7.19e+01 3.24e-1 –	7.45e+01 1.04e-1 1,2
WFG8^{-1}	7.42e+01 7.25e-2 2	7.36e+01 2.82e-1 –	7.49e+01 9.12e-2 1,2
WFG9^{-1}	7.30e+01 3.08e-1 2	7.26e+01 3.41e-1 –	7.44e+01 1.61e-1 1,2

Table 3.10: Median and standard deviation of the hypervolume indicator on inverted test problems for **four objectives.**

Problem	MOEA/D (1)	NSGA-III (2)	MOMBI-III (3)
DTLZ1^{-1}	3.68e+08 5.88e+5 –	8.34e+08 1.01e+8 1	1.35e+09 4.42e+7 1,2
DTLZ2^{-1}	3.09e+01 3.73e-1 –	3.23e+01 7.38e-1 1	3.52e+01 2.78e-1 1,2
DTLZ3^{-1}	3.42e+12 1.30e+1 –	3.88e+12 1.33e+1 1	4.30e+12 4.51e+0 1,2
DTLZ4^{-1}	3.03e+01 1.25e+1 –	3.13e+01 7.00e-1 1	3.52e+01 2.44e-1 1,2
DTLZ5^{-1}	3.57e+01 3.07e-1 2	3.51e+01 4.74e-1 –	3.95e+01 1.09e-1 1,2
DTLZ6^{-1}	2.67e+03 1.72e+1 –	2.82e+03 5.57e+1 1	3.07e+03 1.60e+1 1,2
DTLZ7^{-1}	4.37e+01 4.61e+0 –	4.37e+01 7.90e-2 –	4.38e+01 1.45e-2 1,2
WFG1^{-1}	5.60e+01 9.80e+0 2	1.21e+01 2.81e+0 –	5.36e+01 1.02e+1 2
WFG2^{-1}	1.92e+02 2.59e-1 2	1.66e+02 1.01e+1 –	2.01e+02 5.95e-1 1,2
WFG3^{-1}	1.44e+02 1.98e-1 2	1.26e+02 4.40e+0 –	1.55e+02 6.33e-1 1,2
WFG4^{-1}	3.87e+02 1.95e+0 2	3.40e+02 9.93e+0 –	4.05e+02 1.78e+0 1,2
WFG5^{-1}	3.88e+02 1.36e+0 2	3.61e+02 8.60e+0 –	4.03e+02 6.54e-1 1,2
WFG6^{-1}	3.91e+02 7.46e-1 2	3.71e+02 7.34e+0 –	4.06e+02 1.02e+0 1,2
WFG7^{-1}	3.87e+02 1.67e+0 2	3.50e+02 9.88e+0 –	4.02e+02 1.62e+0 1,2
WFG8^{-1}	3.91e+02 8.46e-1 2	3.76e+02 8.34e+0 –	4.06e+02 1.00e+0 1,2
WFG9^{-1}	3.83e+02 3.24e+0 2	3.69e+02 7.60e+0 –	4.05e+02 1.85e+0 1,2

Table 3.11: Median and standard deviation of the hypervolume indicator on inverted test problems for **five objectives.**

Problem	MOEA/D (1)	NSGA-III (2)	MOMBI-III (3)
DTLZ1^{-1}	5.25e+08 3.81e+7 –	9.02e+09 2.83e+9 1	2.11e+10 4.14e+9 1,2
DTLZ2^{-1}	3.45e+01 1.74e-1 2	2.35e+01 2.24e+0 –	3.53e+01 9.61e-1 1,2
DTLZ3^{-1}	2.14e+15 5.53e+3 2	1.23e+15 2.07e+4 –	2.19e+15 9.49e+3 1,2
DTLZ4^{-1}	3.43e+01 1.30e+1 2	2.10e+01 3.81e+0 –	3.51e+01 9.13e-1 1,2
DTLZ5^{-1}	4.66e+01 1.42e-1 2	3.52e+01 1.90e+0 –	5.11e+01 4.27e-1 1,2
DTLZ6^{-1}	9.10e+03 4.83e+1 2	6.53e+03 5.80e+2 –	9.63e+03 2.75e+2 1,2
DTLZ7^{-1}	2.41e+01 4.81e+0 –	6.28e+01 2.10e-1 1	6.37e+01 5.38e-2 1,2
WFG1^{-1}	7.96e+01 1.17e+1 2	1.59e+01 1.98e+0 –	1.05e+02 2.22e+1 1,2
WFG2^{-1}	3.86e+02 2.63e-1 2	3.14e+02 3.80e+1 –	5.39e+02 2.96e+0 1,2
WFG3^{-1}	4.10e+02 4.08e-1 2	2.81e+02 1.88e+1 –	4.65e+02 6.33e+0 1,2
WFG4^{-1}	1.90e+03 5.74e+0 2	1.41e+03 8.87e+1 –	2.01e+03 2.00e+1 1,2
WFG5^{-1}	1.89e+03 3.30e+0 2	1.59e+03 9.20e+1 –	2.01e+03 1.40e+1 1,2
WFG6^{-1}	1.89e+03 2.73e+0 2	1.64e+03 8.58e+1 –	2.02e+03 1.51e+1 1,2
WFG7^{-1}	1.89e+03 2.72e+1 2	1.41e+03 8.46e+1 –	2.00e+03 1.52e+1 1,2
WFG8^{-1}	1.89e+03 2.31e+0 2	1.67e+03 9.76e+1 –	2.01e+03 1.31e+1 1,2
WFG9^{-1}	1.87e+03 1.62e+1 2	1.66e+03 7.57e+1 –	2.06e+03 1.35e+1 1,2

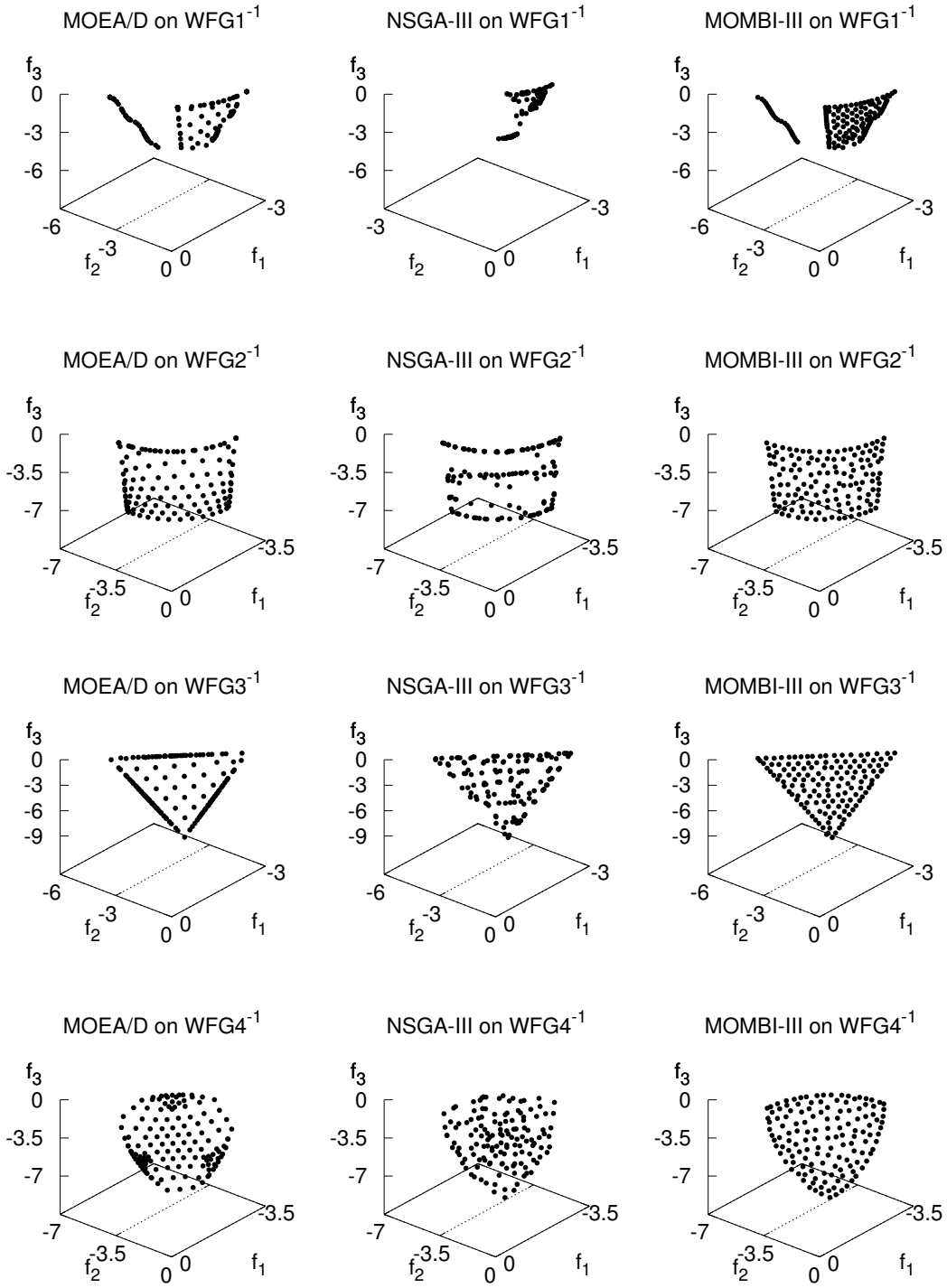


Figure 3.6: Pareto fronts produced by the MOEAs on WFG^{-1} test problems.

3.4.4 Many-objective optimization problems

In this experiment, we investigated the behavior of MOMBI-III in many-objective instances of DTLZ1. For this purpose, we performed a comparative study with the

Table 3.12: Median and standard deviation of the hypervolume indicator on many-objective instances of DTLZ1.

m	MOEA/D (1)	NSGA-III (2)	MOMBI-II (3)	MOMBI-III (4)
4	1.5995e+01 1.23e-4 3	1.5995e+01 8.66e-5 1,3	1.5945e+01 8.54e-3 –	1.5995e+01 4.35e-5 1,3
5	3.1999e+01 9.18e-5 3	3.1999e+01 6.97e-5 1,3	3.1930e+01 1.68e-2 –	3.1999e+01 3.65e-5 1,2,3
6	6.3998e+01 4.27e-4 3	6.4000e+01 9.66e-5 1,3	6.3922e+01 2.08e-2 –	6.4000e+01 7.23e-5 1,2,3
7	1.2800e+02 8.25e-4 3	1.2800e+02 1.36e-2 1,3	1.2784e+02 5.71e-2 –	1.2800e+02 1.17e-4 1,3
8	2.5599e+02 4.47e-3 3	2.5600e+02 2.00e-2 1,3	2.5577e+02 8.23e-2 –	2.5600e+02 8.56e-4 1,3
9	5.1196e+02 2.69e-2 3	5.1199e+02 7.76e-2 1,3	5.1164e+02 1.78e-1 –	5.1200e+02 1.36e-3 1,2,3
10	1.0238e+03 7.78e-2 3	1.0240e+03 3.41e-2 1,3	1.0232e+03 2.63e-1 –	1.0240e+03 8.59e-4 1,2,3

same algorithms of the previous experiment. The results of the hypervolume indicator are presented in Table 3.12. In this case, MOMBI-III outperformed MOEA/D and MOMBI-II in all instances and performed slightly better than NSGA-III.

3.5 Summary

In this chapter, we presented for the first time, a Hyper-Heuristic of Scalarizing Functions (MOMBI-III) for solving continuous multi-objective optimization problems, which are transformed into single-objective ones. The adopted set of scalarizing functions has several advantages, from which the most relevant are its low computational cost and compatibility with Pareto dominance. Although MOMBI-III can incorporate scalarizing functions that are incompatible with any form of Pareto optimality (e.g., PBI), an extra effort is required at each iteration to filter dominated solutions. Furthermore, MOMBI-III incorporates the s-energy indicator for generating even distributions in objective space. Our experimental results showed that MOMBI-III significantly outperformed single heuristics as well as advanced algorithms, such as NSGA-III and MOEA/D in the majority of the instances of the ZDT, DTLZ, DTLZ^{-1}, WFG and WFG^{-1} test suites. Although MOMBI-III relies on weight vectors, it does not suffer from the overspecialization problem that MOEA/D and NSGA-III have. Furthermore, our proposal showed promise in solving many-objective problems.

Chapter 4

A Density Estimator based on Parallel Coordinates

The Parallel Coordinates [67], or Value-Path [102], is a widely used visualization technique for multivariate data, allowing to identify patterns, clusters and correlation between variables [55]. In the context of multi-objective optimization, Parallel Coordinates has been frequently used for visualizing approximation sets, specially in high-dimensional spaces. Its applicability varies from the identification of differences and similarities between alternatives, guidance to the decision-maker in selecting solutions, monitoring the progress of an optimization run, assessment of the relative performance of different algorithms [102, 126] to, more recently, guidance during the search in bio-inspired meta-heuristics [62, 63].

In the following, we propose a density estimator based on this visualization tool, which can be seen as an off-line selection hyper-heuristic. To validate its efficiency, we first couple it with a Pareto-based MOEA, named Multi-objective Optimizer based on Value Path (MOVAP). Then, in Chapter 5, the use of this density estimator is extended to an archiving technique for an island-based MOEA.

The remainder of this chapter is organized as follows. In Section 4.1, we provide the motivation behind our work and a brief overview of the related work. Section 4.2 is devoted to the description of our proposed algorithm. In Section 4.3, the computational complexity of MOVAP is derived. In Section 4.4, we present a comparative study using the ZDT and the WFG test suites. A discussion of the experimental results is presented in Section 4.5. Section 4.6 provides a summary of this chapter.

4.1 Motivation

Value-Path is built in the 2-dimensional plane, where m copies of the real line $\mathbb{R}$ are placed perpendicular to the x-axis and a point in $\mathbb{R}^m$ is represented by a connected series of line segments (known as *polygonal line* or *polyline*) with vertices on the parallel axes. In Figure 4.1, we show an example of this graph with its corresponding Pareto front, composed of eleven non-dominated solutions. The basic idea of our

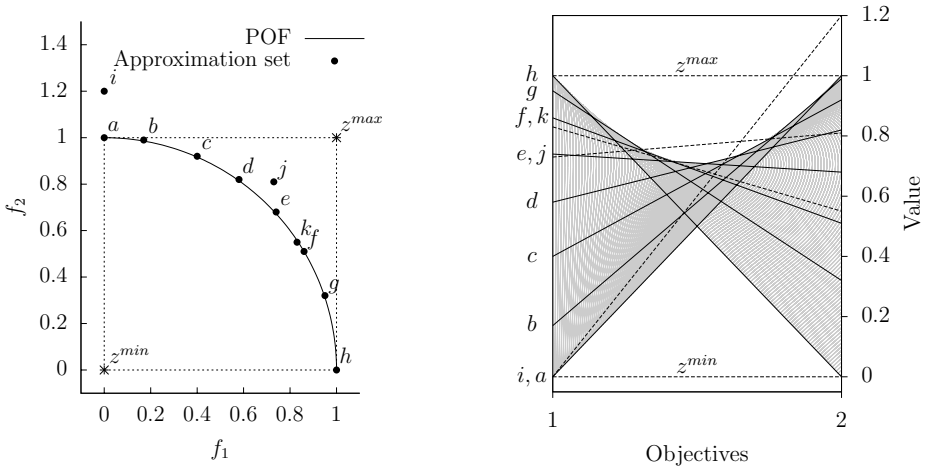


Figure 4.1: An example of approximation set in two-dimensional objective space (left) and its corresponding Parallel-Coordinates graph (right).

proposal is based on the following observations:

1. The POF is represented by the shaded area, which we named *trade-off area*.

2. The boundaries between the white regions and the trade-off area, which we call *Pareto coordinates*, give us a hint of the shape of the POF.

3. Those individuals intersecting the upper-Pareto coordinate are beyond the POF (see for example points i, j and $z^{\max}$).

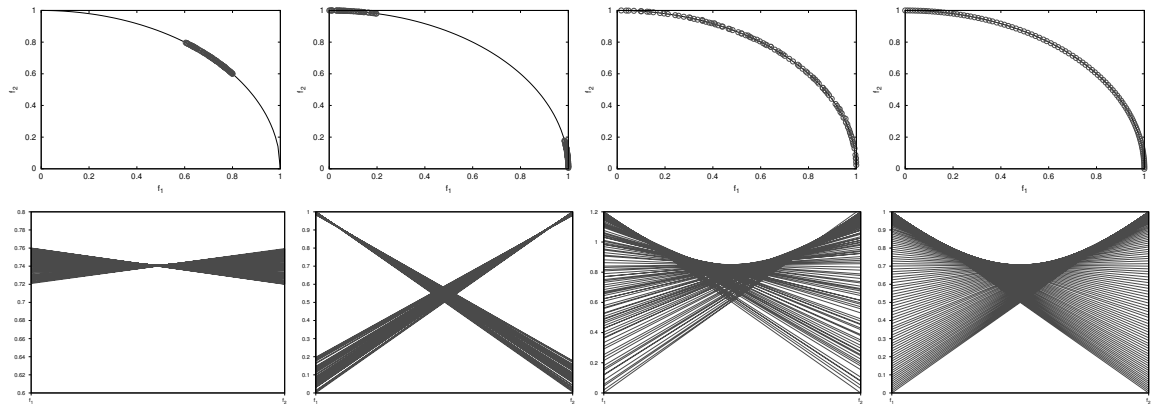
4. As the number of uniformly distributed solutions in the approximation set increases, the coverage of the trade-off area becomes better (see Figure 4.2).

Therefore, the 2D-graphs of each distinct pair of objective functions are transformed into a *digital image*[1] and this information is extracted in order to assign a contribution to each individual.

Hu and Yen [63] have been the only ones to propose density estimators based on Value Path, embedding them within a multi-objective particle swarm optimizer. The basic principle is that non-dominated solutions are ranked according to the height of intersection between the parallel axes and the polylines. In contrast with our proposal, this algorithm is sensitive to objective ordering.

To the best of our knowledge, MOVAP is the first evolutionary algorithm that incorporates automatic image analysis [47] in its search mechanism.

[1] The term *image* refers to a two dimensional light intensity function $g(a, b)$ where a and b denote spatial coordinates and the value of g at any point (a, b) is proportional to the brightness (or gray level) of the image at that point. Thus, a *digital image* is an image that has been discretized in both its spatial coordinates and brightness [47].



a) $HV = 0.226$, $E_1 = 660,543.924$ | b) $HV = 0.240$, $E_1 = 296,738.869$ | c) $HV = 0.419$, $E_1 = 99,370.406$ | d) $HV = 0.420$, $E_1 = 53,425.950$

Figure 4.2: Different distributions of a concave Pareto front (above) and their corresponding Parallel-Coordinates graph (bottom). In each case, the hypervolume and 1-energy values are shown. For the former, the reference point $(1.1, 1.1)$ is adopted.

4.2 Proposed Approach

MOVAP uses Pareto dominance as its primary search engine and a density estimator based on Parallel Coordinates, which is applied to both the parent and the survival selection mechanisms.

The core idea is to create a digital image containing the Parallel Coordinates of each distinct pair of objective functions (see Algorithm 4). These sub-graphs are attached next to each other and only normalized individuals are considered (line 5). The digital image is represented as a matrix, where its dimensions are fixed, depending on the number of objectives (m), the population size ($|P|$) and the resolution parameter (γ) (lines 1-4). An element of this array (*pixel*) corresponds to the number of intersecting polylines (line 14) and the region above the upper-Pareto coordinate is identified with the value $|P| + 1$ (lines 16-20). Therefore, the gray levels oscillate in the range $[0, |P| + 1]$.

In Algorithm 4, we adopt the notation $p.\mathbf{y}$ to refer to the objective vector of an individual p. The total number of sub-graphs is given in line 1, and each line segment is represented by its start and end pixels (lines 9-12). The Midpoint line algorithm [12],[36, p. 74], which is discussed later in this chapter, is used to efficiently compute the coordinates of the pixels that lie close to every line segment. Furthermore, in Figure 4.3, we show an example of a digital image using a resolution parameter $\gamma = 3$.

The next step is to determine the density of an individual (see Algorithm 5). For this purpose, all pixels, as well as their boundaries of a polyline are inspected (lines 13-27). The boundary of a pixel q is limited to the eight points that are at a unit distance from it, denoted by $N_8(q)$ (see, e.g., coordinate $(15, 7)$ from Figure 4.3). This set of neighbors must fall inside the current sub-graph (denoted by c).

Algorithm 4 Build PC-image

Input: Population P, resolution γ, objectives m
Output: Image $I_{A\times B}$, parameter θ

$s \leftarrow m(m-1)/2$
$\theta \leftarrow \gamma|P|/s$
$A \leftarrow \theta, B \leftarrow \theta s$
$I \leftarrow \mathbf{0}_{A\times B}$
for all $\{p \in P\colon p.\mathbf{y} \in [0,1]^m\}$ **do**
  $c \leftarrow 0$
  for all $i \in \{1,\ldots,m-1\}$ **do**
    for all $j \in \{i+1,\ldots,m\}$ **do**
      $x_0 \leftarrow \theta c$
      $y_0 \leftarrow -\lfloor(\theta-1)(1-p.y_i)\rfloor$
      $x_1 \leftarrow \theta c + \theta - 1$
      $y_1 \leftarrow -\lfloor(\theta-1)(1-p.y_j)\rfloor$
      $\langle(b_k,a_k)\rangle_{k=0}^{\theta-1} \leftarrow$ Midpoint Line(x_0,y_0,x_1,y_1)
      $I[-a_k,b_k] \leftarrow I[-a_k,b_k]+1$, for all $k=0,\ldots,\theta-1$
      $c \leftarrow c+1$
for all $b \in \{0,\ldots,B\}$ **do**
  $a \leftarrow 0$
  while $a < A$ and $I[a,b]=0$ **do**
    $I[a,b] \leftarrow |P|+1$
    $a \leftarrow a+1$
return I

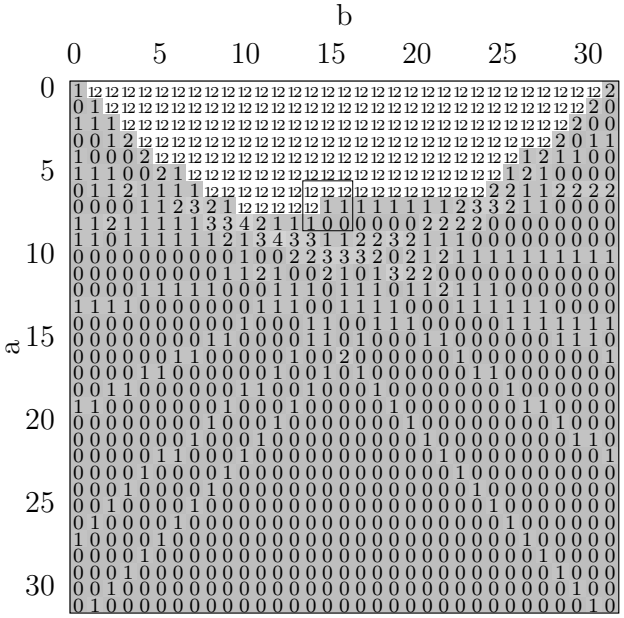


Figure 4.3: Digital image of Fig. 4.1

Algorithm 5 Calculate population density

Input: Population P, image $I_{A\times B}$, parameter θ, extreme solutions E, objectives m
Output: Density values $D_{|P|\times 1}$

1: $D \leftarrow \mathbf{0}_{|P|\times 1}$
2: $v_{\min} \leftarrow \infty$, $v_{\max} \leftarrow -\infty$
3: **for all** $\left\{p \in \left(P \setminus E\right) : p.\mathbf{y} \in [0,1]^m\right\}$ **do**
4: $\quad c \leftarrow 0$
5: $\quad$ **for all** $i \in \{1,\ldots,m-1\}$ **do**
6: $\quad\quad$ **for all** $j \in \{i+1,\ldots,m\}$ **do**
7: $\quad\quad\quad x_0 \leftarrow \theta c$
8: $\quad\quad\quad y_0 \leftarrow -\lfloor(\theta-1)(1-p.y_i)\rfloor$
9: $\quad\quad\quad x_1 \leftarrow \theta c + \theta - 1$
10: $\quad\quad\quad y_1 \leftarrow -\lfloor(\theta-1)(1-p.y_j)\rfloor$
11: $\quad\quad\quad \langle(b_k,a_k)\rangle_{k=0}^{\theta-1} \leftarrow$ Midpoint Line(x_0,y_0,x_1,y_1)
12: $\quad\quad\quad$ **for all** $k \in \{0,\ldots,\theta-1\}$ **do**
13: $\quad\quad\quad\quad S \leftarrow N_8(I[-a_k,b_k])$
14: $\quad\quad\quad\quad n_{edge} \leftarrow \mid \left\{s \in S\colon s = |P|+1\right\} \mid$
15: $\quad\quad\quad\quad n_{empty} \leftarrow \mid \{s \in S\colon s = 0\} \mid$
16: $\quad\quad\quad\quad n_{filled} \leftarrow 8 - n_{edge} - n_{empty}$
17: $\quad\quad\quad\quad n_{sum} \leftarrow \sum_{s\in S, s>0} s$
18: $\quad\quad\quad\quad D[p] \leftarrow D[p] + (A-a)I[-a_k,b_k]$
19: $\quad\quad\quad\quad$ **if** $n_{empty} > 0$ and $n_{edge} > 0$ **then**
20: $\quad\quad\quad\quad\quad D[p] \leftarrow D[p] + (a+1)n_{edge}n_{empty}$
21: $\quad\quad\quad\quad$ **else**
22: $\quad\quad\quad\quad\quad D[p] \leftarrow D[p] - (a+1)\left(n_{edge} + n_{empty}\right)$
23: $\quad\quad\quad\quad$ **if** $n_{filled} > 0$ **then**
24: $\quad\quad\quad\quad\quad$ **if** $m > 2$ and $(b \mod 2) = 0$ **then**
25: $\quad\quad\quad\quad\quad\quad D[p] \leftarrow D[p] - (A-a)n_{filled}$
26: $\quad\quad\quad\quad\quad$ **else**
27: $\quad\quad\quad\quad\quad\quad D[p] \leftarrow D[p] + (A-a)n_{sum}/n_{filled}$
28: $\quad\quad\quad c \leftarrow c + 1$
29: $\quad v_{\min} \leftarrow \min\{v_{\min}, D[p]-1\}$
30: $\quad v_{\max} \leftarrow \max\{v_{\max}, D[p]\}$
31: Normalize $\forall p \in \left(P \setminus E\right)$:

$$D[p] \leftarrow \begin{cases} \frac{D[p]-v_{\min}}{v_{\max}-v_{\min}} & \text{if } p \in [0,1]^m \\ \|p.\mathbf{y}\| & \text{otherwise} \end{cases}$$

32: **return** D

During the inspection process, the number of edged, empty and filled pixels are counted (lines 14-17). The overall density is updated according to their values, in such a way that highly dense points (lines 18 and 27) or those intersecting the upper-Pareto coordinate (lines 19 and 20) are penalized, whereas isolated ones (lines 22 and

25), belonging to the trade-off area, are rewarded. This set of rules or heuristics were derived from empirical knowledge with the aim to obtain well-distributed solutions in objective space. Therefore, the density estimator of MOVAP can be seen as an off-line selection hyper-heuristic.

Once the matrix has been processed, the population density is normalized (line 31). Thus, extreme solutions obtain zero (the best) value and the most crowded gets one (the worst value). The density of individuals lying outside the interval $[0, 1]$ is equal to the Euclidean norm of the objective vector.

The proposed density estimator is coupled with a steady state genetic algorithm, where one offspring is created at each generation. In the following paragraph, we describe the main loop of MOVAP, shown in Algorithm 6.

First, the population is initialized, either randomly or from a previously computed solution (line 3). At each iteration, a binary tournament selection is performed, based on the population density (line 5). Thus, isolated individuals have a high probability of mating. Next, an offspring is created using the variation operators (line 6). The new individual is added to the current population and a normalization procedure (explained later on) is invoked (lines 7 and 8). If there are individuals outside the region of interest, the one with the highest norm is removed from the population. Otherwise, the population is ranked using the non-dominated sorting procedure of NSGA-II [30] (lines 10-13). If the last front consists of one individual (lines 14 and 15), then it is eliminated. Otherwise, the digital image is built for calculating the density estimator, and the individual with the highest value is discarded (lines 17-19).

The normalization done in Algorithm 7 works in objective space and serves for three purposes: it translates vectors to the origin, if their coordinates are negative (lines 1-3); it finds the extreme points (lines 5-8); and it normalizes the population (line 9). It is noteworthy that if there are several candidates parallel to one axis, the solution with the lowest norm is preferred. Additionally, each component of $\mathbf{z}^{\min}$ corresponds to the best objective value found so far (line 4).

Re-taking our previous example of Figure 4.1, we provide the density values of the eleven non-dominated individuals in Table 4.1. The best individuals are the extreme solutions a and h, whereas the three candidates to be removed are i, j and k. This is because they are outside the region of interest, beyond the Pareto front or too close to other individuals. Moreover, the special pattern of solution j, where it intersects the upper-Pareto coordinate, is recognized in the bounded pixels of Figure 4.3. Finally, the concave geometry of the approximation set can be appreciated as a valley-like curve in the upper-Pareto coordinate. Here, it is worth mentioning that when the geometry is convex, the lower-Pareto coordinate takes a mountain-like curved shape, while when it is linear, the upper-Pareto coordinate forms a triangle. In high dimensionality, if the same pattern is repeated for each pair of objectives, we can infer that the Pareto front adopts such form. Otherwise, it corresponds to a mixed shape.

Algorithm 6 Main loop of MOVAP

Input: MOP, stopping criterion, image resolution γ
Output: Approximation set P

$i \leftarrow 1$
$D \leftarrow \mathbf{0}_{|P|\times 1}$
Initialize population P^i
while termination condition is not fulfilled **do**
  Perform parent selection
  Create an offspring o
  $P^i \leftarrow P^i \cup \{o\}$
  $[P', E] \leftarrow$ Normalize objectives (P^i, m)
  $Q \leftarrow \left\{p \in P' : p.\mathbf{y} \notin [0,1]^m\right\}$
  if $Q \neq \emptyset$ **then**
    $r \leftarrow \arg\max_{\mathbf{q}\in Q} \|q.\mathbf{y}\|$
  else
    $\{F_1, \ldots, F_k\} \leftarrow$ Non-dominated sorting(P^i)
    if $|F_k| = 1$ **then**
      $r \leftarrow F_k$
    else
      $[I, \theta] \leftarrow$ Build PC-image (P', γ, m)
      $D \leftarrow$ Calculate pop. density (P', I, θ, E, m)
      $r \leftarrow \arg\max_{p\in F_k} D[p]$
  Reduce population $P^{i+1} \leftarrow P^i \setminus \{r\}$
  $i \leftarrow$i + 1
return P^i

Algorithm 7 Normalize objectives

Input: Population P, objectives m
Output: Population P', extreme points E

$v_i \leftarrow | \min\left(\{0\} \cup \{p.y_i : p \in P\}\right) |$, $\forall i \in \{1, \ldots, m\}$
if $\mathbf{v} \neq \mathbf{0}$ **then**
  $P' \leftarrow \{p.\mathbf{y} + \mathbf{v} : p \in P\}$
Update the minimum reference point $\mathbf{z}^{min}$
for all $i \in \{1, \ldots, m\}$ **do**
  $e \leftarrow \arg\min_{p\in P'} \frac{p.y_i}{\|p.\mathbf{y}\|}$
  $E \leftarrow E \cup \{e\}$
  $z_i^{\max} \leftarrow e_i$ {see Figure 4.1}
$p.\mathbf{y} \leftarrow \frac{p.\mathbf{y} - \mathbf{z}^{min}}{\mathbf{z}^{\max} - \mathbf{z}^{\min}}$, $\forall p \in P'$
return P', E

Table 4.1: Sample data

Solution	**f**	Density
a	(1e-12, 1.00)	0.0000
b	(0.17, 0.99)	0.0003
c	(0.40, 0.92)	0.2491
d	(0.58, 0.82)	0.4747
e	(0.74, 0.68)	0.4753
f	(0.86, 0.51)	0.6107
g	(0.95, 0.32)	0.2067
h	(1.00, 0.00)	0.0000
i	(0.00, 1.20)	1.2000
j	(0.73, 0.81)	1.0000
k	(0.83, 0.55)	0.6819

The Midpoint line method, invoked in Algorithms 4 and 5, is given in Algorithm 8. It was originally proposed by Bresenham [12] and later simplified by Foley et al. [36, p. 74]. Given start (x_0, y_0) and end (x_1, y_1) pixels describing a line with slope between $[-1, 1]$, this method efficiently computes the coordinates of the pixels that lie close to such line. The main advantage of this algorithm is its speed since it avoids divisions and frequent calls to the floor function ($\lfloor \cdot \rfloor$). Another merit is that it prompts a maximum error of $\frac{1}{2}$.

Algorithm 8 Midpoint Line [12],[36, p. 74]

Input: Start (x_0, y_0) and end (x_1, y_1) points
Output: Sequence of points S discretizing a line in the fourth quadrant

$dx \leftarrow x_1 - x_0$
$dy \leftarrow y_1 - y_0$
$m \leftarrow dy/dx$ {Slope's line}
$incrE \leftarrow 2dy$ {Increment used for moving east}
$incrNE \leftarrow 2(dy - dx)$ {Increment used for moving northeast}
$incrSE \leftarrow 2(dy + dx)$ {Increment used for moving southeast}
$x \leftarrow x_0$
$y \leftarrow y_0$
$S \leftarrow (x, y)$
if $m \geq 0$ **then** {Increasing or horizontal line}
  $d \leftarrow 2dy - dx$ {Midpoint evaluation $d = F(x, y + \frac{1}{2}) = Ax + By + C$}
  while $x < x_1$ **do**
    if $d \leq 0$ **then** {Midpoint is above or in the line, thus we choose east}
      $d \leftarrow d + incrE$
    else {Midpoint is below the line, so we choose northeast}
      $d \leftarrow d + incrNE$
      $y \leftarrow y + 1$
    $x \leftarrow x + 1$
    $S \leftarrow S \parallel (x, y)$
else {Decreasing line}
  $d \leftarrow 2dy + dx$ {Midpoint evaluation $d = F(x, y - \frac{1}{2}) = Ax + By + C$}
  while $x < x_1$ **do**
    if $d \geq 0$ **then** {Midpoint is above or in the line, thus we choose east}
      $d \leftarrow d + incrE$
    else {Midpoint is below the line, so we choose southeast}
      $d \leftarrow d + incrSE$
      $y \leftarrow y - 1$
    $x \leftarrow x + 1$
    $S \leftarrow S \parallel (x, y)$
return S

The conceptual idea is that the next pixel (x_{i+1}, y_{i+1}) is calculated incrementally, i.e., by using information of the current pixel (x_i, y_i). For this purpose the midpoint is defined as $(x_i+1, y_i+\frac{1}{2})$ if the slope m is positive, or $(x_i+1, y_i-\frac{1}{2})$ if it is negative. Depending on the position of the midpoint concerning the line (above or below), there are three possible movements: northeast, east or southeast. Moreover, to determine the position of the midpoint, the algorithm tests the sign of the general form of the linear equation at the midpoint. This value is denoted by d in Algorithm 8. In lines 1-3 , the changes in the coordinate axes, as well as the line's slope are calculated. The possible increments of d are calculated in lines 4-6. Next, the current pixel is initialized and added to the sequence of points S in lines 7-9. According to the slope's sign, the d value is initialized in lines 11 and 21. At each iteration of lines 12-19 and 22-29, the d is inspected for making the right movement. Then, d is updated, and the new pixel is added to the sequence ($\|$ denotes concatenation). This procedure is repeated until the end pixel is reached.

4.3 Computational Complexity

In the following, we determine the complexity of MOVAP (Algorithm 6). Parent selection is performed in $\mathcal{O}(1)$, as well as the offspring generation and population reduction. The normalization (Algorithm 7) and verification of individuals inside the region of interest is done in $\mathcal{O}(|P|m)$, each. Nondominated sorting is of $\mathcal{O}(|P|^2m)$. The building of the image (Algorithm 4) and the density calculation (Algorithm 5) can be performed in $\mathcal{O}(|P|^2m^2)$. Therefore, the overall complexity of MOVAP at each iteration is $\mathcal{O}(|P|^2m^2)$ with a maximum storage of $\mathcal{O}(|P|^2)$.

4.4 Experimental Results

In this section, we investigate the effectiveness of MOVAP, not only in artificial many-objective problems (of 5 and 7 objectives), but also in low dimensionality. For this purpose, we present a comparative study that includes the algorithms SPEA2, NSGA-II, NSGA-III and HypE, all of which were developed in our framework EMO Project (see Chapter 6 on page 89). Next, we describe the experimental settings, test problems, and the performance assessment measure adopted.

All the MOEAs were implemented using real-numbers encoding and their parameters were identical (see Table 4.2). The variation operators were polynomial-based mutation and SBX. As suggested in [28], the crossover rate and its distribution index were set to 0.9 and 20, for 2 and 3 objectives, and 1.0 and 30 for many-objective problems. The mutation rate and its distributed index was set to $1/n$ and 20, respectively. For NSGA-III, the set of weight vectors was generated using the Simplex-Lattice Design method (see page 23) with a proportionality parameter H shown in Table 4.2. For HypE, the number of sampling points was fixed to 20,000. In MOVAP, the image resolution was empirically determined, being independent of the problem to

Table 4.2: Parameters adopted in our study

m	WFG		$\|P\|$	NSGA-III	MOVAP
	n	k		H	γ
2	24	4	100	99	3
3	24	4	120	14	2
5	47	8	196	4,5	2
7	71	12	210	4,5	2

be solved. For this purpose, we correlated its behavior with the hypervolume indicator, and we found the average optimum values of Table 4.2. We observed that as this value increases, the overlapping level among polylines is minimum, and even though the individuals are not well distributed, the density estimator reflects the opposite.

For two objectives, we selected the five real-valued problems of the ZDT test suite and for larger dimensionality, we used the WFG benchmark. The decision variables (n) for ZDT were set according to the original specification. In the case of WFG, the variables and position-related parameter (k) are specified in Table 4.2. The number of function evaluations was set to 40,000 and 50,000 for the ZDT and WFG test problems, respectively. For comparing results, we adopted the hypervolume indicator. The reference points used were $(1.1, \ldots)$ for the ZDT test suite, $(3, 5, 7, \ldots)$ for the instances WFG1 and WFG3; and $(2.2, 4.2, 6.2, \ldots)$ for the rest of the problems. We also used the Value Path for inspecting diversity. We performed 30 independent runs of each of the five MOEAs compared on all the test instances adopted. With the aim of comparing the performance of all algorithms among themselves in a pairwise fashion, the Wilcoxon rank sum test (one-tailed) with the Bonferroni correction was applied to the hypervolume indicator values. Experimental results appear in Tables 4.3 and 4.4, and some examples of the approximation sets, corresponding to the median values, are depicted in Figures 4.4 and 4.5. For comparison purposes in the ZDT benchmark, the approximation sets are plotted with a vertical shift.

4.5 Discussions

With seven and five objectives, the best algorithm was MOVAP, obtaining the highest hypervolume values and significantly outperforming the other algorithms: HypE, NSGA-III, NSGA-II and SPEA2. Only in WFG2 (a problem with disconnected geometry), for seven objectives, MOVAP was second, without being significantly surpassed by NSGA-II. The second best optimizer was NSGA-III, which was able to get very close to the Pareto optimal fronts. However, it produced very poor diversity. On the other hand, HypE encouraged spread over distribution, being unable to cover the complete Pareto front. NSGA-II and SPEA2, in general, experimented some stagnation during the search, being incapable of reaching the optimal solution.

In three and two objectives, MOVAP ranked second, producing similar results to HypE (which ranked first) and significantly outperforming SPEA2, NSGA-II and NSGA-III in almost all cases. This behavior was expected, since HypE is using the

Table 4.3: Median and standard deviation of the hypervolume indicator. In each case, the outperformance relation among algorithms is shown, using a significance level of $\alpha = 0.5$ (for example, SPEA2 performs significantly better than HypE on WFG1). The two best values are shown in gray scale, where a darker tone corresponds to the best value.

Problem	SPEA2 (1)	NSGA-II (2)	NSGA-III (3)	HypE (4)	MOVAP (5)
7 objectives					
WFG1	6.18e+05(7.97e+3) 4	6.49e+05(9.21e+3) 1,4	6.71e+05(1.33e+4) 1,2,4	5.32e+05(1.96e+4) –	8.36e+05(1.64e+4) 1,2,3,4
WFG2	7.88e+05(7.98e+3) 4	8.11e+05(5.74e+3) 1,3,4	7.96e+05(4.65e+4) 4	7.24e+05(6.64e+4) –	8.09e+05(6.77e+3) 1,3,4
WFG3	1.06e+06(8.32e+4) –	1.32e+06(1.87e+4) 1,3,4	1.05e+06(3.21e+4) –	1.13e+06(3.48e+4) 1,3	1.42e+06(6.50e+3) 1,2,3,4
WFG4	4.65e+05(1.89e+4) 2,4	3.15e+05(1.69e+4) 4	5.72e+05(8.12e+3) 1,2,4	2.76e+05(1.59e+4) –	6.52e+05(5.12e+3) 1,2,3,4
WFG5	4.31e+05(1.79e+4) 2,4	3.25e+05(1.85e+4) 4	5.52e+05(5.52e+3) 1,2,4	2.94e+05(1.60e+4) –	6.14e+05(3.70e+3) 1,2,3,4
WFG6	4.24e+05(2.17e+4) 2,4	3.34e+05(2.26e+4) 4	5.64e+05(1.58e+4) 1,2,4	2.68e+05(2.64e+4) –	5.99e+05(1.40e+4) 1,2,3,4
WFG7	3.68e+05(2.36e+4) 2,4	3.25e+05(1.91e+4) 4	5.99e+05(8.52e+3) 1,2,4	2.94e+05(1.63e+4) –	6.65e+05(4.40e+3) 1,2,3,4
WFG8	3.42e+05(2.34e+4) 2,4	3.23e+05(1.48e+4) 4	4.69e+05(4.10e+4) 1,2,4	2.42e+05(1.72e+4) –	5.05e+05(4.43e+3) 1,2,3,4
WFG9	4.14e+05(1.87e+4) 2,4	2.78e+05(1.78e+4) 4	4.71e+05(2.52e+4) 1,2,4	2.32e+05(1.74e+4) –	4.82e+05(1.81e+4) 1,2,4
5 objectives					
WFG1	3.60e+03(8.17e+1) 4	3.71e+03(7.79e+1) 1,3,4	3.67e+03(4.94e+1) 1,4	2.82e+03(1.17e+2) –	4.57e+03(1.07e+2) 1,2,3,4
WFG2	4.61e+03(2.56e+2) 4	4.63e+03(2.19e+2) 4	4.61e+03(2.26e+2) 4	4.24e+03(3.00e+2) –	4.64e+03(3.78e+2) 4
WFG3	6.19e+03(1.65e+2) 4	6.93e+03(1.11e+2) 1,3,4	6.33e+03(1.02e+2) 4	5.55e+03(1.55e+2) –	7.34e+03(5.36e+1) 1,2,3,4
WFG4	2.72e+03(6.26e+1) 2,4	2.34e+03(8.39e+1) 4	3.11e+03(4.16e+1) 1,2,4	1.69e+03(9.10e+1) –	3.38e+03(2.12e+1) 1,2,3,4
WFG5	2.63e+03(5.72e+1) 2,4	2.39e+03(7.30e+1) 4	2.94e+03(2.33e+1) 1,2,4	1.96e+03(1.33e+2) –	3.19e+03(1.31e+1) 1,2,3,4
WFG6	2.55e+03(5.85e+1) 2,4	2.21e+03(8.35e+1) 4	2.97e+03(5.63e+1) 1,2,4	1.80e+03(1.36e+2) –	3.21e+03(4.93e+1) 1,2,3,4
WFG7	2.44e+03(6.60e+1) 2,4	2.19e+03(1.34e+2) 4	3.22e+03(2.94e+1) 1,2,4	1.82e+03(1.10e+2) –	3.45e+03(1.62e+1) 1,2,3,4
WFG8	2.03e+03(7.13e+1) 2,4	1.88e+03(4.94e+1) 4	2.39e+03(4.80e+1) 1,2,4	1.48e+03(1.24e+2) –	2.66e+03(2.56e+1) 1,2,3,4
WFG9	2.47e+03(8.43e+1) 2,4	2.09e+03(1.17e+2) 4	2.63e+03(1.21e+2) 1,2,4	1.75e+03(1.68e+2) –	2.75e+03(5.47e+1) 1,2,3,4

exact hypervolume in its search mechanism and we are using the same performance indicator for comparing results. Nonetheless, we found that HypE focused more on the spread than on the distribution of solutions, while MOVAP favored distribution over spread. For example, in ZDT2, which has a concave Pareto front, MOVAP was able to find good representatives near the extreme points (see Figure 4.4). Moreover

Table 4.4: Median and standard deviation of the hypervolume indicator (cont'd).

Problem	SPEA2 (1)	NSGA-II (2)	NSGA-III (3)	HypE (4)	MOVAP (5)
3 objectives					
WFG1	4.75e+01(2.65e+0) 3	4.41e+01(2.12e+0) 3	4.22e+01(2.93e+0) –	5.66e+01(1.62e+0) 1,2,3,5	5.25e+01(2.32e+0) 1,2,3
WFG2	5.20e+01(3.62e+0) 2,3	5.15e+01(3.73e+0) –	5.14e+01(4.02e+0) –	5.34e+01(4.21e+0) 1,2,3,5	4.43e+01(4.13e+0) –
WFG3	7.28e+01(3.22e-1) –	7.45e+01(3.61e-1) 1,3	7.28e+01(3.81e-1) –	7.59e+01(2.19e-1) 1,2,3,5	7.51e+01(3.71e-1) 1,2,3
WFG4	2.67e+01(2.41e-1) 2	2.51e+01(4.26e-1) –	2.78e+01(1.09e-1) 1,2	2.97e+01(4.66e-2) 1,2,3,5	2.88e+01(1.13e-1) 1,2,3
WFG5	2.54e+01(2.03e-1) 2	2.42e+01(2.93e-1) –	2.59e+01(1.09e-1) 1,2	2.74e+01(9.65e-1) 1,2,3,5	2.68e+01(1.25e-1) 1,2,3
WFG6	2.51e+01(3.69e-1) 2	2.39e+01(4.60e-1) –	2.59e+01(3.08e-1) 1,2	2.77e+01(2.68e-1) 1,2,3,5	2.71e+01(3.16e-1) 1,2,3
WFG7	2.74e+01(2.53e-1) 2	2.62e+01(3.61e-1) –	2.84e+01(7.93e-2) 1,2	2.98e+01(2.01e-2) 1,2,3,5	2.92e+01(3.78e-2) 1,2,3
WFG8	2.21e+01(2.03e-1) 2	2.11e+01(2.94e-1) –	2.30e+01(1.60e-1) 1,2	2.34e+01(2.81e-1) 1,2,3	2.40e+01(1.22e-1) 1,2,3,4
WFG9	2.36e+01(1.02e+0) 2,4	2.25e+01(8.25e-1) 4	2.58e+01(1.33e+0) 2,4	2.17e+01(1.75e+0) –	2.40e+01(1.44e+0) 2,4
2 objectives					
ZDT1	8.71e-01(3.10e-4) 2	8.70e-01(5.70e-4) –	8.71e-01(3.36e-5) 1,2	8.72e-01(2.79e-5) 1,2,3,5	8.72e-01(8.58e-5) 1,2,3
ZDT2	5.38e-01(5.40e-4) 2	5.37e-01(4.66e-4) –	5.38e-01(5.15e-5) 1,2	5.39e-01(2.59e-5) 1,2,3,5	5.38e-01(3.69e-5) 1,2,3
ZDT3	1.33e+00(9.72e-4) 3	1.33e+00(1.18e-3) 3	1.33e+00(4.01e-4) –	1.33e+00(1.52e-2) 1,2,3,5	1.33e+00(3.39e-2) 1,2,3
ZDT4	8.68e-01(1.94e-3) –	8.68e-01(1.35e-3) –	8.69e-01(1.64e-3) –	8.71e-01(5.63e-4) 1,2,3,5	8.70e-01(2.03e-3) –
ZDT6	4.99e-01(1.14e-3) 3	4.99e-01(1.14e-3) 3	4.97e-01(1.59e-3) –	5.03e-01(3.01e-4) 1,2,3,5	5.03e-01(3.12e-4) 1,2,3

in WFG8, a non-separable and biased problem, for three objectives, MOVAP outperformed HypE. Only in WFG2, MOVAP could not perform significantly better than the other algorithms. With respect to NSGA-III, it produced a more uniform distribution than SPEA2 and NSGA-II, obtaining the best results in WFG9 (a multimodal, deceptive, non-separable and biased problem) for three objectives. Finally, SPEA2 obtained slightly better results than NSGA-II, standing out by the diversity of its solutions.

In conclusion, we observed that MOVAP produced much better solutions near the Pareto optimal front than NSGA-II and SPEA2 in low and high dimensionality. With respect to NSGA-III, it had better diversity and in comparison with HypE, MOVAP was competitive in low dimensionality and produced much better results for five and seven objectives. For this reason, we believe that our proposed approach is a promising alternative for solving MOPs, in both low and high dimensionality.

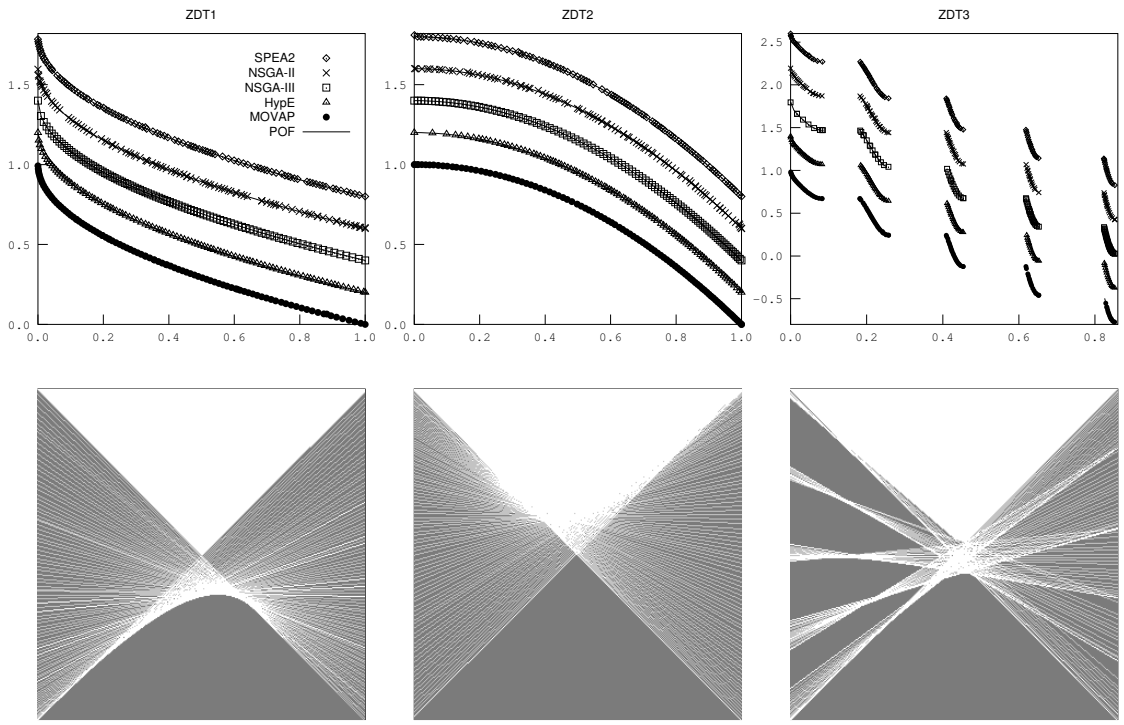


Figure 4.4: Pareto fronts produced by MOEAs on some problems of the ZDT test suite (top) and the corresponding digital images generated by MOVAP (bottom).

4.6 Summary

This chapter presents a new algorithm, called MOVAP (Multi-Objective Optimizer based on Value Path), which uses image analysis concepts in its selection mechanism. The basic idea consists in discretizing the Parallel Coordinates graph and assigning a fitness value to each individual based on the density of its polylines. Our experimental results indicated that the proposed approach significantly outperforms HypE, NSGA-III, NSGA-II and SPEA2 in more than 35% of the test instances, producing much better diversity of solutions, and exploring more regions of the search space in high-dimensionality than the MOEAs with respect to which it was compared. Whereas in low dimensionality, our proposed approach was competitive, producing very similar results to those generated by HypE. Moreover, the complexity of MOVAP is quadratic with respect to the number of objectives and the population size. Based on these preliminary results, we believe that our proposed approach is a suitable alternative for solving many-objective problems.

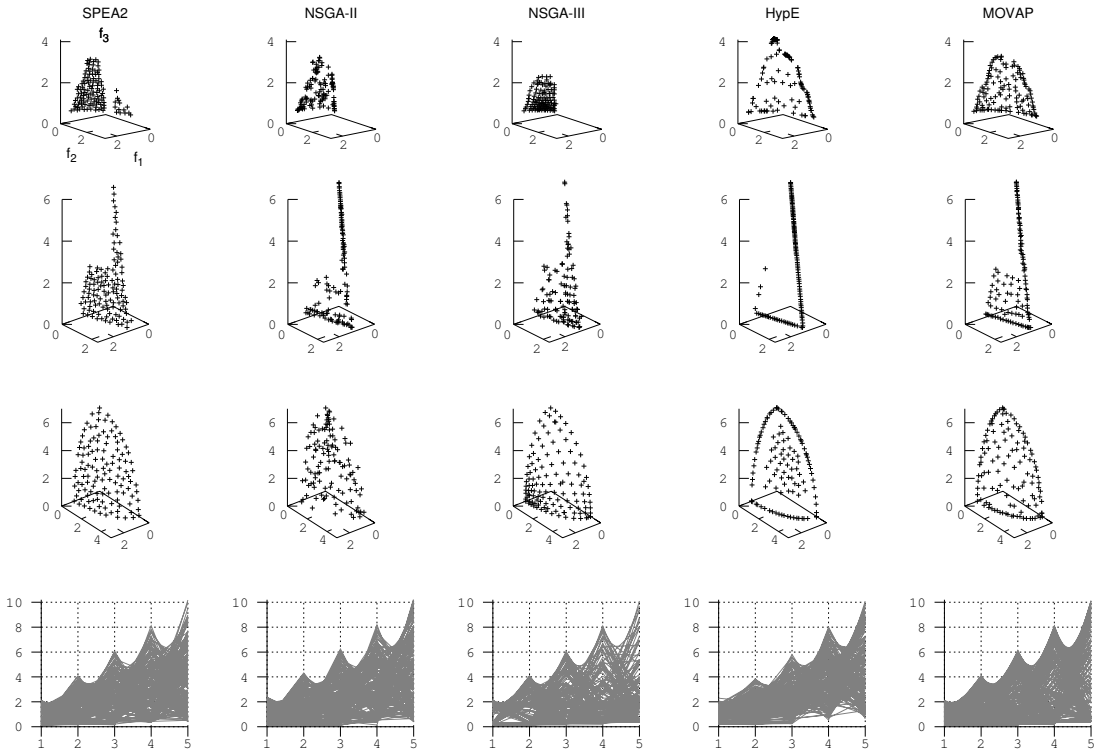


Figure 4.5: Approximation sets of MOEAs in WFG1, WFG3, WFG4 and WFG9 (from top to bottom).

Chapter 5

A Parallel Version of SMS-EMOA

In the last decade, there has been a growing interest in multi-objective evolutionary algorithms that use performance indicators to guide the search. A simple and effective one is the $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) [34], which is based on the hypervolume indicator. Even though the maximization of the hypervolume is equivalent to achieving Pareto optimality, its computational cost increases exponentially with the number of objectives, which severely limits its applicability to many-objective optimization problems. In this chapter, we present a parallel version of SMS-EMOA, where the execution time is reduced through an asynchronous island model with micro-populations. Furthermore, diversity is preserved by external archives that are pruned to a fixed size employing a technique based on the Parallel-Coordinates graph (presented in Chapter 4). The proposed approach is called $\mathcal{S}$-PAMICRO (PArallel MICRo Optimizer based on the $\mathcal{S}$ metric).

The organization of this chapter is as follows. In Section 5.1, we give the motivation and some background. Section 5.2 is devoted to the description of our proposed parallel MOEA. In Section 5.3 we present our experiments. Finally, Section 5.4 provides a summary of the Chapter.

5.1 Motivation

We focus on SMS-EMOA due to its simplicity and superiority over several Pareto- and aggregation-based algorithms [34, 77, 135]. This optimizer creates an offspring at each iteration from randomly selected parents. In the survival selection, it adopts the non-dominated sorting procedure of NSGA-II [30], where the discarded individual is chosen from the last front, having the lowest hypervolume contribution. For a deeper discussion on the computation of the hypervolume contribution, see Appendix A, where the most efficient algorithm proposed by the Walking Fish Group is reproduced without errors. Since the worst-case complexity of SMS-EMOA is $\mathcal{O}(|P|^m)$ [143], parallelizing it arises as a possible alternative to reduce its computational cost, where at least two strategies are possible [94]: 1) parallelization of the computations, in which the operations applied to an individual are performed in parallel, and

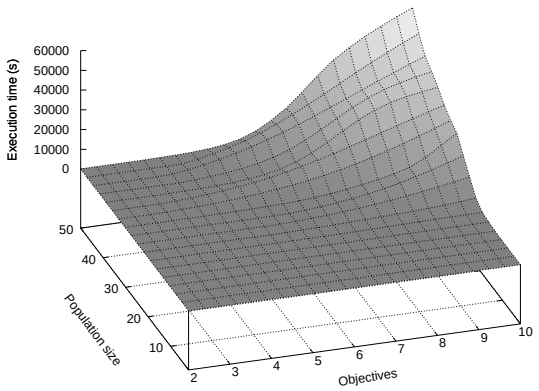


Figure 5.1: Average execution time of SMS-EMOA.

2) parallelization of the population, in which the population is partitioned and each subpopulation evolves in semi-isolation (individuals can be exchanged between subpopulations). Klinkenberg et al. [77], Wessing et al. [141] and Manoatl et al. [96] have studied the first approach. In [77], a variation of SMS-EMOA parallelized the expensive evaluations of individuals using a surrogate model, whose purpose was to approximate the function values. In [141], the synchronous and asynchronous master-slave implementations of SMS-EMOA were compared on problems with fluctuations in the evaluation time of the objective functions. In [96], the exact hypervolume contributions of SMS-EMOA were parallelized through the use of GPUs. To the best of our knowledge, our work is the first attempt to incorporate the second sort of approach (parallelization of the population) into SMS-EMOA.

In order to get a better grasp of the variability of the execution time of SMS-EMOA, we sampled several points on the multi-frontal test problem DTLZ1 (see Section B.3 on page 156), varying the number of objective functions and the population size on a PC Intel(R) Core(TM) i7 CPU 950 @ 3.07 GHz $\times$ 8 with 3.8 GB of memory, using the same parameters in all our experiments [34]. The average resulting surface is shown in Figure 5.1. An interesting observation is that, regardless of the number of objectives, time was almost negligible when using small populations (less than 20 individuals). This fact is considered in our proposal, where we use micro-populations in an asynchronous island model [127]. Moreover, diversity is improved by external archives that are kept to a constant size by the proposed density estimator of Chapter 4, which is scalable in objective space.

5.2 Proposed Approach

The PArallel MICRo Optimizer based on the S metric ($\mathcal{S}$-PAMICRO) draws ideas from the island model. To deal with the prohibitive computational cost of SMS-EMOA, we divide the population into several micro-populations, i.e., subpopulations (or islands) with no more than 10 or 11 individuals. Islands are connected in a logical unidirectional ring topology, where occasionally, few copies of the solutions

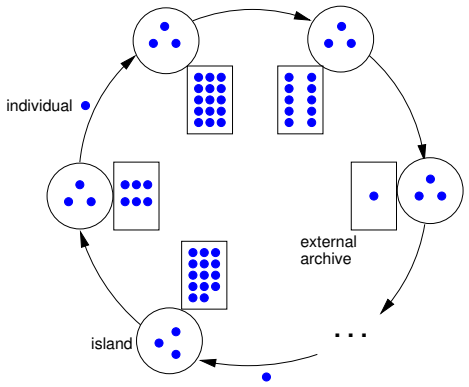


Figure 5.2: Schematic representation of the parallel micro optimizer based on the S metric.

Algorithm 9 Outline of an island in $\mathcal{S}$-PAMICRO

Input: MOP, stopping criterion, island identification id, number of islands $nisland$, number of migrants $nmig$, frequency of migration $fmig$, and image resolution γ.
Output: Final sub-population A

1: Initialize micro-population P at random
2: $A \leftarrow \emptyset$ {initialize external archive}
3: $n \leftarrow nisland \times |P|$ {archive size limit}
4: **while** the stopping criterion is not satisfied **do**
5: $P \leftarrow$ SMS-EMOA(MOP, $fmig$, P) {execute during $fmig$ evaluations of the objective vector}
6: $S \leftarrow$Migration_Policy($A \cup P$, $nmig$) {$nmig$ random solutions are selected}
7: Send copies of S to the $(id+1) \pmod n$*island* island
8: $R \leftarrow$ Check the arrival of immigrants from the island $(nisland + id - 1) \pmod{nisland}$
9: $A \leftarrow \text{NDS}(A \cup P \cup R)$
10: **if** $|A| > n$ **then**
11: $A \leftarrow$ Pruning(A, n, γ, m) {see Algorithm 10}
12: $P \leftarrow$ Replacement_Policy($P \cup R$) {dominated individuals are likely to be discarded}
13: **return** A

are transferred using asynchronous communication (i.e., they migrate). Each island evolves independently a serial SMS-EMOA in combination with an external archive of non-dominated solutions. These external archives help to maintain diversity and are pruned to a fixed size employing the technique described in Chapter 4, which is based on the Parallel-Coordinates graph. The goal of $\mathcal{S}$-PAMICRO is not only to reduce the execution time of SMS-EMOA but may also improve its exploration capabilities, because of the separated search of the islands, which changes the behavior of the serial version and yields a new kind of algorithm [94, 127]. In Figure 5.2, we present a schematic representation of $\mathcal{S}$-PAMICRO.

The pseudocode of an island is given in Algorithm 9. At the beginning, the

Algorithm 10 Pruning

Input: Population A, desired size n, image resolution γ, objectives m
Output: Reduced population A
1: $E \leftarrow$ Determine extreme solutions
2: Calculate $\mathbf{z}^*$ and $\mathbf{z}^{nad}$
3: Normalize population $a.\vec{y}' \leftarrow \frac{a.\vec{y}-\mathbf{z}^*}{\mathbf{z}^{nad}-\mathbf{z}^*}$, $\forall a \in A$, $a.\vec{y} \in \mathbb{R}^m$
4: **while** $|A| > n$ **do**
5: $[I, \theta] \leftarrow$ Build PC-image (A, γ, m) (see Algorithm 4 on page 64)
6: $D \leftarrow$ Calculate pop. density (A, I, θ, E, m) (see Algorithm 5 on page 65)
7: $r \leftarrow \arg\max_{\vec{a} \in A} D[a]$
8: $A \leftarrow A \setminus \{r\}$
9: **return** A

subpopulation and the external archive are initialized. The maximum number of individuals that the external archive can store is $|P|$ times the number of islands, where $|P|$ stands for the subpopulation size. The purpose of the external archive is to introduce elitism by retaining "good" representatives in terms of convergence and diversity. At each iteration, the parallel MOEA evolves the current subpopulation using a serial SMS-EMOA during $fmig$ evaluations of the objective vector. Next, $nmig$ individuals are selected from the external archive and the current population according to the migration policy for sending copies to the destination island. Subsequently, $\mathcal{S}$-PAMICRO checks for the arrival, without blocking, of immigrants from the source island. The external archive is updated adding the current subpopulation as well as the immigrants and then removing dominated solutions. If the external archive exceeds its limit size, a pruning step is invoked, which will be explained below. In the following, the immigrants are incorporated into the subpopulation regarding the replacement policy. At the end, the final sub-populations of all islands are collected and adjusted to the size $nisland \times |P|$, using the same pruning technique. This operation is performed by a designated island. Regarding the migration and replacement policies any of those presented on page 28 can be coupled to our scheme.

In the pruning technique (see Algorithm 10), the extreme solutions of the current Pareto front, as well as the reference points are determined. Here, extreme points are chosen in such a way that are parallel to the axes having the lowest Euclidean norm (see Algorithm 7 on page 67). Next, the population is normalized using this information. In line 3 of Algorithm 10, we adopt the notation $a.\mathbf{y}$ to refer to the objective vector of an individual a. While the population has not reached the desired size, members with the highest population density are removed one by one.

5.3 Experimental Results

In this section, we investigate the behavior and performance of $\mathcal{S}$-PAMICRO. Thus, the experiments were divided in two parts: analysis of the migration parameters

(Subsection 5.3.1), and comparison of our proposed algorithm with respect to SMS-EMOA, its parallel version using the asynchronous island model without external archives and HypE. The algorithms were implemented under EMO Project using MPICH version 3.2 and the C compiler was gcc 6.3.0. The variation operators were polynomial-based mutation and SBX. The crossover rate and its distribution index were set to 0.9 and 20, respectively, for 2 and 3 objectives, and 1.0 and 30 for many-objective problems. The mutation rate and its distributed index were set to $1/n$ and 20, respectively. We performed 30 independent runs for all scenarios. In the following, we provide further details about these experiments.

5.3.1 Migration parameters

The island model introduces six additional parameters. The numerical ones are the frequency of migration, the number of individuals to migrate at each event, and the number of islands in the model. On the other hand, the categorical parameters are the logical network topology, as well as the migration and replacement policies. In the next experiments, we analyze the behavior of all these parameters on $\mathcal{S}$-PAMICRO, except for the topology, which was originally established to be an unidirectional ring. Here, we present the results for only DTLZ1 with 2, 4, 6, 8 and 10 objectives since in the other instances, we observed similar patterns. Furthermore, when analyzing a specific parameter, the others were fixed to the following values:

- migration frequency = 100
- number of migrants = 2
- migration policy = random (R)
- replacement policy = elitist ranking (EK)

As indicated in Table 5.1, the number of islands, the external archive size and the resolution parameter varied according to the number of objectives. The stopping criterion consisted of reaching a maximum number of evaluations (fevals) of the MOP. In $\mathcal{S}$-PAMICRO, each island contained a subpopulation of 10 individuals.

For the performance assessment, we relied on the hypervolume indicator using the reference point $(2, 2, \ldots)$, and the IGD$^+$ indicator where the reference set was sampled at random $m \times 10,000$ times from the well-known Pareto fronts. Then, these points were filtered using the non-dominated sorting algorithm [30]. Executions have been done over the GNU/Linux ABACUS Cluster[1] of 268 nodes with 128 GB of RAM and Infiniband interconnection network. Each processor is a fourteen-core Intel Xeon E5-2697v3 2.6GHz.

Figure 5.3 shows the median and interquartile ranges of the hypervolume (left) and IGD$^+$ (right) indicators when varying the migration frequency (note that the

[1] `http://www.abacus.cinvestav.mx`

Table 5.1: Parameters adopted in our experiments

Objectives (m)	2	3	4	6	8	10
Islands	10	13	16	20	23	26
Archive size	100	130	160	200	230	260
Image resolution (γ)	3	2	2	2	2	2
feval ($\times 10^3$)	40	60	70	80	100	110

x-axis is in logarithmic scale). This parameter is measured in objective function evaluations (fevals). The interquartile range was interpolated by using cubic splines [38], which ensure monotonicity and convexity of the original 20 sampling points. In the case of 2 objectives, consistent results were obtained from 8 to 1000 fevals. Beyond this value, the behavior is erratic, showing a significant decrease in performance. It is worth mentioning that after 40,000 fevals, which corresponds to the maximum budget granted (see Table 5.1), there is no migration. Thus, subpopulations evolve in complete isolation. The IGD^+ indicator (to be minimized) shows alike results, having the best performance in the range $[8, 100]$. Using 4 objectives, the highest variability occurs in the interval $[1, 30]$, and the best results occur in $[50, 200]$ for both indicators. For 6, 8 and 10 objectives, the effect of the migration frequency suffers less variability beyond 20 fevals (see the ranges of the y-axis). The common range of better values is in $[100, 1000]$ regarding both performance indicators. In all cases, for the migration frequency parameter, it is observed that high rates of migration (less than 10 fevals) are harmful, and the absence of migration reduces performance.

The next parameter to analyze is the number of copies that are sent or received at each migration event. The range in the x-axis is from zero to 10 individuals since we are dealing with very small subpopulations. Adopting the same methodology of the previous parameter, we observe in Figure 5.4 that for all objectives when there is no migration, both indicators decrease their performance. In the range $[1, 7]$, stable results occur having the best values with 2 and 3 individuals. However, from 8 individuals onwards, the performance degrades.

In traditional parallel MOEAs, the population and the total budget, regarding objective function evaluations or execution time, are divided among the islands. $\mathcal{S}$-PAMICRO handles the budget in the same way. However, the handling of the population is a little different since the subpopulation size is limited to a fixed number of individuals. Therefore, if $\mathcal{S}$-PAMICRO has a low number of islands, subpopulations will consume the budget for a long time, perhaps without having any improvement. On the other hand, if $\mathcal{S}$-PAMICRO has many islands, subpopulations will have no opportunity to converge due to an insufficient budget. This trade-off between islands and budget can be observed in Figure 5.5. The more stable results regarding the hypervolume indicator are in the intervals: $[10, 20]$, $[11, 36]$, $[25, 45]$, $[18, 28]$, $[21, 36]$ for 2, 4, 6, 8 and 10 objectives, respectively. Concerning the IGD^+ indicator, they change a little bit: $[10, 15]$, $[35, 45]$, $[43, 53]$, $[36, 51]$ for 2, 6, 8 and 10 objectives, respectively.

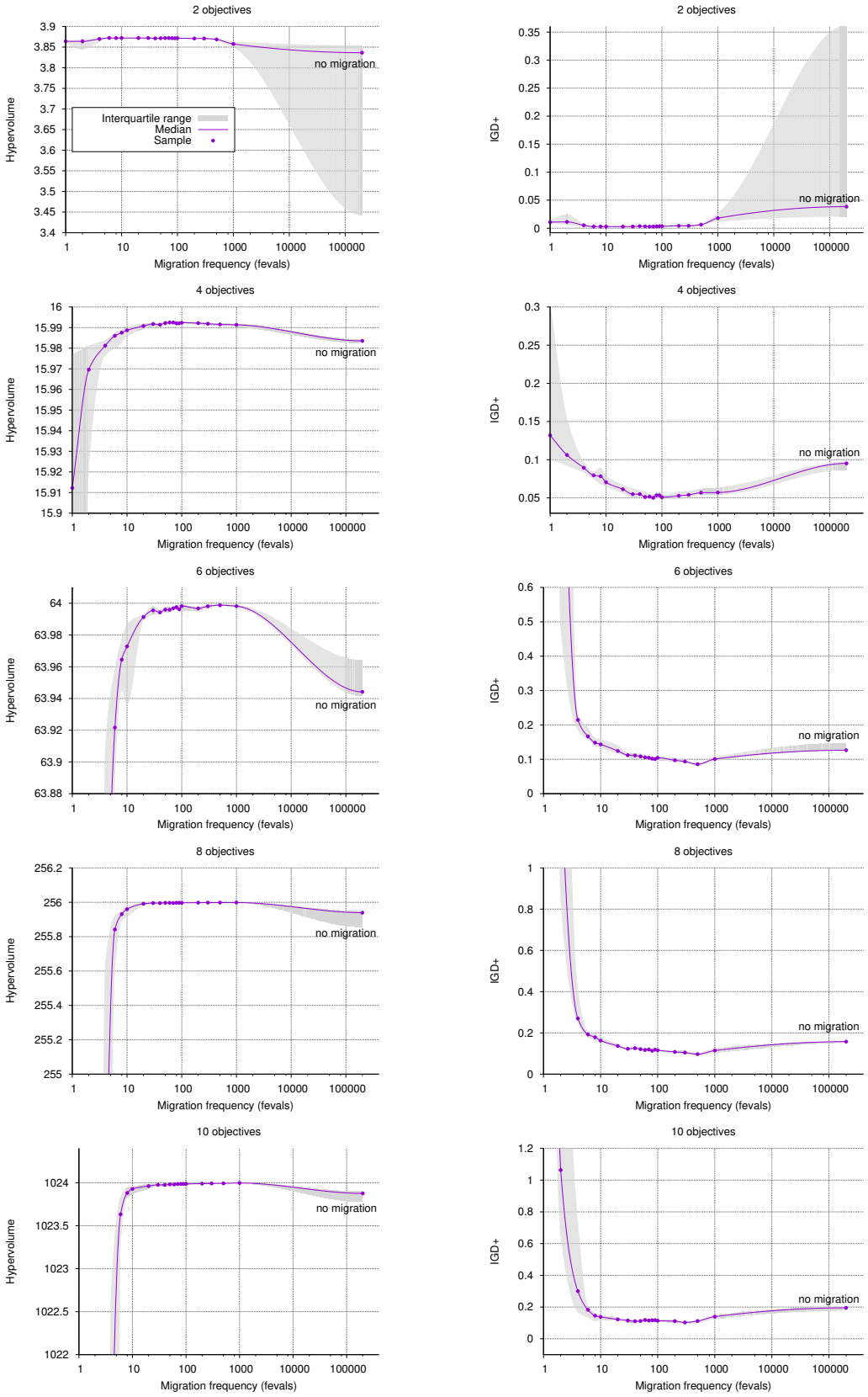


Figure 5.3: The effect of the migration frequency in $\mathcal{S}$-PAMICRO.

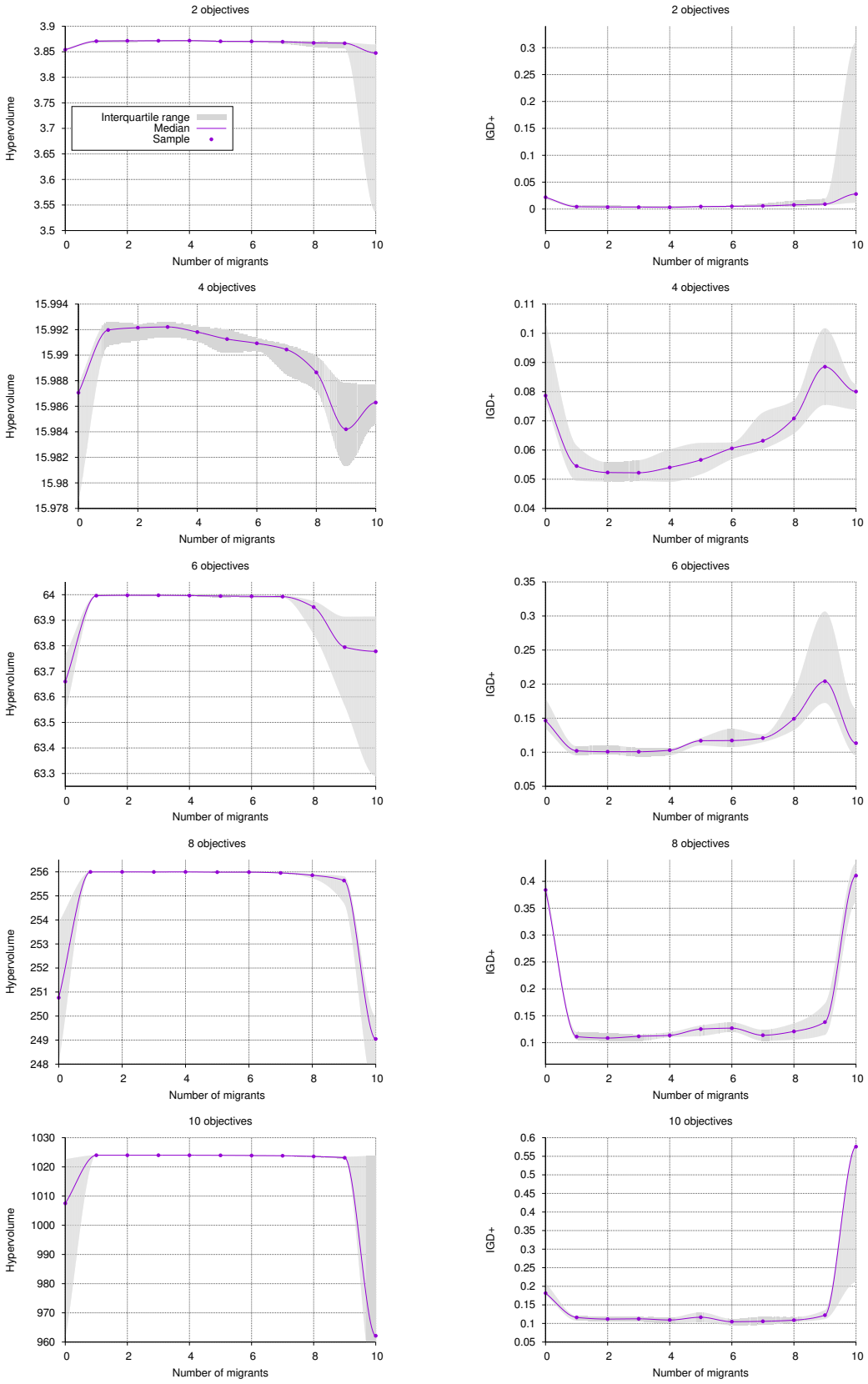


Figure 5.4: The effect of the number of migrants in $\mathcal{S}$-PAMICRO.

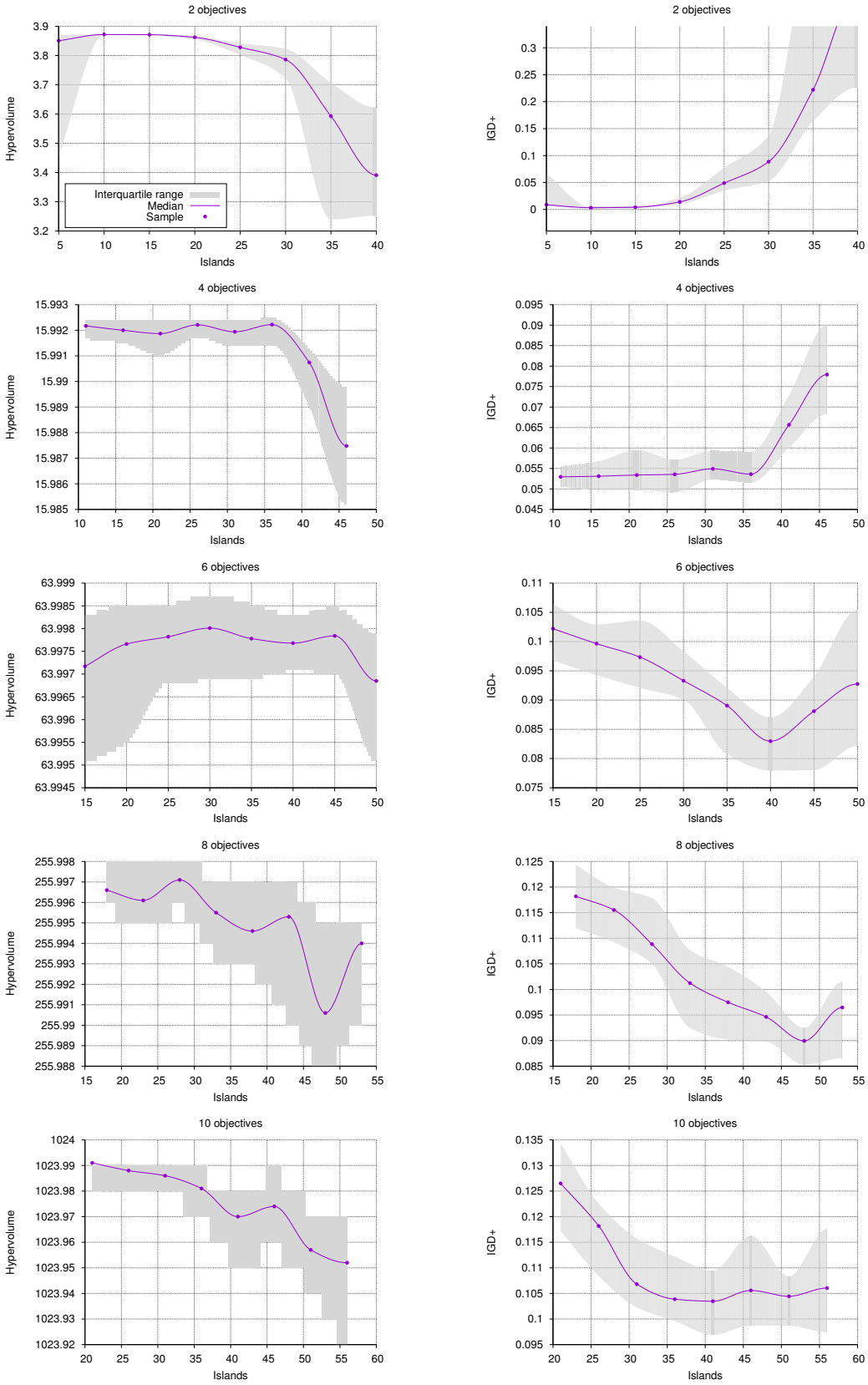


Figure 5.5: The effect of the number of islands in $\mathcal{S}$-PAMICRO.

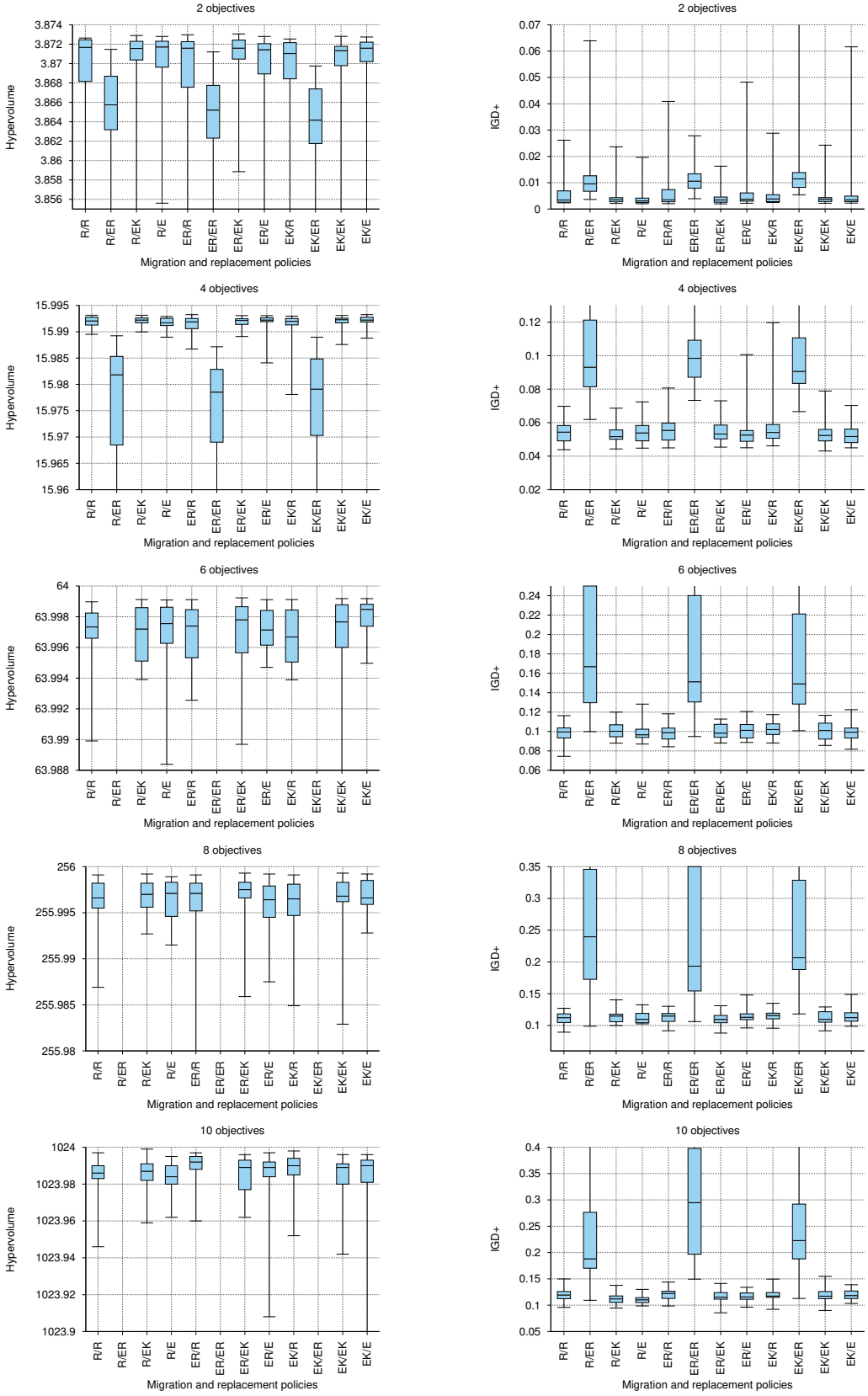


Figure 5.6: Comparison of different migration and replacement policies in $\mathcal{S}$-PAMICRO.

Table 5.2: Parameters adopted in our experiments

m	WFG		MOEAs	pMOEAs		$feval$	$\mathcal{S}$-PAMICRO
	n	k	$\|P\|$	$\|P\|$	$nisland$		γ
2	24	4	100	10	10	40,000	3
3	24	4	120	10	12	50,000	2
5	47	8	196	11	18	50,000	2
10	105	18	276	11	25	80,000	2

The behavior of the migration and replacement policies are studied together, making combinations of the different possibilities presented in Section 2.2.3 (page 28). In total, we consider three migration schemes: Random (R), Elitist Random (ER) and Elitist Ranking (EK); and four replacement schemes: Random (R), Elitist Random (ER), Elitist Ranking (EK), and Elitist (E). Therefore, in Figure 5.6 for 2 objectives the rightmost boxplot corresponds to the elitist ranking migration (EK) and elitist replacement (E). As a common factor for all objectives, the worst behaved schemes are R/ER, ER/ER and EK/ER (for the hypervolume values with 6, 8 and 10 objectives box-plots are outside the range, much farther down). Concerning the other schemes, there is no significant evidence that one combination excels over the others. The top 5 schemes with better performance indicators and lower variability are ER/EK, ER/R, R/E, EK/E, and R/EK.

5.3.2 Comparison with parallel MOEAs

In this section, we investigate the effectiveness of $\mathcal{S}$-PAMICRO on the WFG test suite. We adopted the policies of random migration and the elitist ranking replacement. The decision variables (n) and the position-related parameter (k) are specified in Table 5.2. We compared the results of our proposed algorithm with respect to SMS-EMOA, its parallel version using the asynchronous island model without external archives (pSMS-EMOA), and HypE for 2, 3, 5 and 10 objectives. For HypE, the number of sampling points was fixed to 20,000 and the resolution parameter of $\mathcal{S}$-PAMICRO (γ) is shown in Table 5.2.

The stopping criterion consisted of reaching a maximum number of objective function evaluations ($feval$), limiting the execution time to no more than two hours for each run. In order to allow a fair comparison, the parameters were similar in the sequential and parallel cases. The population size $|P|$ of the sequential algorithms (SMS-EMOA/HypE) and the parallel MOEAs (pSMS-EMOA/$\mathcal{S}$-PAMICRO) are defined in Table 5.2, as well as the number of islands or processors ($nisland$) in the latter case. Here, $nisland$ is equivalent to the division of the overall population size among the micro-population size. Experiments were carried on a Cluster of 10 PCs Intel(R) Core(TM) i7 CPU 950 @ 3.07 GHz $\times$ 8 with 3.8 GB of memory. The frequency of migration, $fmig$, was set to 80 function evaluations and the number of migrants $nmig$ was set to 2 (these values were determined according to the experiment

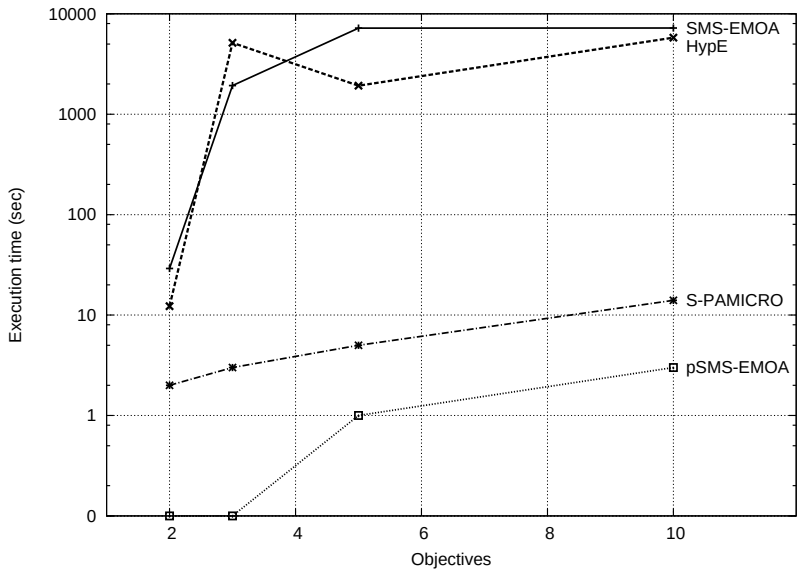


Figure 5.7: Average execution time of optimizers.

of Subsection 5.3.1). For comparing results, we adopted the hypervolume indicator, bounded by the reference points $(3, 5, 7, \ldots)$ for the instances WFG1 and WFG3; and $(2.2, 4.2, 6.2, \ldots)$ for the rest of the problems. We applied the Wilcoxon rank sum test (one-tailed) to the mean hypervolume indicator values, in order to determine whether $\mathcal{S}$-PAMICRO performed better than the other MOEAs at the significance level of 5%. Numerical results are provided in Figure 5.7 and Tables 5.3 and 5.4.

The average execution time, using a logarithmic scale for the y-axis, is shown in Figure 5.7. As it can be observed, $\mathcal{S}$-PAMICRO spent considerably less time than SMS-EMOA and HypE. For example, with 10 objectives, a run of our proposed approach took only 16 seconds out of the two hours that were allowed to the other MOEAs. Using 5 objective functions, $\mathcal{S}$-PAMICRO ended in 5 seconds, in contrast to the 26 minutes spent by HypE. Even in low dimensionality, our algorithm could reduce the running time a little bit. Furthermore, the overhead of handling the external archive in $\mathcal{S}$-PAMICRO is relatively low, compared to pSMS-EMOA that was the fastest optimizer.

On the other hand, interesting results with respect to the quality of solutions were obtained. In Tables 5.3 and 5.4, we present the hypervolume indicator values of all the experiments. An arrow pointing upwards ($\uparrow$) means that our algorithm outperformed in a significantly better way, the other MOEAs compared. Conversely, an arrow pointing downwards ($\downarrow$) means that our algorithm was significantly beaten. An asterisk ($*$) means that the algorithm was interrupted because the allowed execution time was exceeded. In the majority of the cases for 5 and 10 objectives, $\mathcal{S}$-PAMICRO obtained the best results, outperforming SMS-EMOA, HypE and pSMS-EMOA. While with 2 and 3 objectives, our proposal only surpassed pSMS-EMOA, being competitive with respect to SMS-EMOA and HypE.

In conclusion, we observed that $\mathcal{S}$-PAMICRO could achieve much better results

Table 5.3: Median and standard deviation of the hypervolume indicator on the WFG benchmark. The two best values are shown in gray scale, where a darker tone corresponds to the best value.

m	HypE			SMS-EMOA			pSMS-EMOA			$\mathcal{S}$-PAMICRO	
WFG1											
2	5.17e+00	4.11e-1	↑	4.45e+00	3.63e-1	↑	3.66e+00	2.59e-1	↑	6.61e+00	9.65e-1
3	5.66e+01	1.62e+0	↓	5.28e+01	2.50e+0	↑	4.23e+01	3.08e+0	↑	5.56e+01	3.71e+0
5	2.82e+03	1.17e+2	↑	3.18e+03	7.20e+1	∗ ↑	3.91e+03	4.83e+1	↑	5.16e+03	3.88e+2
10	4.19e+09	1.81e+8	↑	1.88e+09	2.62e+8	∗ ↑	5.28e+09	5.76e+7	↑	5.87e+09	2.33e+8
WFG2											
2	5.46e+00	2.79e-2	↑	5.47e+00	1.25e-1	↑	5.39e+00	1.71e-1	↑	5.49e+00	4.00e-2
3	5.34e+01	4.21e+0	↓	4.47e+01	4.47e+0		5.18e+01	2.00e+0	↑	5.32e+01	2.50e-1
5	4.24e+03	3.00e+2	↑	4.41e+03	3.32e+2	∗ ↑	4.66e+03	1.52e+1	↑	4.75e+03	2.00e+1
10	4.66e+09	3.22e+8	↑	3.80e+09	2.86e+8	∗ ↑	4.91e+09	1.75e+8	↑	4.93e+09	1.96e+8
WFG3											
2	1.09e+01	3.06e-2	↑	1.09e+01	2.09e-2	↑	1.08e+01	3.23e-2	↑	1.09e+01	4.50e-2
3	7.59e+01	2.19e-1	↑	7.60e+01	1.52e-1		7.48e+01	1.06e-1	↑	7.61e+01	3.61e-1
5	5.55e+03	1.55e+2	↑	6.84e+03	5.88e+1	∗ ↑	6.93e+03	3.11e+1	↑	7.22e+03	5.86e+1
10	8.37e+09	1.38e+8	↓	7.64e+09	1.95e+8	∗ ↑	5.91e+09	3.30e+8	↑	8.19e+09	1.98e+9

than SMS-EMOA and HypE in high dimensionality, spending much less computational time. For this reason, we claim that our proposed approach is a promising alternative for solving many-objective optimization problems.

5.4 Summary

This chapter presented a parallel version of the $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA). The new approach, called PArallel MICRo Optimizer based on the $\mathcal{S}$ metric ($\mathcal{S}$-PAMICRO), draws ideas from the asynchronous island model with relatively small populations. Diversity is preserved through external archives that are pruned to a limit size, using a technique based on automatic image analysis. In this chapter, we have analyzed the effects of the migration parameters on $\mathcal{S}$-PAMICRO, finding that: 1) the absence of migration reduces performance, 2) high rates of migration (less than 10 objective function evaluations) are harmful, 3) during migration, 2 or 3 individuals should be considered. This has the additional advantage that the communication costs will be low, 4) there is a trade-off between the number of islands and the budget allowable for performing function evaluations, and 5) the worst behaved replacement scheme is the elitist random. Moreover, we compared our proposal with respect to HypE, and with respect to the serial version of SMS-EMOA and another parallel version of it. We observed that $\mathcal{S}$-PAMICRO is a viable alternative for solving many-objective optimization problems at an affordable computational time. In fact, the execution time seems to be dominated by polynomial terms and not the exponential terms when using micro-populations. The model of the execution time (in seconds) of $\mathcal{S}$-PAMICRO is $1.526m - 1.632$, using least-squares approximation, where m is the number of objectives.

Table 5.4: Median and standard deviation of the hypervolume indicator on the WFG benchmark (cont'd).

m	HypE			SMS-EMOA			pSMS-EMOA			$\mathcal{S}$-PAMICRO	
WFG4											
2	2.91e+00	3.46e-3	↓	2.90e+00	1.08e-2		2.77e+00	2.05e-2	↑	2.90e+00	2.10e-2
3	2.96e+01	5.19e-2	∗ ↓	2.97e+01	5.43e-2	↓	2.66e+01	2.41e-1	↑	2.88e+01	4.45e+0
5	1.69e+03	9.10e+1	↑	2.50e+03	6.71e+1	∗ ↑	3.13e+03	7.15e+1	↑	3.47e+03	1.16e+2
10	1.86e+09	1.03e+8	∗ ↓	1.37e+09	6.15e+7	∗ ↓	2.00e+09	4.38e+8	↓	1.22e+09	5.81e+8
WFG5											
2	2.59e+00	2.40e-3	↑	2.58e+00	2.82e-3	↑	2.53e+00	1.21e-2	↑	2.59e+00	8.62e-3
3	2.74e+01	7.07e-1	∗ ↓	2.73e+01	1.38e-1	↓	2.52e+01	1.92e-1	↑	2.70e+01	1.46e-1
5	1.96e+03	1.33e+2	↑	2.47e+03	5.10e+1	∗ ↑	2.75e+03	1.50e+2	↑	3.31e+03	9.51e+1
10	1.95e+09	1.06e+8	∗ ↑	1.04e+09	3.14e+7	∗ ↑	1.04e+09	3.47e+8	↑	3.99e+09	6.24e+8
WFG6											
2	2.65e+00	5.79e-2	↑	2.64e+00	5.43e-2	↑	2.56e+00	3.93e-2	↑	2.68e+00	2.11e-2
3	2.77e+01	2.68e-1		2.79e+01	2.12e-1	↓	2.52e+01	3.86e-1	↑	2.77e+01	4.05e-1
5	1.80e+03	1.37e+2	↑	2.08e+03	7.00e+1	∗ ↑	2.93e+03	6.19e+1	↑	3.39e+03	6.23e+1
10	1.83e+09	1.28e+8	↑	9.82e+08	3.55e+7	∗ ↑	2.02e+09	2.55e+8	↑	3.83e+09	5.36e+8
WFG7											
2	2.92e+00	1.60e-3	↓	2.91e+00	1.05e-2	↓	2.84e+00	1.25e-2	↑	2.91e+00	3.05e-1
3	2.97e+01	2.72e-2	∗ ↓	2.99e+01	1.35e-2	↓	2.73e+01	2.64e-1	↑	2.93e+01	1.95e-1
5	1.82e+03	1.10e+2	↑	2.66e+03	7.07e+1	∗ ↑	3.20e+03	7.84e+1	↑	3.55e+03	4.62e+1
10	2.22e+09	1.08e+8	↓	1.26e+09	5.23e+7	∗	1.12e+09	2.77e+8		8.52e+08	7.72e+8
WFG8											
2	2.25e+00	1.46e-2	↓	2.24e+00	1.13e-2	↓	2.10e+00	2.99e-2	↑	2.24e+00	3.37e-2
3	2.34e+01	2.82e-1	↑	2.52e+01	8.04e-2	↓	2.19e+01	4.28e-1	↑	2.43e+01	5.25e-1
5	1.52e+03	1.20e+2	↑	2.26e+03	5.62e+1	∗ ↑	2.55e+03	1.16e+2	↑	2.86e+03	3.62e+2
10	1.84e+09	1.29e+8	↓	1.06e+09	4.60e+7	∗ ↓	1.53e+09	3.69e+8	↓	4.64e+08	7.71e+8
WFG9											
2	2.30e+00	2.61e-1	↑	2.78e+00	2.34e-1	↑	2.63e+00	2.09e-1	↑	2.81e+00	4.88e-1
3	2.16e+01	1.56e+0	∗ ↑	2.82e+01	1.77e+0	↓	2.25e+01	1.10e+0	↑	2.74e+01	6.78e+0
5	1.75e+03	1.65e+2	↑	2.36e+03	1.12e+2	∗ ↑	2.57e+03	6.33e+1		2.61e+03	8.93e+2
10	1.66e+09	1.10e+8	↑	1.12e+09	6.31e+7	∗ ↑	1.87e+09	3.46e+8	↑	2.31e+09	9.27e+8

Chapter 6

EMO Project

In the literature, we can find few software frameworks for evolutionary multi-objective optimization [93, 32, 125], where the majority of them rely on object oriented programming [93, 32]. Even though the ideas proposed in this paradigm are revolutionary, several multi-objective users belong to different knowledge areas, and therefore, they face difficulties when dealing with these tools. For this reason, in this Chapter, we propose a novel software framework, called **EMO Project** (Evolutionary Multi-objective Optimization Project), which highlights for its simplicity, efficiency and parallel support.

EMO Project runs on Unix-based or Unix-like[1] systems, and it is implemented in ANSI C, MPI (Message Passing Interface)[2] and Gnuplot.[3] We chose the C programming language because it supports structured programming, it has been employed to build operating system kernels, it has been the basis of many other programming languages (e.g., Java, Phyton, Matlab), and because, it is one of the fastest programming languages[4]. MPI is used for the parallelization of MOEAs and concurrent execution of commands over several processors. MPI has the advantage of supporting multi-processor scalability. On the other hand, Gnuplot is used for visualization purposes. Both MPI and Gnuplot are optional in the installation process and their functionality can be incorporated later when required. In addition, GNU Make[5] is used to build and install our free software framework, which is distributed under the GNU General Public License.[6] The main features of EMO Project are listed below:

- Currently available optimizers: SPEA2, NSGA-II, NSGA-III, MOEA/D, IBEA, SMS-EMOA, HypE, MOMBI-II, MOMBI-III, and MOVAP.

[1]For Windows users, we recommend to use Cygwin (https://www.cygwin.com) or MinGW (http://www.mingw.org).

[2]Support with two implementations: Open MPI (https://www.open-mpi.org) and MPICH (https://www.open-mpi.org)

[3]http:/www.gnuplot.info

[4] http://benchmarksgame.alioth.debian.org, https://attractivechaos.github.io/plb

[5]https://www.gnu.org/software/make

[6]http://www.gnu.org/licenses/gpl.html

- Independent implementation of almost all MOEAs. However, when the code is re-used, it is reported, referencing the original source.
- Validation of each MOEA with respect to the author's version or reported results.
- Easy incorporation of new MOPs.
- A wide range of test problems: FON1, FON2, KUR, LAU, LIS, MUR, POL, QUA, REN1, REN2, SCH1, SCH2, VIE1, VIE2, VIE3, DEB1, DEB2, DEB3, OKA1, OKA2, BNH1, BNH3, SDD, STZ1, STZ2, STZ3, ZDT, CEC09, DTLZ, EBN, LAME, MIRROR, WFG. Constrained test problems: BEL, BNH2, BNH4, JIM, KITA, OBA, OSY1, OSY2, SRN, TMK, TNK, VIE4, DTLZ8, DTLZ9. Real world applications: WATER, CAR-SIDE IMPACT, 2 BAR TRUSS.
- Setting parameters via one single configuration file.
- Structure and organization of output data.
- Log file creation with execution times.
- Flexible stopping criteria, including maximum number of function evaluations, maximum execution time or a combination of both.
- Mersenne Twister pseudorandom number generator with the huge period of $2^{19937-1}$.
- Different seeds for each run of a MOEA, loaded from a provided file.
- Real-time visualization of the progress of a MOEA.
- Various performance indicators: hypervolume, R2, GD, IGD, Δ_p, etc.
- Fastest known algorithm for calculating the hypervolume, proposed by the Walking Fish Group.[7]
- Diverse set of scalarizing (or utility) functions: CHE, PBI, ASF, etc.
- Initialization of the population at random or from a user-defined file.
- Filtering of non-dominated solutions.
- Synchronous and asynchronous island model of all the available MOEAs with configurable migration and replacement schemes.
- Concurrent execution of line-command programs over several processors.
- Summary statistics of performance indicators or any other set of observations.

[7] http://www.wfg.csse.uwa.edu.au/hypervolume

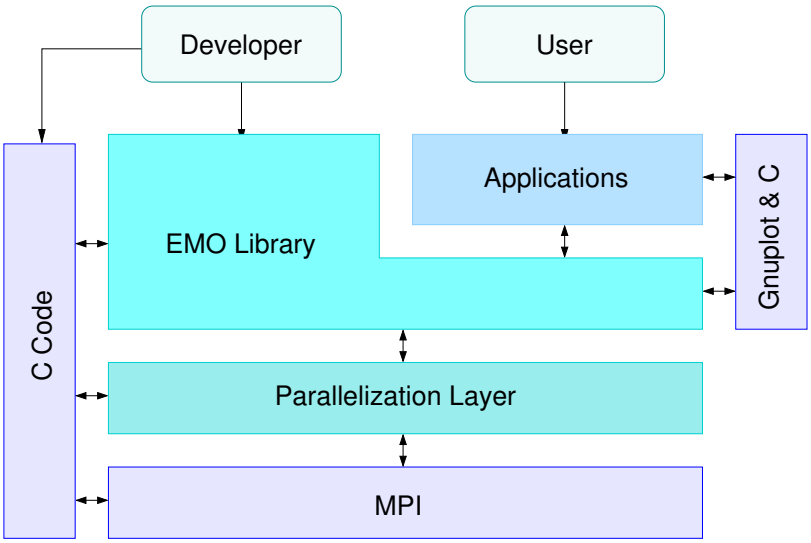


Figure 6.1: Architecture of EMO Project.

EMO Project is constituted by three main parts: the applications, the EMO library, and the parallelization layer. Figure 6.1 shows a basic diagram of how these parts fit together and interact with two special actors: the common user and the developer. The former is able to invoke predefined applications, while the second can define new problems, implement algorithms and create more applications. The organization of the remainder of this chapter is the following: Sections 6.1, 6.2, and 6.3 describe in detail the main parts of EMO Project. Section 6.4 delineates how to define MOPs in our software framework. Whereas, Section 6.5 describes how to add new MOEAs. Finally, Section 6.6 summarizes this chapter.

6.1 Applications

The applications consist of a set of command-line programs, which start with the prefix "**emo_**", and their purpose is to perform essential operations in evolutionary multi-objective optimization. Table 6.1 shows the list of available applications in EMO Project. When they are invoked without arguments a help message will appear about their usage syntax. In the remainder of this section, we describe each command adopting the following conventions. An upper case item represents a variable parameter to be specified by the user. Items enclosed in brackets $[\cdot]$ are optional. Items enclosed in curly brackets $\{\cdot\}$ denote a list of options, from which only one should be selected by the user.

The command **emo_moea** invokes a specific optimizer for solving a multi-objective optimization problem. Its syntax is as follows:

```
emo_moea MOEA parameter_file {MOP, default} runs [initial_population_file(s)]
```

The argument `MOEA` stands for the name of the optimizer, such as NSGA2, SPEA2, MOMBI3, MOVAP, etc. The argument `parameter_file` is a file that contains the

Table 6.1: Available command applications in EMO Project.

Command	Purpose
emo_moea	Solve a multi-objective optimization problem through an evolutionary algorithm.
emo_indicator	Evaluate the performance indicator of a solution set.
emo_stat	Retrieve statistics from a sample.
emo_ndset	Filter the non-dominated solutions form a set.
emo_refpoint	Determine the reference points from a solution set.
emo_norm	Normalize a solution set.
emo_task	Parallelize a list of Unix commands over a set of processors.
emo_pmoea	Parallelize the solution of a multi-objective optimization problem through an evolutionary algorithm.

parameters of the optimizer, the MOP, as well as the environment. Comments in the configuration file start with `#`, the list of parameters are separated by a new line, and the assignment follows the pattern `parameter = value`. Parameter files are located in the directory `EMO_Project/demo/input` having the extension `cfg`. An example of a parameter file is shown next.

```
#####################################################################
# Param_02D.cfg        Configuration file of EMO Project      #
#                      for two objectives.                    #
#####################################################################
# File that contains the seeds of the random number generator.
# In sequential MOEAs a different seed is used for each run,
# while in parallel MOEAs, each processor gets one distinct
# seed for each run.
seed = ./input/seed.dat

# Population size
psize = 100

######################### MOP definition ########################
# Values are ignored when using the 'default' option or when
# the MOP is not scalable w.r.t the number of objectives,
# decision variables or inequality constraints.

# Number of decision variables (0 default value)
nvar = 0

# Number of objective functions
nobj = 2

# Number of inequality constraints (0 default value)
ncon = 0

###################### WFG1-9 test problems #####################
# Number of decision variables
```

```
wfg_nvar = 24

# Number of position-related parameters
wfg_npos = 20

########## Test Problems Based on Lame Superspheres ###########
# Only apply for problems EBN, LAME and MIRROR.

# Pareto front shape (1: linear, <1: convex, >1: concave)
lame_gamma = 6.0

# Difficulty of the problem
# 0: unimodal, 1: unimodal with equidistant local Pareto fronts,
# 2: many-to-one mappings
lame_difficulty = 2

############ Tanaka constrained problems (TNK1, TNK2) ##########
# In general, (a,b) controls the length of the continuous
# region of the Pareto front.
#
# By increasing the value of a, the length of the ‘‘cuts’’
# become deeper requiring the search to proceed along a
# narrower corridor. A suggested value is 0.1
tnk_a = 0.1
# The number of disconnected regions in the Pareto front
# increase for a bigger value of b.
# 16 and 32 are some suggested values.
tnk_b = 16.0
###################### Variation operators #####################
# Crossover probability
pc = 0.9

# Mutation probability (by default -1 is equivalent to pm = 1/nvar)
pm = -1

# Crossover distribution index (SBX)
nc = 20

# Mutation distribution index (Real polynomial mutation)
nm = 20

###################### Stopping condition ######################
# This section defines the stopping condition of the program,
# given by the number of function evaluations, execution
# time (in seconds), or a combination of them.

feval = 30000
#time = 350

########################## Plotting ############################
# Animation of the evolution of approximation sets in objective
# space. This option is disabled when MPI is active.
```

```
# Frequency of plotting (given in function evaluations)
# -1 value disables plotting
#  0 value plots at the end
plot_freq = 500

# file's name (inside of quotation marks) or mathematical
# expression (according to Gnuplot) of the true Pareto front.
# It will be omitted, if it is left blank.
# Examples: "~/fronts/SEQ_ZDT2_02D.pof.nd", 1-sqrt(x)
plot_pftrue =

# Terminal type in Gnuplot (wxt, x11, etc.)
plot_term = x11

########################### Output ############################
# Output directory
output = ./output

# Debug (0 disabled, 1 enabled)
debug = 1

############ Utility (or Scalarization) Functions #############
# List of available utility functions for a particular MOEA.
# The 2 at the end of some functions means that each component
# of a weight vector is used as a divisor and not as a factor.
#
# { weighted_compromise_programming, weighted_compromise_programming2,
# weighted_power, weighted_power2, exponential_weighted_criteria,
# exponential_weighted_criteria2, weighted_product,
# weighted_product2, weighted_sum, weighted_sum2, weighted_norm,
# weighted_norm2, least_squares, least_squares2, chebyshev,
# augmented_chebyshev, augmented_chebyshev2, modified_chebyshev,
# modified_chebyshev2, achievement_scalarizing_function,
# augmented_achievement_scalarizing_function,
# penalty_boundary_intersection, two_level_penalty_boundary_intersection,
# quadratic_penalty_boundary_intersection, general_scalarizing_function,
# general_scalarizing_function2, normalized_scalarizing_function,
# normalized_scalarizing_function2, conic_scalarization,
# conic_scalarization2, vector_angle_distance_scaling,
# vector_angle_distance_scaling2, didass, didass2 }
#
moead_utility = chebyshev

mombi2_utility = achievement_scalarizing_function

# The reference point is the nadir point (0 disabled, 1 enabled)
utility_inverted = 0

# Weight vector file (NSGA-III, MOMBI-II, MOMBI-III, R2-IBEA, MOEA/D)
wfile = ./input/weight/weight_02D_99.sld
```

```
# Parameters of the model of each utility function

# H is the partition in Simplex Lattice Design method (SLD),
# parameter H for two_level_penalty_boundary_intersection
# and quadratic_penalty_boundary_intersection
utility_H = 99

# Penalization paramter for penalty_boundary_intersection
# and quadratic_penalty_boundary_intersection
utility_theta = 5

# Penalization parameters for two_level_penalty_boundary_intersection
utility_theta1 = 0.1
utility_theta2 = 10

# Parameter for augmented_chebyshev[2],
#  modified_chebyshev[2],
#  augmented_achievement_scalarizing_function,
#  two_level_penalty_boundary_intersection,
#  quadratic_penalty_boundary_intersection,
#  general_scalarizing_function[2],
#  normalized_scalarizing_function[2]
#  and conic_scalarization[2]
utility_alpha = 0.02

# Parameter for weighted_compromise_programming[2],
#  weighted_power[2], exponential_weighted_criteria[2],
#  weighted_norm[2] and vector_angle_distance_scaling[2]
utility_p = 200

# Parameter for general_scalarizing_function[2] and didass[2]
utility_beta = 0.1

# Parameter for didass[2]
utility_gamma = 0.1,0.1

############################ MOEA/D ############################
# Normalization of objective functions, useful when objectives
# are in different scale (0 disabled, 1 enabled)
moead_norm = 0

# Number of the weight vectors in the neighborhood of each
# weight vector.
moead_niche = 20

# Original version of MOEA/D, see \cite{Zhang07b}
# (0 disabled, 1 enabled)
moead_ver2007 = 1

# Variant of MOEA/D, see \cite{Li09c}
# the parameters moead_delta and moead_nr
# are ignored when moead_ver2007 = 1.
```

```
# The probability that parent solutions are selected from the neighborhood
moead_delta = 0.9

# The maximal number of solutions replaced by each child solution,
# it should be in the range [1, moead_niche]
moead_nr = 2

########################## SMS-EMOA ############################
# Scaling factor for the reference point in the hypervolume
# indicator, assuming objectives are normalized:
# zref = sfactor * (1,1,...,1)
smsemoa_sfactor = 20
# Incremental IWFG algorithm for calculating hypervolume
# contributions (see \cite{Cox16}), otherwise it uses the naive
# approach using the WFG algorithm (see \cite{While12}).
smsemoa_iwfg = 1

########################## MOMBI-II ############################
# Number of generations in which the nadir point is stored
mombi2_record = 5

# Threshold of variances for the nadir vector
mombi2_alpha = 0.5

# Tolerance threshold
mombi2_epsilon = 1e-3

# Incorporate preferences in order to guide the search to the
# regions of main interest. This option will be omitted, if left blank.
# refpoint should be an infeasible point in R^nobj, where its entries
# are separated by comma and it can be set from previous executions.
# Examples:
#  (zdt3) 0.547,-0.49
#  (pol)  0.8472,20.6106
#  (kur)  -19.99,-11.99
mombi2_refpoint =

############################## IBEA ############################
# Reference point for the variant that uses the hypervolume
# indicator
hvibea_rho = 1.1

# Scaling factor for the variants that uses the epsilon and
# hypervolume indicators
ibea_kappa = 0.05

# Scaling factor for the variant that uses the R2 indicator
# For the variant that is based on the R2 indicator
r2ibea_kappa = 0.005

############################## HypE ############################
```

```
# Number of sampling points used in Monte Carlo simulation
# (with the -1 value calculates the exact hypervolume)
hype_samples = 10000

# Reference point
hype_bound = 2

############################## MOVAP ############################
# Image resolution
movap_xrs = 3

############## Parallel MOEAs - Island Model ##################
# Logical topology of connection
# topology = {line, ring, star, tree, full, torus, torusd, mesh}
topology = ring
# Only for tree topology
degree = 2
# Only for torus and torusd topologies
torus_row = 2
torus_col = 4
# Only for mesh topology
mesh_file =

# communication flow
# comm = {uni, bidi, scatter, gather}
# The default flow in full topology is bidirectional
comm = uni

# Enable synchronous communication (0 = disable, 1 = enable)
sync = 0

# External interruption (0 = disable, 1 = enable)
interrupt = 0

# Migration policy
# migra = {random, elitist_random, elitist_ranking,
#          front, front_random, front_ranking}
mpolicy = random

# Replacement policy
# repla = {random, elitist_random, elitist_ranking, elitist}
rpolicy = elitist_ranking

# pMOEA
# Migration frequency in the island model
# (recommended 8 times the population size)
epoch = 100

# pMOEA
# Number of individuals to migrate in the island model
# subscript indicates the rank of the process
migrant = 2
```

```
#################################################################
```

The third argument of **emo_moea** refers to the MOP to be solved. In this case, the user can opt for a predefined test problem, such as FON1, POL, ZDT1, OKA1, BNH1, etc. Their specification can be found in the EMO Book[8] [21]. Another option is to solve a user-defined problem using the argument `default`. Here, the MOP is defined in `EMO_Project/demo/emo_moea.c:myMOP_eval` (see Section 6.4 for more details).

The argument `run` specifies how many times the MOEA will be executed using different seeds. The next arguments are optional and indicate the prefixes of the files containing the initial population for each execution run. Here, the prefix omits the filename extension. If only one prefix is provided, this will be used for all the runs.

The final results of **emo_moea** will be written in the directory specified by the `output` parameter in the configuration file, under the following names:

`MOEA_MOP_##D.log` Log file of events registered during the MOEA execution

`MOEA_MOP_##D.sum` Summary of results per execution run

`MOEA_MOP_##D_R##.pos` Decision variables

`MOEA_MOP_##D_R##.pof` Objective functions

`MOEA_MOP_##D_R##.con` Constraint values (if applicable)

`##D` indicates the number of objectives using a format with two digits, `R##` denotes the execution run with also two digits. The file `log` keeps a register of the parameter values that were read from the configuration file. It also stores unexpected errors, developer-defined messages, as well as the function evaluations and execution time required per run. Since the writing of the log file can be a time-consuming and storage demanding task, this functionality can be disabled from the configuration file setting the parameter `debug = 0`. The file `sum` summarizes the number of function evaluations and execution time for each run. On the other hand, the files with extensions `pos`, `pof` and `con` correspond to the final population produced by the selected MOEA, and they are structured as follows:

```
# 10 2
1.408662e-04 9.881314e-01
9.065329e-02 6.989141e-01
2.141310e-02 8.536679e-01
4.158174e-01 3.551610e-01
1.651340e-01 5.936332e-01
8.057537e-01 1.023803e-01
6.475280e-01 1.953107e-01
2.872088e-01 4.640814e-01
9.998090e-01 9.553892e-05
5.407637e-01 2.646353e-01
```

[8] https://www.cs.cinvestav.mx/~emoobook

The first row corresponds to the header, which indicates the number of rows and columns contained in the file. Components are separated by a space, whereas solutions are separated by a new line. This is the *standard layout* embraced by EMO Project for input and output files.

If Gnuplot is installed and the parameter `plot_freq` is greater than zero, then an animation of the evolution of the population in objective space will be shown during the execution of **emo_moea**. If this parameter is set to zero, only the final population will be presented. We conclude our description of **emo_moea** with some examples of its usage.

- The following instruction solves DTLZ1 for three objectives using SMS-EMOA:

  ```
  emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 1
  ```

- This instruction solves the same problem, but with NSGA-III and recycling the final population of SMS-EMOA. Note that the optimizer is executed ten times.

  ```
  emo_moea NSGA3 input/Param_03D.cfg DTLZ1 10 output/SMS_EMOA_DTLZ1_03D_R01
  ```

- The next line solves a user-defined problem with SPEA2:

  ```
  emo_moea SPEA2 input/Param_02D.cfg default 1
  ```

The command **emo_indicator** calculates the performance indicator of a solution set, usually in objective space (`pof` files). Its syntax is as follows:

```
emo_indicator INDICATOR {file, prefix} number_of_runs [options...]
```

The argument `INDICATOR` refers to the name of the performance indicator, which can be the hypervolume (HV), GD, IGD$^+$, Δ_p, $R2$, spacing (SP), etc. The second argument stands for the file that contains the solution set or the common filename prefix corresponding to several executions. These files should have the standard layout of EMO Project. The third argument indicates the number of executions to be processed. The argument `options` varies according to the performance indicator. Therefore, when an argument is required, **emo_indicator** provides a hint:

```
emo_indicator HV {prefix,file} number_of_runs reference_point [ALGORITHM]
emo_indicator GD {prefix,file} number_of_runs p_norm reference_set_file
emo_indicator IGD {prefix,file} number_of_runs p_norm reference_set_file
emo_indicator IGD+ {prefix,file} number_of_runs reference_set_file
emo_indicator Deltap {prefix,file} number_of_runs p_norm reference_set_file
emo_indicator EPS* {prefix_A,file_A} number_of_runs {prefix_B,file_B}
emo_indicator EPS+ {prefix_A,file_A} number_of_runs {prefix_B,file_B}
emo_indicator MAXIMIN {prefix_A,file_A} number_of_runs
emo_indicator R2 {prefix,file} #runs weight_filename UTILITY_FUNCTION
emo_indicator ONVG {prefix_A,file_A} number_of_runs
emo_indicator C {prefix_A,file_A} number_of_runs {prefix_B,file_B}
emo_indicator SP {prefix_A,file_A} number_of_runs
emo_indicator S-ENERGY {prefix_A,file_A} number_of_runs
```

When `number_of_runs` is one, the performance indicator value is shown in the standard output, and when it is for more runs, the result is also stored in the file `prefix.INDICATOR`. Some examples of usage of the command **emo_indicator** are provided next.

- Compute the hypervolume of a Pareto set using the reference point $(0.7, 0.7, 0.7)$:

```
> emo_indicator HV output/NSGA3_DTLZ1_03D_R01.pof 1 0.7 0.7 0.7
1 output/NSGA3_DTLZ1_03D_R01.pof 0.316578
```

- Calculate the s-energy of ten MOEA runs:

```
> emo_indicator S-ENERGY output/NSGA3_DTLZ1_03D 10
Data stored in output/NSGA3_DTLZ1_03D.s-energy
> cat output/NSGA3_DTLZ1_03D.s-energy
# 10 1
5.267043e+05
4.402657e+05
4.417264e+05
1.822245e+06
1.485764e+06
4.401744e+05
4.723929e+05
6.044821e+05
4.424452e+05
4.383740e+05
```

The command **emo_stat** determines the statistics from a set of samples, being useful to compare the performance of several optimizers. Its syntax is as follows:

```
emo_stat {max,min} data_file(s)
```

The first argument indicates if the data should be minimized or maximized. Each data file must contain one column. For example, the following instruction obtains the statistics of the s-energy for ten MOEA runs:

```
> emo_stat min output/NSGA3_DTLZ1_03D.s-energy
#id|file|min|max|median|mean|std|var|q1|q2|q3|iqr|imin|imax|imedian|samples
1|output/NSGA3_DTLZ1_03D.s-energy|4.383740e+05|1.822245e+06|4.424452e+05|
                                 7.114574e+05|4.798922e+05|2.302965e+11|
                                 4.402657e+05|4.574191e+05|6.044821e+05|
                                 1.642164e+05|10|4|9|10
#rank|id|1|1|1|1|1
```

The results are displayed in the standard output. Here, the fields are separated by the character | and the statistics of a data file are separated by a new line.

The first line is the header, which contains the name of the columns. `id` represents an identification number of the data file. `min`, and `max` are the minimum, and maximum values, respectivley. `std` is the standard deviation, `var` is the variance, `q1-3` are the first, second, and third quartiles. `iqr` is the interquartile range. `imin`, `imax`,

and `imedian` are the samples corresponding to the minimum, maximum, and median values, respectively. `samples` represents the total number of samples in the data file. The last line of the file is the tail, and it contains the identification number of the best statistics for the minimum, maximum, median, mean, and standard deviation.

```
> emo_stat min output/NSGA2_DTLZ1_03D.s-energy output/NSGA3_DTLZ1_03D.s-energy
  output/MOEAD_DTLZ1_03D.s-energy
#id|file|min|max|median|mean|std|var|q1|q2|q3|iqr|imin|imax|imedian|samples
1|output/NSGA2_DTLZ1_03D.s-energy|1.283365e+06|7.372916e+08|2.171375e+08|
                                  2.246065e+08|2.093779e+08|4.383912e+16|
                                  4.024848e+07|2.201429e+08|2.722729e+08|
                                  2.320244e+08|7|4|1|10
2|output/NSGA3_DTLZ1_03D.s-energy|4.383740e+05|1.822245e+06|4.424452e+05|
                                  7.114574e+05|4.798922e+05|2.302965e+11|
                                  4.402657e+05|4.574191e+05|6.044821e+05|
                                  1.642164e+05|10|4|9|10
3|output/MOEAD_DTLZ1_03D.s-energy|4.321494e+05|4.409231e+05|4.382830e+05|
                                  4.381763e+05|2.403019e+03|5.774499e+06|
                                  4.372996e+05|4.385977e+05|4.399242e+05|
                                  2.624600e+03|9|10|8|10
#rank|id|3,2,1|3,2,1|3,2,1|3,2,1|3,2,1|3,2,1
```

The command **emo_ndset** filters the non-dominated solutions from a set. Its syntax is as follows:

```
emo_ndset file
```

The result will be written in `file.nd`, for example:

```
> emo_ndset output/NSGA3_DTLZ1_03D_R01.pof
Data stored in output/NSGA3_DTLZ1_03D_R01.pof.nd
```

The command **emo_refpoint** retrives the minimum, maximum, ideal and nadir points from a solution set. Its syntax is as follows:

```
emo_refpoint file
```

The result will be shown in the standard output:

```
> emo_refpoint output/NSGA3_DTLZ1_03D_R01.pof
output/NSGA3_DTLZ1_03D_R01.pof
zmin: 0.000000 0.000000 0.000000
zmax: 0.501256 0.528085 0.501018
ideal: 0.000000 0.000000 0.000000
nadir: 0.501256 0.528085 0.501018
```

The command **emo_norm** adjusts the values of a solution set in the interval $[0, 1]$. Its syntax is as follows:

```
emo_norm file minimum_point maximum_point
```

The output will be written in `file.norm`:

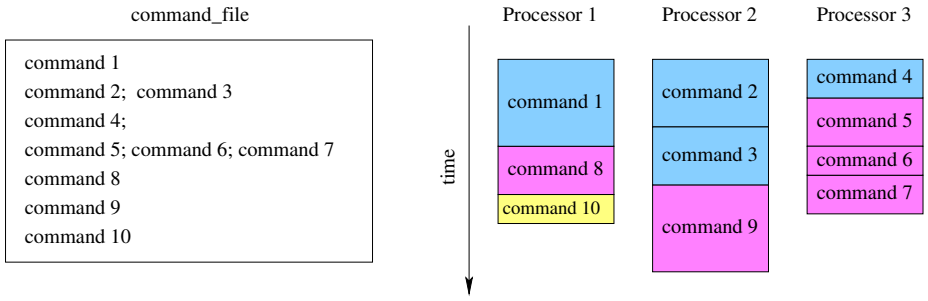


Figure 6.2: `mpirun -np 4 emo_task command_file`

```
> emo_norm output/NSGA3_DTLZ1_03D_R01.pof 0.0 0.0 0.0 0.501256 0.528085 0.501018
File created: output/NSGA3_DTLZ1_03D_R01.pof.norm
```

The command **emo_task** simultaneously executes a list of unix commands over a set of available processors. This command requires MPI to be installed. Its syntax is as follows:

```
mpirun -np num_procs emo_task command_file
```

The argument `num_procs` indicates the number of processors. If `num_procs` is set to one, the commands are processed sequentially. When `num_procs` is greater than one, it should be considered that a master processor manages the task distribution, while `num_procs` -1 slaves execute the commands. `command_file` is a file that contains the list of unix commands. Independent commands are separated by a new line, whereas dependent commands are separated by a semicolon. Comments start with `#` and they are ignored by **emo_task**.

Figure 6.2 illustrates how **emo_task** works. In this case, ten commands are processed simultaneously using four processors (processor 0 is the master). `command 3` depends on the completion of `command 2`. Thus, they are processed sequentially by the same assigned processor. Similarly occurs with `commands 5-7`. Commands are distributed among processors using a Round-robin scheme. Therefore, each processors has equal chance to attend tasks. When there is no available processor, the master waits until one is free. The list of completed commands and their result status are stored in the file `command_file.done`. A result different from zero means that the command execution was not successful. Moreover, `command_file` can be edited after the launch of **emo_task**. Thus, the master will process new or modified commands, as long as the last ones are not yet completed. Log files, named `Proc_#.log`, are generated for each processor in order to track tasks, where `#` stands for the processor identifier. Next, we show an example.

```
> cat command_file
emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_indicator HV output/SMS_EMOA_DTLZ1_03D 3 0.7 0.7 0.7
emo_moea NSGA3 input/Param_03D.cfg DTLZ1 5; gzip output/NSGA3_DTLZ1_03D*
emo_moea SPEA2 input/Param_03D.cfg DTLZ1 1
> mpirun -np 3 emo_task command_file
> cat command_file.done
DONE:result 0:emo_moea NSGA3 input/Param_03D.cfg DTLZ1 5; gzip output/NSGA3_DTLZ1_03D*
DONE:result 0:emo_moea SPEA2 input/Param_03D.cfg DTLZ1 1
DONE:result 0:emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_indicator HV output/SMS_EMOA_DTLZ1_03D 3 0.7 0.7 0.7
> cat Proc_0.log
1|2018/02/13 13:47:44|pid 0 => Start of the debug
2|2018/02/13 13:47:44|pid 0 => Rank: 0, MPI version: 3, subversion: 1
3|2018/02/13 13:47:44|pid 0 => Master|send to slave 1|emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_indicator HV...
4|2018/02/13 13:47:44|pid 0 => Master|send to slave 2|emo_moea NSGA3 input/Param_03D.cfg DTLZ1 5; gzip output/NSGA3_D...
5|2018/02/13 13:47:44|pid 0 => Master|there are no available slaves, waiting for one
6|2018/02/13 13:48:02|pid 0 => Master|recv from 2|DONE:result 0:emo_moea NSGA3 input/Param_03D.cfg DTLZ1 5; gzip outp...
7|2018/02/13 13:48:02|pid 0 => Master|send to slave 2|emo_moea SPEA2 input/Param_03D.cfg DTLZ1 1|result 0
8|2018/02/13 13:48:02|pid 0 => End of file command_file
9|2018/02/13 13:48:02|pid 0 => Master|send to 1|END|result 0
10|2018/02/13 13:48:02|pid 0 => Master|send to 2|END|result 0
11|2018/02/13 13:48:07|pid 0 => Master|recv from 2|DONE:result 0:emo_moea SPEA2 input/Param_03D.cfg DTLZ1 1
12|2018/02/13 13:50:05|pid 0 => Master|recv from 1|DONE:result 0:emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_...
13|2018/02/13 13:50:05|pid 0 => End of the debug
> cat Proc_1.log
1|2018/02/13 13:47:44|pid 1 => Start of the debug
2|2018/02/13 13:47:44|pid 1 => Rank: 1, MPI version: 3, subversion: 1
3|2018/02/13 13:47:44|pid 1 => Slave|recv from master|emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_indicator HV...
4|2018/02/13 13:47:44|pid 1 => Slave|executing emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_indicator HV output...
5|2018/02/13 13:50:05|pid 1 => Slave|send to master|DONE:result 0:emo_moea SMS_EMOA input/Param_03D.cfg DTLZ1 3; emo_...
6|2018/02/13 13:50:05|pid 1 => Slave|recv from master|END|result 0
7|2018/02/13 13:50:05|pid 1 => Slave|halt
8|2018/02/13 13:50:05|pid 1 => End of the debug
```

When processors are distributed in a collection of machines or hosts, **emo_task** can be invoked as follows:

```
mpirun --hostfile host_file emo_task command_file
```

where `host_file` is a file containing the names of the hosts. Each host is included on a separate line, and hosts should be reachable by ssh. For example, three processors are launched for `machineA`, one for `machineB`, and two for `machineC`:

```
> cat host_file
machineA
machineA
machineA
machineB
machineC
machineC
```

The command **emo_pmoea** parallelizes the execution of a MOEA using the island model. This command requires MPI to be installed. Its syntax is as follows:

```
mpirun -np num_procs emo_pmoea MOEA parameter_file {MOP, default} runs
                               [initial_population_file(s)]
or

mpirun --hostfile host_file emo_pmoea MOEA parameter_file {MOP, default} runs
                                    [initial_population_file(s)]
```

The argument `num_procs` indicates the number of islands or processors. When processors are distributed in a collection of hosts, the second instruction can be adopted. Here, `host_file` contains the name of the hosts (for more details see the description of **emo_task**). The remainder arguments of **emo_pmoea** are the same as those in **emo_moea**.

The configuration file must include parameters related to the island model, such as: logical topology of connection (`topology`), synchronous (`sync = 1`) or asynchronous communication (`sync = 0`), migration (`mpolicy`) and replacement policy (`mpolicy`), migration frequency (`epoch`), as well as the number of individuals to migrate (`migrant`). **emo_pmoea** equally divides the population (`psize`) and the function evaluations (`feval`) among the islands. The output files include the processor identifier before the name of the MOP:

`pMOEA_#_MOP_##D.log` Log file of events registered during the MOEA execution

`pMOEA_#_MOP_##D.sum` Summary of results per execution run

`pMOEA_#_MOP_##D_R##.pos` Decision variables

`pMOEA_#_MOP_##D_R##.pof` Objective functions

`pMOEA_#_MOP_##D_R##.con` Constraint values (if applicable)

As an example, the next instruction executes the island version of NSGA-II using five processors:

```
mpirun -np 5 emo_pmoea NSGA2 input/Param_02D.cfg DTLZ1 1                                [initial_p
```

6.2 EMO Library

The EMO library is defined in the `emo.h` header, and it is composed of built-in functions, as well as structured data types. Built-in functions are thread-safe, and do not allocate dynamic memory during their execution. Next, we list them.

Multi-objective optimization

EMO_maxBound[2]	Determine the maximum point of a set of points.
EMO_minBound	Determine the minimum point of a set of points.
EMO_maxminBound	Determine the minimum and maximum points of a set of points.
EMO_findminBound	Find the points that belong to the minimum point.
EMO_findmaxminBound	Find the points that belong to the maximum point.
EMO_Dominance_weak	Check weak dominance relationship between solutions.
EMO_Dominance_strict	Check strict dominance relationship between solutions.
EMO_Dominance_strong	Check strong dominance relationship between solutions.
EMO_Dominance_alpha[2]	Check α dominance relationship between solutions.
EMO_Dominance_favor	Check favor relationship between solutions.
EMO_Dominance_incomparable	Check if two solutions are incomparable to each other according to the given dominance relation.
EMO_Dominance_indifferent	Check if two solutions are indifferent.
EMO_Dominance_constraint[2,3]	Check if a solution constraint dominates another one.
EMO_Dominance_feasible	Check if a solution is feasible or not.
EMO_Dominance_ndset	Filter the non-dominated solutions from a set according to a given dominance relation.
EMO_Indicator_gd	Calculate the generational distance of a solution set.
EMO_Indicator_igd_plus	Calculate the IGD^+ of a solution set.
EMO_Indicator_deltap	Calculate the Δ_p of a solution set.
EMO_Indicator_r2	Calculate the $R2$ indicator of a solution set.
EMO_Indicator_senergy	Calculate the s-energy indicator of a solution set.
...	...

Vector operations

EMO_vnorm	Calculate the norm of a vector.
EMO_vmul	Calculate the multiplication of a vector by a scalar.
EMO_vsum	Calculate the sum of a vector by a scalar.
EMO_vdot	Calculate the dot product.
EMO_vproj	Calculate the projection of a vector onto another vector.
EMO_vadd	Calculate the addition of two vectors.
EMO_vdiff	Calculate the difference of two vectors.
EMO_vorth	Calculate the orthogonal vector between a point and a line.
EMO_vdist	Calculate the distance between two points.
EMO_vzero	Return the zero vector.
EMO_isvzero	Check if the components of a vector are zero.
EMO_vaxes	Provide the axes of a Cartesian coordinate system.
EMO_vcopy	Copy the components of a vector into another vector.
EMO_vprint[i]	Write the components of a vector to the given stream.

Matrix operations

EMO_matmul	Calculate the multiplication of two matrices.
EMO_minverse	Calculate the inverse of a matrix.
EMO_mprint	Write the elements of a matrix to the given stream.

Statistics

EMO_mean	Determine the mean of a data sample in $\mathbb{R}$.
EMO_median	Determine the median of a data sample in $\mathbb{R}$.
EMO_quartile	Determine the q1, q2, q3 quartiles, and the interquartile range.
EMO_var	Calculate the variance.
EMO_std	Calculate the standard deviation.
EMO_[d]min	Determine the minimum value.
EMO_[d]max	Determine the maximum value.

Miscelaneous functions

EMO_File_read	Read a set of elements in $\mathbb{R}^k$ from a file.
EMO_File_write	Write a set of elements in $\mathbb{R}^k$ to a file.
EMO_quicksort	Sort a set of elements in $\mathbb{R}^k$ according to the given criterion.
EMO_Dictionary_find	Find the index position of a word into an array of strings.
EMO_Dictionary_print	Write the array of strings to the given stream.
EMO_Parser_get_token	Get the next token from a string.
EMO_shift	Translates a set of elements in $\mathbb{R}^k$ in a given direction.
EMO_toupper	Convert lowercase string to uppercase.
EMO_tolower	Convert uppercase string to lowercase.
EMO_trim	Remove spaces from the left and right of a string.

The structured data types in EMO Project are employed for data organization, where we adopt the philosophy to allocate dynamic memory once at the beginning of the program. This action allows us to invoke recurrently functions that manipulate such structures without any extra computational overhead. At the end of the program, the allocated memory is released. Moreover, structs are passed through functions by reference, avoiding copies onto the stack. The function prototypes that accomplish this task, for a given component, are:

```
EMO_Component_alloc(EMO_Component *comp, ...);
EMO_Component_run(EMO_Component *comp, ...);
EMO_Component_free(EMO_Component *comp);
```

Another technique that helps to improve efficiency is the way the population is represented. As shown in Figure 6.3, the variables of the parent and offspring populations are stored in contiguous blocks of memory, allowing fast access and memory copy. This also allows to send and receive data over MPI as a single block.

In the following, we describe the structures of the EMO Library. The related functions are thread-safe.

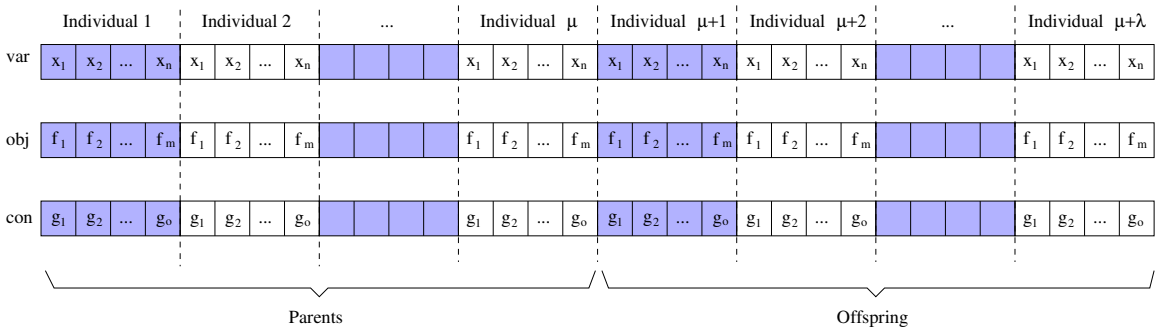


Figure 6.3: Population representation in EMO Project.

EMO_Population

Members	
double *var	Array of decision variables of all individuals.
double *obj	Array of objective function values of all individuals.
double *con	Array of constraint values of all individuals.
int *vio	Flag that indicates if an individual violates any constraint.
double *cv	Constraint violation value.
int mu	Parent population size.
int lambda	Offspring population size.
int size	Total population size ($\mu + \lambda$)
Functions	
EMO_Population_alloc	Allocate memory for the members of the structure using the given MOP's information and the sizes μ, and λ.
EMO_Population_init	Randomly initialize a population.
EMO_Population_init_from_file	Initialize a population from the given files: `pos`, `pof`, and `con` (if applicable).
EMO_Population_write	Write a population to the files: `pof`, `pof`, and `con` (if applicable).
EMO_Population_copy	Copy individual i to the current position of individual j.
EMO_Population_copy2	Copy one population to another population.
EMO_Population_swap	Interchange individual i by j in the population.
EMO_Population_survive	Keep the selected μ individuals in the first half of the arrays var, obj and con.
EMO_Population_evaluate	Evaluate the population.
EMO_Population_constrain	Calculate the constraint violation value.
EMO_Population_free	Free memory of the structure.

EMO_MOP

Members	
int nvar	Number of decision variables.
int nobj	Number of objective functions.
int ncon	Number of inequality constraints.
double *xmin, *xmax	Bounds on the decision variables.
void (*f)(struct EMO_MOP *, double *, double *)	Pointer to the MOP's evaluation function having no inequality constraints.

void (*fc)(struct EMO_MOP *, double *, double *, double *)	Pointer to the MOP's evaluation function having inequality constraints.
name	MOP's name.
feval	Current number of MOP's evaluations.
Functions	
EMO_Benchmark_alloc	Allocate memory and initialize the EMO_MOP structure using the given name of the test problem and parameters.
EMO_Benchmark_zdt1	Evaluate a solution on the ZDT1 test problem.
EMO_Benchmark_dtlz1	Evaluate a solution on the DTLZ1 test problem.
EMO_WFG_wfg1	Evaluate a solution on the WFG1 test problem.
...	...
EMO_Benchmark_free	Free memory of the EMO_MOP structure.

EMO_List

Members	
int size	Current number of elements.
int max_size	Maximum number of elements allowed to store.
Functions	
EMO_List_alloc	Allocate memory for a maximum number of elements.
EMO_List_clear	Remove all elements from a list.
EMO_List_queue	Add an element at the end of a list.
EMO_List_dequeue	Extract the first element from a list.
EMO_List_add	Insert an element at the given position.
EMO_List_remove	Delete the first occurrence that matches with the given element.
EMO_List_retrieve	Extract the element from the given position.
EMO_List_get	Get the element at the given position.
EMO_List_seek	Find the position of a given element.
EMO_List_count	Count how many times an element appears in a list.
EMO_List_append	Add the elements of a list to another list.
EMO_List_move	Remove an element from a list and add it to another list.
EMO_List_move_all	Remove all elements from a list and add them to another list.
EMO_List_print	Write the contents of a list to the given stream.
EMO_List_free	Free memory of the structure.

EMO_SMSEMOA

EMO_SMSEMOA_alloc	Allocate memory for the members of the structure.
EMO_SMSEMOA_run	Solve a MOP using SMS-EMOA and the specified parameters.
EMO_SMSEMOA_free	Free memory of the structure.

EMO_MOEA

EMO_MOEA_alloc	Allocate memory for a generic MOEA.
EMO_MOEA_run	Solve a MOP using the specified MOEA and parameters.
EMO_MOEA_free	Free memory of the structure.

EMO_Stop

EMO_Stop_alloc	Allocate memory for the members of the structure.
EMO_Stop_start	Prepare the stopping criterion.
EMO_Stop_set_feval	Enable the stopping criterion by setting a maximum number of function evaluations.
EMO_Stop_set_time	Enable the stopping criterion by setting a time limit.
EMO_Stop_set_fitness	Enable the stopping criterion by setting a limit fitness.
EMO_Stop_set_epoch	Enable the stopping criterion by setting epochs (used in pMOEAs).
EMO_Stop_update_epoch	Update current epoch according to the spend MOP's evaluations after initialization of the population.
EMO_Stop_now	Activate immediate interruption.
EMO_Stop_end	Indicate if a MOEA should stop its execution.
EMO_Stop_free	Free memory of the structure.

EMO_NDSort

Members	
int nfront	Total number of fronts.
EMO_List *front	Array of (nfront) fronts.
int *rank	Rank for each individual.
Functions	
EMO_NDSort_alloc	Allocate memory for the members of the structure.
EMO_NDSort_run[2]	Execute the non-dominated sorting algorithm of NSGA-II.
EMO_NDSort_free	Free memory of the structure.

EMO_HV

EMO_HV_alloc	Allocate memory for the members of the structure.
EMO_HV_run[2]	Compute the hypervolume indicator using the implementation of the Walking Fish Group (WFG).
EMO_HV_contribution	Compute the hypervolume contributions of a set of solutions using a naive approach.
EMO_HV_free	Free memory of the structure.

EMO_IWFG

EMO_IWFG_alloc	Allocate memory for the members of the structure.
EMO_IWFG_run	Execute the incremental IWFG algorithm for identifying the smallest hypervolume-contributor of a set of non-dominated solutions.
EMO_IWFG_free	Free memory of the structure.

EMO_Debug

EMO_Debug_alloc	Allocate memory for the log file.
EMO_Debug_rename	Rename the log file.
EMO_Debug_print{f,v}	Write a message in the log file.
EMO_Debug_error	Write an error message in the log file.
EMO_Debug_free	Free memory of the structure.

EMO_Param

EMO_Param_alloc	Allocate memory for the members of the structure.
EMO_Param_alloc_from_file	Allocate memory for the members of the structure and initialize parameters from a given file.
EMO_Param_init	Initialize output files and plot display.
EMO_Param_save	Save parameters in log files.
EMO_Param_set	Set a parameter and its value.
EMO_Param_get_int	Retrieve the int value of a given parameter name.
EMO_Param_get_double	Retrieve the double value of a given parameter name.
EMO_Param_get_char	Retrieve the string value of a parameter name.
EMO_Param_get_vector_double	Retrieve the double array of a parameter name.
EMO_Param_free	Free memory of the structure.

EMO_Rand

EMO_Rand_alloc	Allocate memory for the members of the structure.
EMO_Rand_alloc_from_file	Allocate and initialize the random number generation from a file.
EMO_Rand_next_seed	Read the next seed from a file and reset the random number generator. If indicated, discard certain number of seeds.
EMO_Rand_prob1	Generate a random number on the real interval [0,1].
EMO_Rand_prob2	Generate a random number on the real interval [0,1).
EMO_Rand_prob3	Generate a random number on the real interval (0,1).
EMO_Rand_real1	Generate a random number on the real interval [a,b].
EMO_Rand_real2	Generate a random number on the real interval [a,b).
EMO_Rand_real3	Generate a random number on the real interval (a,b).
EMO_Rand_int1	Generate a random number on the discrete interval [a,b].
EMO_Rand_int2	Generate a random number on the discrete interval [a,b).
EMO_Rand_int3	Generate a random number on the discrete interval (a,b).
EMO_Rand_flip	Flip a coin based on a given probability.
EMO_Rand_shuffle	Generate a random permutation using the Fisher-Yates algorithm.
EMO_Rand_box_muller	Generate a Gaussian random number.
EMO_Rand_free	Free memory of the structure.

EMO_Utility

EMO_Utility_alloc	Allocate memory for the members of the structure.
EMO_Utility_asf	Compute the achievement scalarizing function for a given solution.
EMO_Utility_pbi	Compute the penalty boundary intersection for a given solution.
EMO_Utility_ws	Compute the weighted sum for a given solution.
...	...
EMO_Utility_free	Free memory of the structure.

EMO_Plot

EMO_Plot_alloc	Allocate memory for the members of the structure.
EMO_Plot_run	Plot the given Pareto front.
EMO_Plot_free	Free memory of the structure.

6.3 Parallelization Layer

The parallelization layer is defined in the `emo_mpi.h` header, and it is composed of four structured data types.

EMO_Task

EMO_Task_alloc	Allocate memory for the members of the structure.
EMO_Task_run	Execute a set of unix commands, given in a file, over a set of available processors.
EMO_Task_free	Free memory of the structure.

EMO_Migration

EMO_Migration_alloc	Allocate memory for the members of the structure.
EMO_Migration_get_type	Determine the migration scheme.
EMO_Replacement_get_type	Determine the replacement scheme.
EMO_Migration_random	Select a given number of random individuals for migration or replacement.
EMO_Migration_elitist_random	Select a given number of random individuals from the non-dominated set. If there are not enough members, the rest is selected at random from the remainder of the population.
EMO_Migration_elitist_ranking	Select a given number of random individuals from the Pareto front. If there are not enough members, the rest is selected from the successively ranked Pareto fronts.
EMO_Replacement_elitist_random	Replace a given number of random solutions, which are dominated by the immigrants.
EMO_Replacement_elitist_ranking	Rank all Pareto fronts and replace individuals from the worst ranked front(s) with the immigrants.
EMO_Replacement_elitist	Combine immigrants with the current population, rank all Pareto fronts and remove individuals from the worst ranked front(s).
EMO_Migration_free	Free memory of the structure.

EMO_Island

EMO_Island_alloc	Allocate memory for the members of the structure.
EMO_Island_reset	Reset communications (used for different pMOEA runs).
EMO_Island_send	Send copies of individuals to destiny islands.
EMO_Island_receive	Receive individuals from source islands.
EMO_Island_free	Free memory of the structure.

EMO_PMOEA

EMO_PMOEA_alloc	Allocate memory for a generic parallel MOEA.
EMO_PMOEA_run	Solve a MOP using the island version of a given MOEA and parameters.
EMO_PMOEA_free	Free memory of the structure.

6.4 User-defined Problems

In order to define a new MOP, two functions should be edited in the files:

- `EMO_Project/demo/emo_moea.c`
- `EMO_Project/demo/emo_pmoea.c`

The former is for solving the problem with a sequential MOEA, and the latter is for solving the problem with a parallel MOEA.

The function `myMOP_alloc` assigns memory and initializes a MOP structure, providing the name, the number of decision variables, objective functions, and constraints. The function `myMOP_eval` represents the MOP's function evaluation and must not allocate dynamic memory to avoid latency. Next, we show these functions:

```
/**** Definition of a new Multi-objective Optimization Problem (MOP) ***

    minimize   {f_0(x), f_1(x), ..., f_{mop->nobj-1}(x)}

    subject to g_0(x) >= 0
               g_1(x) >= 0
               ...
               g_{mop->ncon - 1}(x) >= 0

               x \in [x_min, x_max]

 f: vector of objective functions
 g: vector of inequality constraints
 x: vector of decision variables

 Note: the following function prototype should be defined for a
       MOP with inequality constraints:

 void myMOP_eval(EMO_MOP *mop, double *f, double *g, double *x)
```

```
 A non feasible solution is that one which for any i g_i(x) < 0.

*/
void myMOP_eval(EMO_MOP *mop, double *f, double *x) {
  f[0] = x[0] * x[0];
  f[1] = pow(x[0] - 1.0, 2.0);
}

void myMOP_alloc(EMO_MOP *mop) {
  int i;

  /* MOP's name */
  mop->name = (char *) malloc(sizeof(char) * 200);
  strcpy(mop->name, "WING_DESIGN");

  /* Number of decision variables */
  mop->nvar = 2;

  /* Number of objectives */
  mop->nobj = 2;

  /* Box constraints */
  mop->xmin = (double *) calloc(sizeof(double), mop->nvar);
  mop->xmax = (double *) calloc(sizeof(double), mop->nvar);

  for(i = 0; i < mop->nvar; i++) {
    mop->xmin[i] = 0.0;
    mop->xmax[i] = 1.0;
  }

  /* Required */
  mop->npos = 0;
  mop->feval = 0;
  mop->coding = EMO_REAL;

  /* MOP without inequality constraints */
  mop->ncon = 0;
  mop->f = myMOP_eval;

  /* MOP with inequality constraints */
  //mop->ncon = #;
  //mop->fc = myMOP_eval;
}
```

After these changes, the user should recompile the project and execute the command `emo_moea` or `emo_pmoea` with the `default` option as a MOP's name.

6.5 Adding New MOEAs

With the aim to include a new evolutionary algorithm in EMO Project, developers need to follow four simple steps.

1. Define the structure of the new optimizer, for example `EMO_SMSEMOA`, in the file `EMO_Project/src/smsemoa.h`, and encode the functions `EMO_SMSEMOA_alloc`, `EMO_SMSEMOA_run`, and `EMO_SMSEMOA_free` in the file `EMO_Project/src/smsemoa.c`.

2. Include the definitions of the new algorithm in the generic MOEA structure. For this purpose, the files `EMO_Project/src/moea.h`, and `EMO_Project/src/moea.c` should be edited for adding the following marked lines:

```
/* File EMO_Project/src/moea.h */
#ifndef _MOEA_H
#define _MOEA_H

#include "common.h"
#include "param.h"
#include "smsemoa.h"       /* NEW ALGORITHM */
#include "nsga2.h"
#include "nsga3.h"
#include "mombi2.h"
#include "mombi3.h"
#include "ibea.h"
#include "moead.h"
#include "spea2.h"
#include "hype.h"
#include "movap.h"

/* Generic MOEA */
typedef void (*EMO_MOEA_falloc)(void *, EMO_Param *,
               EMO_Population *, EMO_MOP *);
typedef void (*EMO_MOEA_ffree)(void *);
typedef void (*EMO_MOEA_frun)(void *, EMO_Param *,
               EMO_Population *, EMO_MOP *);

typedef struct {
  EMO_MOEA_falloc alloc;
  EMO_MOEA_ffree free;
  EMO_MOEA_frun run;

  union {
    EMO_SMSEMOA smsemoa;   /* NEW ALGORITHM */
    EMO_NSGA2 nsga2;
    EMO_NSGA3 nsga3;
    EMO_MOMBI2 mombi2;
    EMO_MOMBI3 mombi3;
    EMO_IBEA ibea;
    EMO_MOEAD moead;
    EMO_SPEA2 spea2;
```

```
    EMO_HYPE hype;
    EMO_MOVAP movap;
  } alg;

  void *palg;
} EMO_MOEA;

void EMO_MOEA_alloc(EMO_MOEA *moea, EMO_Param *param,
                    EMO_Population *pop, EMO_MOP *mop, const char *str);
void EMO_MOEA_free(EMO_MOEA *moea);
void EMO_MOEA_run(EMO_MOEA *moea, EMO_Param *param,
                  EMO_Population *pop, EMO_MOP *mop);
#endif


/* File EMO_Project/src/moea.c */

#include <string.h>
#include "string2.h"
#include "moea.h"

const char *EMO_MOEA_list[] = {
  "SMS_EMOA",                    /* NEW ALGORITHM */
  "NSGA2",
  "NSGA3",
  "MOMBI2",
  "MOMBI3",
  "HV_IBEA",
  "EPS_IBEA",
  "R2_IBEA",
  "MOEAD",
  "SPEA2",
  "HYPE",
  "MOVAP",
  NULL };

void EMO_MOEA_alloc(EMO_MOEA *moea, EMO_Param *param,
                    EMO_Population *pop, EMO_MOP *mop, const char *str) {

  const EMO_MOEA_falloc a[] = {
    (EMO_MOEA_falloc) EMO_SMSEMOA_alloc,   /* NEW ALGORITHM */
    (EMO_MOEA_falloc) EMO_NSGA2_alloc,
    (EMO_MOEA_falloc) EMO_NSGA3_alloc,
    (EMO_MOEA_falloc) EMO_MOMBI2_alloc,
    (EMO_MOEA_falloc) EMO_MOMBI3_alloc,
    (EMO_MOEA_falloc) EMO_IBEA_alloc,
    (EMO_MOEA_falloc) EMO_IBEA_alloc,
    (EMO_MOEA_falloc) EMO_IBEA_alloc,
    (EMO_MOEA_falloc) EMO_MOEAD_alloc,
    (EMO_MOEA_falloc) EMO_SPEA2_alloc,
    (EMO_MOEA_falloc) EMO_HYPE_alloc,
    (EMO_MOEA_falloc) EMO_MOVAP_alloc };
```

```
const EMO_MOEA_ffree f[]  = {
  (EMO_MOEA_ffree) EMO_SMSEMOA_free,    /* NEW ALGORITHM */
  (EMO_MOEA_ffree) EMO_NSGA2_free,
  (EMO_MOEA_ffree) EMO_NSGA3_free,
  (EMO_MOEA_ffree) EMO_MOMBI2_free,
  (EMO_MOEA_ffree) EMO_MOMBI3_free,
  (EMO_MOEA_ffree) EMO_IBEA_free,
  (EMO_MOEA_ffree) EMO_IBEA_free,
  (EMO_MOEA_ffree) EMO_IBEA_free,
  (EMO_MOEA_ffree) EMO_MOEAD_free,
  (EMO_MOEA_ffree) EMO_SPEA2_free,
  (EMO_MOEA_ffree) EMO_HYPE_free,
  (EMO_MOEA_ffree) EMO_MOVAP_free };

const EMO_MOEA_frun r[]   = {
  (EMO_MOEA_frun) EMO_SMSEMOA_run,     /* NEW ALGORITHM */
  (EMO_MOEA_frun) EMO_NSGA2_run,
  (EMO_MOEA_frun) EMO_NSGA3_run,
  (EMO_MOEA_frun) EMO_MOMBI2_run,
  (EMO_MOEA_frun) EMO_MOMBI3_run,
  (EMO_MOEA_frun) EMO_IBEA_run,
  (EMO_MOEA_frun) EMO_IBEA_run,
  (EMO_MOEA_frun) EMO_IBEA_run,
  (EMO_MOEA_frun) EMO_MOEAD_run,
  (EMO_MOEA_frun) EMO_SPEA2_run,
  (EMO_MOEA_frun) EMO_HYPE_run,
  (EMO_MOEA_frun) EMO_MOVAP_run };

char *aux;
int i;

aux = EMO_toupper(str);
i = EMO_Dictionary_find(EMO_MOEA_list, aux);

if(i == -1) {
  printf("Error, unknown MOEA %s in EMO_MOEA_alloc.\n", aux);
  free(aux);
  exit(1);
}

moea->alloc = a[i];
moea->free = f[i];
moea->run = r[i];

switch(i) {
  case 0:  moea->palg = &moea->alg.smsemoa;    /* NEW ALGORITHM */
           break;
  case 1:  moea->palg = &moea->alg.nsga2;
           break;
  case 2:  moea->palg = &moea->alg.nsga3;
           break;
  case 3:  moea->palg = &moea->alg.mombi2;
```

```
            break;
    case 4:  moea->palg = &moea->alg.mombi3;
            break;
    case 5: case 6: case 7:
            moea->palg = &moea->alg.ibea;
            break;
    case 8:  moea->palg = &moea->alg.moead;
            break;
    case 9:  moea->palg = &moea->alg.spea2;
            break;
    case 10: moea->palg = &moea->alg.hype;
            break;
    case 11: moea->palg = &moea->alg.movap;
            break;

    default: printf("Error, unknown MOEA (2) %s.\n", aux);
            free(aux);
            exit(1);
  }

  moea->alloc(moea->palg, param, pop, mop);
  free(aux);
}


void EMO_MOEA_free(EMO_MOEA *moea) {
  moea->free(moea->palg);
}

void EMO_MOEA_run(EMO_MOEA *moea, EMO_Param *param,
                  EMO_Population *pop, EMO_MOP *mop) {
  moea->run(moea->palg, param, pop, mop);
}
```

3. Update the structure of `EMO_MOEA` in the file `EMO_Project/include/emo.h`. Besides, add the structure of `EMO_SMSEMOA` and its prototype functions.

4. Incorporate a new target in the Makefiles before compilation:

```
# EMO_Project/src/Makefile and EMO_Project/demo/Makefile

OBJS_NOMPI        = moead.o nsga3.o smsemoa.o.

smsemoa.o:        smsemoa.h smsemoa.c
                $(CC) $(CFLAGS) -c smsemoa.c
```

If the new optimizer requires new parameters, they should be included in the configuration file. With these four steps, we guarantee that the parallel version of the new algorithm will be available in the command `emo_pmoea`.

6.6 Summary

In this chapter, we have presented EMO Project, a free software framework for evolutionary multi-objective optimization, which is implemented in ANSI C, MPI and Gnuplot. EMO Project highlights for its simplicity, efficiency and parallel support. It can be used by employing a set of command-line programs for regular users or as a toolkit with predefined libraries for advanced users. Through the chapter, we have presented the architecture of EMO Project, illustrating how to implement new problems and algorithms. Furthermore, EMO Project has support to parallelization of MOEAs, and concurrent execution of commands over several processors. Our proposal is available for download at:

`http://computacion.cs.cinvestav.mx/~rhernandez`

Chapter 7

Real World Applications

There are a wide range of real-world applications of multi-objective optimization in different areas, such as engineering, science, industry, and medicine [92, 20, 119, 115]. Most of these applications involve equality ($h_j(\mathbf{x}) = 0$), inequality ($g_i(\mathbf{x}) \geq 0$), and box constraints ($x_k^l \leq x_k \leq x_k^u$). In practice, these constraints define material characteristics, regulations, physical conditions and logical restrictions, among others [82]. The way multi-objective evolutionary algorithms deal with box constraints is by avoiding them, i.e., all generated individuals are within the valid bounds of the decision variables. This task is ensured by the variation operators. Nevertheless, equality and inequality constraints require a special treatment. Before solving a MOP, each equality constraint must be transformed into two inequalities with different signs. Then, all inequalities are changed to the same direction (see the MOP definition on page 9) by multiplying -1 in both sides. It is said that a solution $\mathbf{x} \in \mathcal{X}$ is *feasible* if all constraints are satisfied, or *infeasible* if at least one is not fulfilled.

Constraints can make MOPs even more complicated since their mathematical formulations may lack of linearity, convexity or differentiability. Besides, constraints may demarcate small or disconnected feasible regions, where objective vectors are not always available for infeasible solutions. For these reasons, it is essential to consider an effective constraint-handling technique when designing optimizers. In the literature, we can find several approaches, where the majority have been adapted from single-objective optimization [100, 21, 82, 81, 101].

In this chapter, we successfully adopt an easy-to-implement constraint-handling technique that, although it can be incorporated to any of our proposals, is applied to MOMBI-III (see page 39). The organization of the remainder of this chapter is the following: Section 7.1 describes the constraint-handling technique. Section 7.2 outlines the experiments. Section 7.3 validates this approach on three difficult test problems with two and three objectives. Then, we investigate the performance on three engineering design optimization problems: the car side-impact problem in Section 7.4, which is defined by three objectives and ten constraints; the water resources planning in Section 7.5, which is a five-objective problem with seven constraints; and the design of an analog integrated circuit in Section 7.6, which involves eight objective functions with more than thirty constraints. Finally, Section 7.7 summarizes this chapter.

7.1 Constraint-Handling Technique

The adopted constraint-handling technique consists in dividing the population into two subsets according to the feasibility and infeasibility of solutions. Each subset receives special treatment. These changes are addressed in the main loop of MOMBI-III, which is presented in Algorithm 11. In line 1, the population is initialized uniformly at random. After that, it is evaluated according to the MOP definition. At each iteration, new individuals are created from parents selected uniformly at random (lines 4 and 5). These individuals are evaluated in line 6. Next, the feasible solutions from the union of parents and offspring are segregated in subset Q. Depending on the cardinality of this set, we proceed as follows:

- If there are more feasible solutions than the population size, then the individuals from $\left\{P \bigcup P'\right\} \setminus Q$ are discarded, and the next population is selected from Q as done in the unconstrained version of MOMBI-III (lines 9-12). The $R2$ Ranking and Reduce procedures were described in Chapter 3 (page 44).

- In case there are less feasible solutions than the population size, then all infeasible solutions are stored in set Q' (line 15). Next, the constraint violation (CV) is calculated as the number of constraints that were not satisfied for each solution $q' \in Q'$. In lines 17 and 18, the set Q' is sorted regarding the CV values in increasing order, removing the $|P'|$ individuals having the highest values. Therefore, in line 19, the next generation is formed with the feasible solutions (Q) and the best infeasible solutions (Q').

- In case the number of feasible solutions matches exactly the population size, then set Q becomes the next generation.

The principal advantages of this method are: 1) it does not require extra parameters, 2) the computational complexity remains the same, and 3) it is applicable even for an infeasible solution with invalid objective function values. This latter scenario commonly happens in simulators as a result of undefined operations on floating point numbers, such as dividing by zero or numerical overflow.

7.2 Experimental Methodology

In all the experiments of this chapter, we compared our MOMBI-III against NSGA-II, NSGA-III, and MOEA/D with the constraint handling techniques that were proposed for them in the literature. In the following, we describe such techniques and define the parameters used by the algorithms.

For NSGA-II, Deb et al. [30] extended the preference relation $\prec_n$, used when comparing solutions in the parent and survival selection mechanisms. Here, a solution p is said to *constrait-dominate* a solution q if any of the following conditions is true: 1) p is feasible and q is not; 2) p and q are both infeasible, but p has a smaller constraint

Algorithm 11 Main loop of MOMBI-III for handling constraints

Input: MOP, stopping criterion, set of weight vectors W, set of heuristics H
Output: Final population P

Initialize population P at random
Evaluate MOP for each element $p \in P$
while the stopping criterion is not satisfied **do**
  Select random parents from P
  $P' \leftarrow$ Generate offspring using variation operators
  Evaluate MOP for each $p' \in P'$
  $Q \leftarrow$ Filter feasible solutions from $P \bigcup P'$
  if $|Q| > |P|$ **then**
    Update the reference points $\mathbf{z}^{\min}$ and $\mathbf{z}^{\max}$ from Q
    Normalize objective functions by setting
    $q.\mathbf{y} \leftarrow \frac{q.\mathbf{y} - \mathbf{z}^{\min}}{\mathbf{z}^{\max} - \mathbf{z}^{\min}}$, $\forall q \in Q$, where $q.\mathbf{y} \in \mathbb{R}^m$
    $R \leftarrow R2$ Ranking (Q, W, H)
    $P \leftarrow$ Reduce $(Q, R, |P|)$
  else
    if $|Q| < |P|$ **then**
      $Q' \leftarrow$ Filter infeasible solutions from $P \bigcup P'$
      $CV[q'] \leftarrow |\{i \in \{1, 2, \ldots, o\} : q'.g_i(\mathbf{x}) < 0\}|$ for all $q' \in Q'$
      Sort Q' w.r.t. CV in increasing order
      Remove from Q' the last $|P'|$ solutions
      $P \leftarrow Q \bigcup Q'$
    else
      $P \leftarrow Q$
return P

violation; 3) p and q are feasible, but $p \prec q$. The *constraint violation* of a solution p is calculated as follows:

$$CV[p] := \sum_{i=0}^{o} \langle g_i(p.\mathbf{x}) \rangle, \tag{7.1}$$

where the bracket operator $\langle \alpha \rangle$ returns $|\alpha|$ if $\alpha < 0$ and returns zero, otherwise.

For NSGA-III, Jain and Deb [52] suggested to normalize constraints by dividing them by a constant term. For example, given $g_i(\mathbf{x}) \geq b_i$, the normalized constraint function becomes $g'_i(\mathbf{x}) = g_i(\mathbf{x})/b_i - 1 \geq 0$, and similarly for equality constraints obtaining the term $h'_j(\mathbf{x})$. Then, the constraint violation for an individual p is given by:

$$CV[p] := \sum_{i=0}^{o} \langle g'_i(p.\mathbf{x}) \rangle + \sum_{j=0}^{r} |h'_j(p.\mathbf{x})|. \tag{7.2}$$

Expression (7.2) is considered to check constraint domination in both selection mechanisms. Nonetheless, in the binary tournament of the parent selection, when both solutions are feasible, a random solution is chosen.

Table 7.1: Parameters adopted for the problems with constraints

Problem	m	$\|P\|$	Objective function evaluations	Crossover		Mutation		Weight vector partitions	MOEA/D niche
				probability	distr. index	probability	distr. index		
TNK1	2	100	25,000	0.9	20	$1/n$	20	99	20
OSY2	2	100	40,000	0.9	20	$1/n$	20	99	20
VIE4	3	106	40,000	0.9	20	$1/n$	20	13	20
Car side-impact	3	156	78,000	0.9	20	$1/n$	20	16	20
Water resources planning	5	212	212,000	1.0	30	$1/n$	20	6	42
Design of analog integrated ciruit	8	250	50,000	0.9	20	$1/n$	15	n/a	20

For MOEA/D, although there is a constraint version proposed by its developers [75], we decided to adopt the method described by Jain and Deb [52], named C-MOEA/D. The main reason was that the first approach relies on a penalty function that requires the proper setting of two scaling parameters for a given MOP. Another disadvantage of the technique in [75] is that it assumes that objective values are always available for infeasible solutions. Nevertheless, this is not the case for the problem of the analog circuit design (Section 7.6), which may generate invalid numbers. On the other hand, C-MOEA/D uses the same comparison principle of NSGA-III. Here, in the survival selection mechanism, when a child and a neighbor are both feasible, the scalarizing function is used instead of Pareto dominance.

Regarding the parameters employed in the algorithms, they are shown in Table 7.1. The scalarizing function in C-MOEA/D was CHE for two objectives and PBI with $\theta = 5$ for the remaining objectives. As suggested by the authors [148], since the objectives are in different scales in the design of the analog circuit, the population of C-MOEA/D was normalized during its evolution. The parameter models adopted for MOMBI-III are shown in Table 2.3 (page 20).

For NSGA-III, C-MOEA/D and MOMBI-III the weight vectors were generated using the Uniform Design method [146] regarding the problem of the analog circuit design. Here, the number of weights is the same as the population size. In the remaining problems, weight vectors were created by the Simplex-Lattice Design method (see page 23) with a partition parameter shown in Table 7.1. The variation operators were Polynomial-based mutation and SBX. The stopping criterion consisted of reaching a maximum number of evaluations.

To assess performance, we adopted the hypervolume, ONVG, IGD^+, and two set coverage. All of them were described in Chapter 2 (page 13). In the case of IGD^+, the approximation set was the result of collecting the non-dominated solutions of all independent executions and algorithms.

Unless otherwise stated, we performed 30 independent runs for each algorithm and problem with different initial populations. We applied the Wilcoxon rank sum test (one-tailed) to the mean of IGD^+ and hypervolume, in order to determine if MOMBI-III outperformed the other algorithms.

7.3 Validation

In this section, we investigate our proposed method via solving three widely known test problems: TNK1, OSY2 and VIE4. Their definition can be found in Section B.6 (page 164). The reference points for the hypervolume indicator were set to $(1.1, 1.1)$ for TNK1, $(-20, 80)$ for OSY2, and $(10, -10, 30)$ for VIE4.

Table 7.2: Median and standard deviation of the performance indicators for the constrained-test problems. The best results are presented in **boldface.**

Optimizer	**Indicator**							
	ONVG		Hypervolume			IGD^+		
TNK1								
NSGA-II	74	2.259794	0.428369	0.000261	↑	0.002970	0.000100	↑
NSGA-III	69	2.512414	0.427553	0.000643	↑	0.003000	0.000156	↑
C-MOEA/D	62	2.167692	0.427636	0.000408	↑	0.003489	0.000225	↑
MOMBI-III	**95**	2.897509	**0.429029**	0.000429	-	**0.002313**	0.000123	-
OSY2								
NSGA-II	96	17.112049	14974.32	3364.193461	↑	0.959872	46.168012	=
NSGA-III	95	4.138706	14933.77	300.499811	↑	1.076940	1.396692	↑
C-MOEA/D	95	34.005490	14897.06	3444.743443	↑	1.440762	26.022540	↑
MOMBI-III	**100**	0.300000	**14998.72**	1147.806237	-	**0.875703**	9.768181	-
VIE4								
NSGA-II	102	1.682921	279.5652	0.332322	↑	0.079862	0.009458	↑
NSGA-III	103	1.580787	278.5841	0.491143	↑	0.059660	0.007093	↑
C-MOEA/D	70	2.222361	268.1726	1.678036	↑	0.055481	0.001812	↑
MOMBI-III	**106**	0.179505	**280.5825**	0.125802	-	**0.035694**	0.000986	-

Table 7.3: Two-set coverage of the optimizers for the constrained-test problems.

Optimizer	NSGA-II	NSGA-III	MOEA/D	MOMBI-III
TNK1				
NSGA-II	-	0.179787	0.231436	0.121631
NSGA-III	0.228514	-	0.231436	**0.126001**
C-MOEA/D	0.173913	0.171277	-	0.104880
MOMBI-III	**0.303337**	**0.265957**	**0.321782**	-
OSY2				
NSGA-II	-	0.191675	0.130526	0.065132
NSGA-III	0.237146	-	0.103158	**0.072368**
C-MOEA/D	0.143757	0.138432	-	0.052632
MOMBI-III	**0.473242**	**0.425944**	**0.174737**	-
VIE4				
NSGA-II	-	**0.029039**	**0.000957**	**0.022794**
NSGA-III	0.016206	-	0.000478	0.013603
C-MOEA/D	0.003337	0.003723	-	0.001838
MOMBI-III	**0.034318**	0.026433	0.000000	-

Tables 7.2 and 7.3 present our experimental results. Regarding the ONVG indicator, MOMBI-III obtained for the three problems the highest number of non-dominated

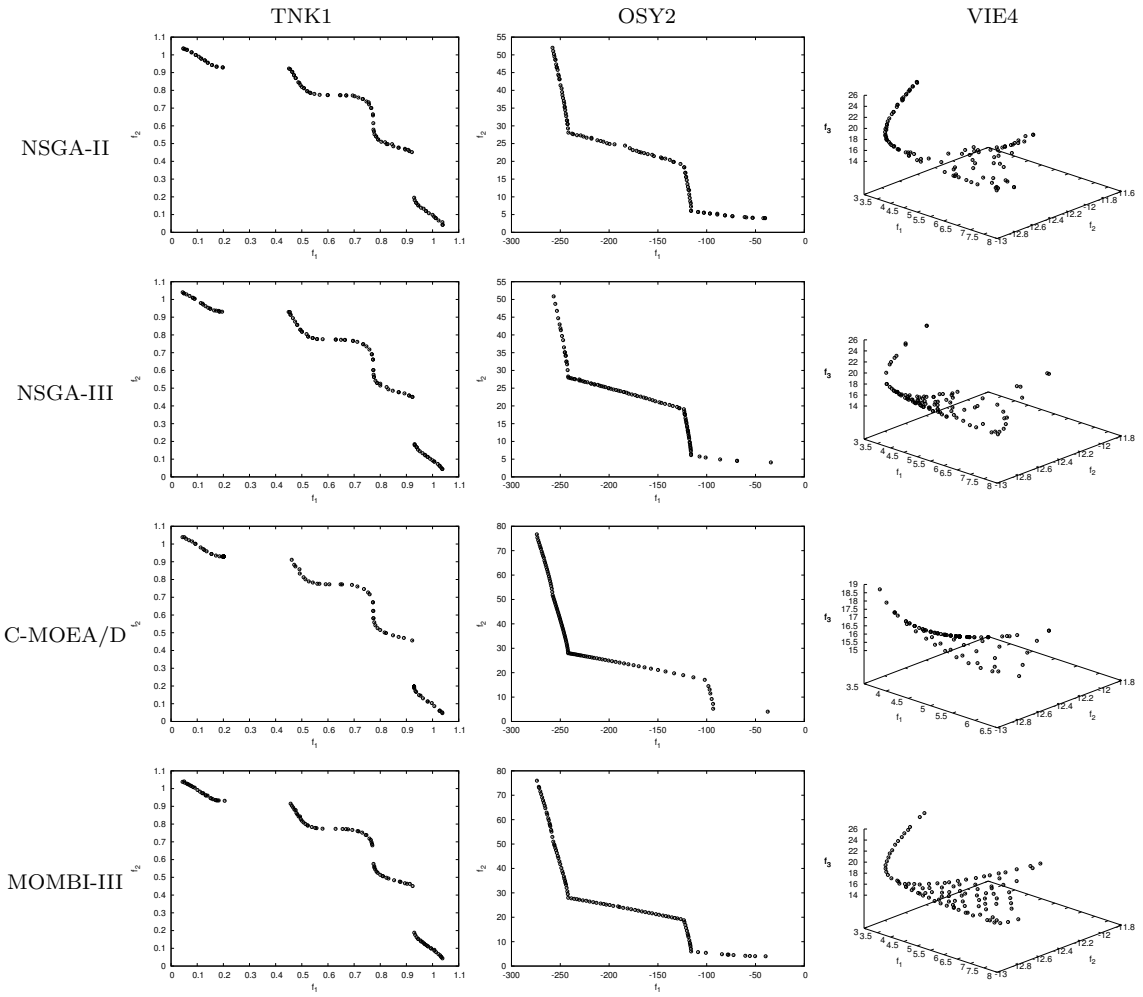


Figure 7.1: Pareto fronts produced by the optimizers corresponding to the median hypervolume for the constrained-test problems.

solutions. NSGA-II ranked second, followed by NSGA-III. Concerning the hypervolume indicator, MOMBI-III significantly outperformed the other algorithms (denoted by $\uparrow$) in all cases at the confidence interval of 99%. In the second place, we had NSGA-II followed by NSGA-III. With respect to the IGD^+, MOMBI-III also significantly outperformed NSGA-II, NSGA-III and C-MOEA/D at the confidence interval of 99%. Only for OSY2, there was a tie with NSGA-II. This last algorithm outperformed NSGA-III and C-MOEA/D in the TNK1 and OSY2 problems. For VIE4, C-MOEA/D was the second best algorithm, whereas NSGA-III was in the third position.

The two set coverage revealed a similar behavior. MOMBI-III performed better than the other optimizers. Only for VIE4, NSGA-II obtained slightly better results than our proposal. Finally, Figure 7.1 illustrates some Pareto fronts. As can be noticed, MOMBI-III achieved a much better distribution than the other algorithms.

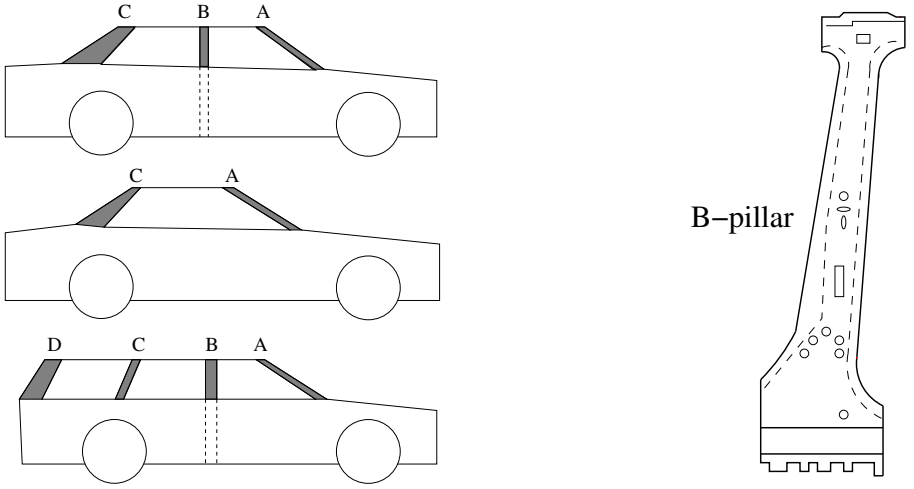


Figure 7.2: Typical pillar configurations of automobiles (left) and isolation of the B-pillar (right). The A-pillar supports the windshield. The B-pillar provides strength to the midsection of the vehicle (many sports cars do not have it). The C-pillar is often the most heavily leveraged pillar from a styling perspective. The D-pillar is the end of the line, designed for housing the rear door in a wagon or suburban.

7.4 Car Side-Impact Problem

In this problem, a car is subject to a side-impact based on European Enhanced Vehicle-Safety Committee (EEVC) procedures. The effect of the side-impact on a dummy passenger in terms of load in abdomen, pubic symphysis force, viscous criterion, and rib deflections are considered. Moreover, the effect on the car is considered in terms of the average velocity of the B-Pillar (see Figure 7.2), which is responsible for withstanding the impact load. An increase in dimensions of the B-pillar may improve the performance on the dummy passenger but with a burden of increased weight of the car, which may have an adverse effect on the fuel economy. Thus, there is a need to find a design balancing the weight and the safety performance.

The problem, redefined by Jain and Deb [52] from Gu et al. [48], is aimed at minimizing the weight of the car (f_1), the pubic force experienced by a dummy passenger (f_2) and the average velocity of the B-pillar (f_3). There are seven decision variables describing the thicknesses (in millimeters) of B-pillar inner (x_1), B-pillar reinforcement (x_2), floor side inner (x_3), cross members (x_4), door beam (x_5), door beltline reinforcement (x_6), and roof rail (x_7). There are ten constraints involving the regulations and requirements for vehicle side impact:

$g_1(x)$ Abdomen load ≤ 1 kN

$g_2(x)$ Upper viscous criteria ≤ 0.32 m/s

$g_3(x)$ Middle viscous criteria ≤ 0.32 m/s

$g_4(x)$ Lower viscous criteria ≤ 0.32 m/s

$g_5(x)$ Upper rib deflection ≤ 32 mm

$g_6(x)$ Middle rib deflection ≤ 32 mm

$g_7(x)$ Lower rib deflection ≤ 32 mm

$g_8(x)$ Pubic symphysis force (F) ≤ 4 kN

$g_9(x)$ Velocity of B-pillar at middle point (V_{MBP}) ≤ 9.9 mm/ms

$g_{10}(x)$ Velocity of front door at B-pillar (V_{FD}) ≤ 15.7 mm/ms

The mathematical formulation of the problem is given below.

$$
\begin{aligned}
F =& 4.72 - 0.5x_4 - 0.19x_2x_3 \\
V_{MBP} =& 10.58 - 0.674x_1x_2 - 0.67275x_2 \\
V_{FD} =& 16.45 - 0.489x_3x_7 - 0.843x_5x_6 \\
f_1(\mathbf{x}) =& 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 0.00001x_6 + 2.73x_7 \\
f_2(\mathbf{x}) =& F \\
f_3(\mathbf{x}) =& 0.5(V_{MBP} + V_{FD}) \\
g_1(\mathbf{x}) =& 0.3717x_2x_4 + 0.0092928x_3 - 0.16 \geq 0 \\
g_2(\mathbf{x}) =& 0.059 + 0.0159x_1x_2 + 0.06486x_1 + 0.019x_2x_7 - 0.0144x_3x_5 - 0.0154464x_6 \geq 0 \\
g_3(\mathbf{x}) =& 0.106 - 0.00817x_5 + 0.045195x_1 + 0.0135168x_1 - 0.03099x_2x_6 + 0.018x_2x_7 \\
& - 0.007176x_3 - 0.023232x_3 + 0.00364x_5x_6 + 0.018x_2^2 \geq 0 \\
g_4(\mathbf{x}) =& 0.61x_2 + 0.031296x_3 + 0.031872x_7 - 0.227x_2^2 - 0.42 \geq 0 \\
g_5(\mathbf{x}) =& 3.02 - 3.818x_3 + 4.2x_1x_2 - 1.27296x_6 + 2.68065x_7 \geq 0 \\
g_6(\mathbf{x}) =& 5.057x_1x_2 - 2.95x_3 + 3.795x_2 + 3.4431x_7 - 3.31728 \geq 0 \\
g_7(\mathbf{x}) =& 9.9x_2 + 4.4505x_1 - 14.36 \geq 0 \\
g_8(\mathbf{x}) =& 4 - F \geq 0 \\
g_9(\mathbf{x}) =& 9.9 - V_{MBP} \geq 0 \\
g_{10}(\mathbf{x}) =& 15.7 - V_{FD} \geq 0 \\
& x_1, x_3, x_4 \in [0.5, 1.5],\ x_2 \in [0.45, 1.35], \\
& x_5 \in [0.875, 2.625],\ \text{and}\ x_6, x_7 \in [0.4, 1.2].
\end{aligned}
$$

Table 7.4: Median and standard deviation of the performance indicators for the car side-impact problem. The best results are presented in **boldface.**

Optimizer	Indicator							
	ONVG		Hypervolume			IGD^+		
NSGA-II	155	6.674995e-01	10.801690	5.423048e-02	↑	8.013675e-02	9.073953e-03	↑
NSGA-III	153	1.797838e+00	10.893650	9.993456e-02	↑	1.060380e-01	7.496957e-02	↑
C-MOEA/D	146	4.016079e+00	4.202341	8.193931e-02	↑	2.379462e-01	2.983392e-03	↑
MOMBI-III	**156**	2.494438e-01	**11.5209**	1.695381e-02	-	**4.602613e-02**	4.210079e-03	-

Table 7.5: Two-set coverage of the optimizers for the car side-impact problem.

Optimizer	NSGA-II	NSGA-III	MOEA/D	MOMBI-III
NSGA-II	-	0.167378	0.000000	0.014601
NSGA-III	0.210963	-	0.000000	0.011025
C-MOEA/D	0.073699	0.039710	-	**0.029499**
MOMBI-III	**0.485951**	**0.427412**	**0.000296**	-

Tables 7.4 and 7.5 present our experimental results. The reference point for the hypervolume indicator was set to $(42.8, 4.1, 12.6)$. Regarding the ONVG indicator, MOMBI-III obtained the maximum number of non-dominated solutions. NSGA-II scored in second place followed by NSGA-III. Concerning the hypervolume indicator, MOMBI-III significantly outperformed the other optimizers (denoted by $\uparrow$) at the confidence interval of 99%. NSGA-III ranked second followed by NSGA-II. With respect to the IGD^+, MOMBI-III also significantly outperformed NSGA-II, NSGA-III and C-MOEA/D at the confidence interval of 99%. NSGA-II ranked second followed by NSGA-III. The two set coverage revealed a similar behavior since MOMBI-III performed better than the other optimizers. Figures 7.3 and 7.4 show some examples of Pareto fronts. It is worth noticing that MOMBI-III achieved much better distribution than the other algorithms.

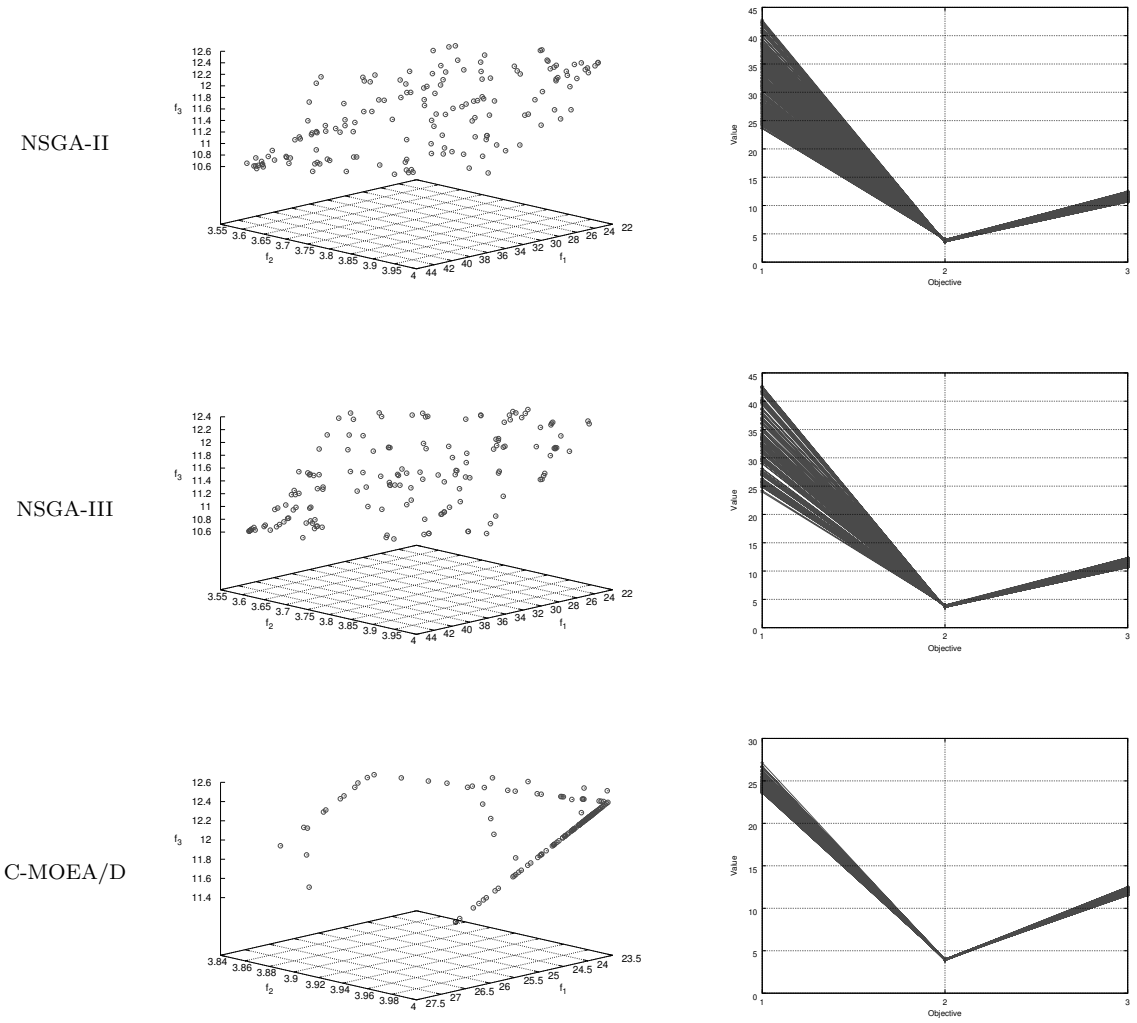


Figure 7.3: Pareto fronts produced by the optimizers corresponding to the median hypervolume for the car side-impact problem.

MOMBI-III

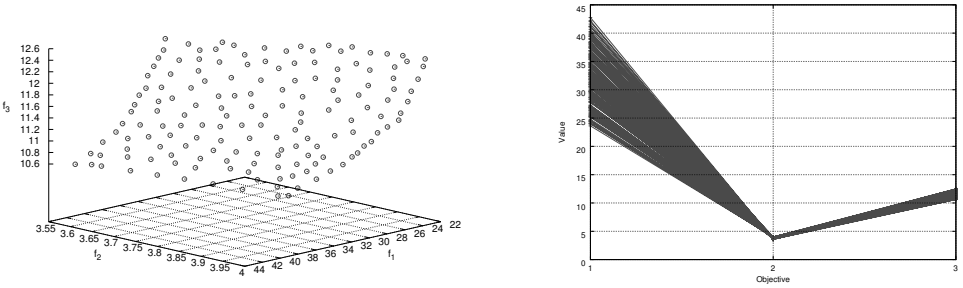


Figure 7.4: Pareto fronts produced by the optimizers corresponding to the median hypervolume for the car side-impact problem (cont'd).

7.5 Water Resources Planning

This application involves optimal planning for storm-drainage systems in urban areas, described originally by Musselman and Talavage [103], and also attempted by Jain and Deb [52], Cheng and Li [18], and Ray et al. [109].

The problem is to examine, in terms of its storm drainage needs, a particular subbasin within a watershed. The subbasin is assumed to be hydrologically independent of other subbasins, having its own drainage network, on-site detention storage facility, treatment plant, and tributary to a receiving water body. The various factors involved in this problem as well as how they conceptually interlink are illustrated in Figure 7.5. Runoff due to rainfall and snowfall is channeled to a treatment facility before being discharged into the receiving body of water. Any water which is not able to be immediately treated is rerouted to a temporary storage facility to await treatment. Once the storage facility has filled, any further runoff either overflows into the receiving body of water or it backs up.

The subbasin's storm drainage system is assumed to be characterized by three decision variables: x_1 local detention storage capacity (basin inches), x_2 maximum treatment rate (basin $\cdot$ inches/hour), and x_3 maximum allowable overflow rate (basin inches/hour). Restricting the overflow rate makes it possible to incorporate into the analysis any damages caused by local flooding. Once the detention storage facility is filled and the overflow rate is at its maximum, the drainage system's conveying capacity is significantly reduced. This results in local flooding, the consequence of which is structural damage and economic disruption to the area.

The problem is modeled with five objective functions to be minimized: f_1 drainage network cost, f_2 storage facility cost, f_3 treatment facility cost, f_4 expected flood damage cost, and f_5 expected economic loss due to flood; and seven constraints: g_1 average number of floods per year, g_2 probability of exceeding a flood depth of 0.01 basin-inches, g_3 average number of pounds per year of suspended solids, g_4 average number of pounds per year of settleable solids, g_5 average number of pounds per year of biochemical oxygen demand, g_6 average number of pounds per year of nitro-

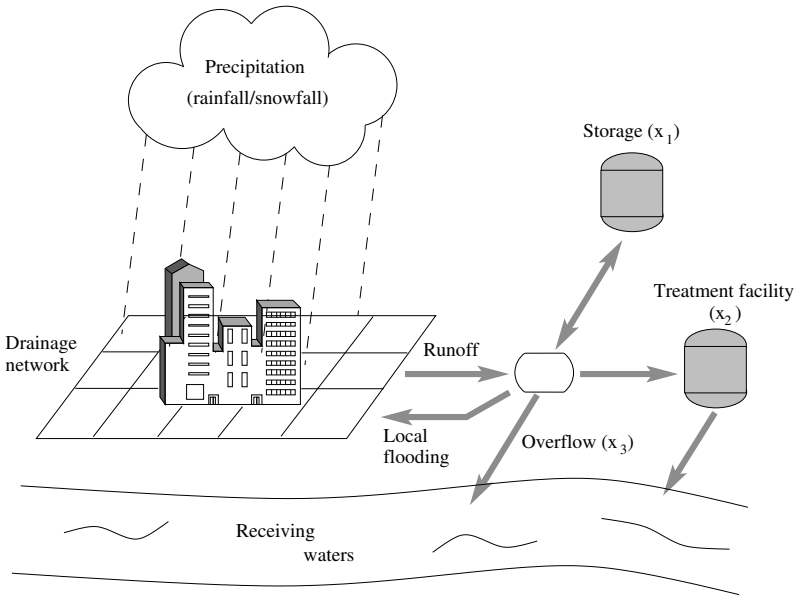


Figure 7.5: Subbasin's storm drainage system (reproduced from [103]).

gen, g_7 average number of pounds per year of orthophosphate. The mathematical formulation of the problem is given as follows:

$$
\begin{aligned}
f_1(\mathbf{x}) =& 106780.37(x_2 + x_3) + 61704.67 \\
f_2(\mathbf{x}) =& 3000x_1 \\
f_3(\mathbf{x}) =& 305700 \times 2289x_2/(0.06 \times 2289)^{0.65} \\
f_4(\mathbf{x}) =& 250 \times 2289 \exp(-39.75x_2 + 9.9x_3 + 2.74) \\
f_5(\mathbf{x}) =& 25((1.39/(x_1x_2)) + 4940x_3 - 80) \\
g_1(\mathbf{x}) =& 1 - 0.00139/(x_1x_2) - 4.94x_3 + 0.08 \geq 0 \\
g_2(\mathbf{x}) =& 1 - 0.000306/(x_1x_2) - 1.082x_3 + 0.0986 \geq 0 \\
g_3(\mathbf{x}) =& 50000 - 12.307/(x_1x_2) - 49408.24x_3 - 4051.02 \geq 0 \\
g_4(\mathbf{x}) =& 16000 - 2.098/(x_1x_2) - 8046.33x_3 + 696.71 \geq 0 \\
g_5(\mathbf{x}) =& 10000 - 2.138/(x_1x_2) - 7883.39x_3 + 705.04 \geq 0 \\
g_6(\mathbf{x}) =& 2000 - 0.417/(x_1x_2) - 1721.26x_3 + 136.54 \geq 0 \\
g_7(\mathbf{x}) =& 550 - 0.164/(x_1x_2) - 631.13x_3 + 54.48 \geq 0 \\
& x_1 \in [0.01, 0.45] \\
& x_2, x_3 \in [0.01, 0.10].
\end{aligned}
$$

Tables 7.6 and 7.7 present the experimental results of our MOMBI-III and the algorithms NSGA-II, NSGA-III and C-MOEA/D. The reference point for the hypervolume indicator was set to $(76490, 1360, 2853470, 8781920, 25010)$.

Regarding the ONVG indicator, MOMBI-III obtained the maximum number of non-dominated solutions. NSGA-II ranked second followed by NSGA-III. Concerning

the hypervolume indicator, MOMBI-III significantly outperformed the other optimizers (denoted by ↑) at the confidence interval of 99%. In the second place, there was NSGA-II followed by NSGA-III. With respect to the IGD^+, MOMBI-III also significantly outperformed NSGA-II, NSGA-III and C-MOEA/D at the confidence interval of 99%. NSGA-II ranked second followed by NSGA-III. The two set coverage revealed a similar behavior since MOMBI-III performed better than NSGA-II and NSGA-III. Finally, Figure 7.6 depicts the parallel coordinates of some optimal solutions. It is worth noticing that NSGA-II and MOMBI-III covered much more area than the other algorithms.

Table 7.6: Median and standard deviation of the performance indicators for the water problem. The best results are presented in **boldface.**

Optimizer	**Indicator**								
	ONVG		Hypervolume			IGD^+			
NSGA-II	211	0.87	4.757204e+24	3.27e+22	↑	1.044788e+04	1.17e+03	↑
NSGA-III	208	2.27	4.496746e+24	2.08e+23	↑	7.272726e+04	7.38e+04	↑
C-MOEA/D	187	2.57	2.104973e+24	2.23e+21	↑	9.089462e+04	1.56e+02	↑
MOMBI-III	**212**	0.37	**4.897308e+24**	1.88e+22	-	**8.716032e+03**	7.81e+02	-

Table 7.7: Two-set coverage of the optimizers for the water problem.

Optimizer	NSGA-II	NSGA-III	C-MOEA/D	MOMBI-III
NSGA-II	-	0.067285	0.000000	0.000475
NSGA-III	0.061300	-	0.000000	**0.000634**
C-MOEA/D	0.001963	0.001078	-	0.000000
MOMBI-III	**0.084206**	**0.077421**	0.000000	-

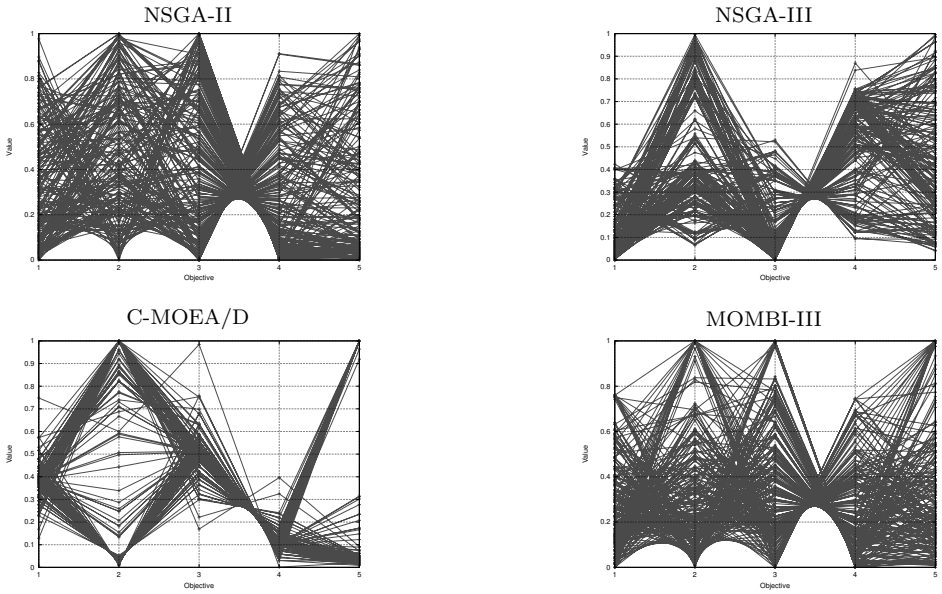


Figure 7.6: Normalized parallel coordinates of the Pareto front produced by optimizers corresponding to the median hypervolume.

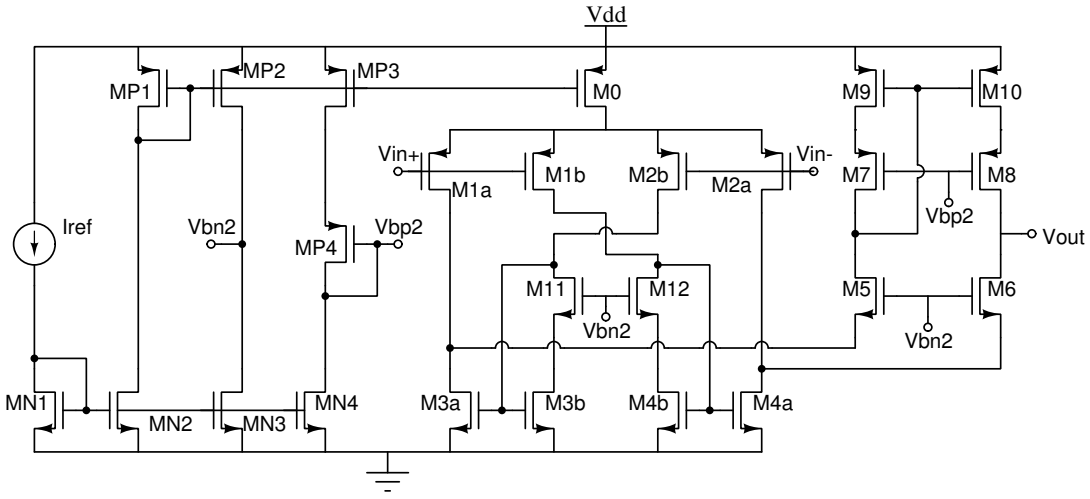


Figure 7.7: Recycled Folded Cascode OTA.

7.6 Design of an Analog Integrated Circuit

In this section, we optimize the design of an analog integrated circuit, more specifically, the Recycled Folded Cascode (RFC) Operational Transconductance Amplifier (OTA). An OTA is a DC-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended current output. The gain is measured as current divided by voltage.

The RFC OTA, initially proposed by Assad and Silva-Martinez [6] and then expanded by Guerra-Gómez et al. [49], delivers a substantially enhanced performance over the conventional folded OTA. The circuit is composed of 25 Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) using the configuration of Figure 7.7. The goal is to find the dimensions of all MOSFETs that optimize at the same time the eight objectives of Table 7.8. Ten decision variables represent the width (W) or length (L) of the MOSFETs as shown in Table 7.9. Each variable is an integer multiple of the minimum value allowed by the fabrication process (0.18 μm). Also, the problem involves 25 constraints for the saturation conditions in all MOSFETs and seven target specifications, which are provided in the last column of Table 7.8.

The evaluation of the MOP is performed via the simulator *ngspice* version 26[1]. The RFC OTA is biased with $Iref = 400\mu$A and $Vdd = 1.8$V. The electrical measurements consider a load capacitor of 5.6 pF, and the ngspice simulations are fixed to LEVEL 49 standard CMOS Technology of 0.18 μm.

In 2013, Guerra-Gómez et al. [49] tackled this problem by employing NSGA-II coupled to a module for handling constraints. The idea was to give priority to those solutions having low sensitivity to natural variations of the fabrication process. For this purpose, authors relied on finite differences and Richardson extrapolation to

[1] `http://ngspice.sourceforge.net`

calculate the partial derivatives of the objective functions concerning the decision variables. The adopted variation operator was Differential Evolution [120]. The main drawback of this method is the incorporation of new user-defined parameters.

Table 7.8: Objective functions in the RFC OTA.

Requirement	**Objective Function**	**Description**	**Unit**	**Constraint**
maximize	f_1	DC gain	dB	≥ 65.35
	f_2	gain bandwidth product (GBW)	MHz	≥ 89.13
	f_3	phase margin (PM)	degrees	≥ 45
	f_4	slew rate (SR)	V/μsec	≥ 76.99
minimize	f_5	power consumption (PW)	mW	$\leq 3.31 \times 10^{-3}$
	f_6	settling time (ST)	nsec	$\leq 20.14 \times 10^{-9}$
	f_7	input offset	μV	$\leq 206.79 \times 10^{-6}$
	f_8	input referred noise	mVrms	–

Table 7.9: Amplifier device dimensions.

Decision Variable	**Size**	**Transistor**	**Range**	**Unit**
x_1	L_1	M0, M3a, M3b, M4a, M4b, M9, M10, MN1,...,MN4,MP1,...,MP4	$[1, 4] \times 0.18$	μm
x_2	L_2 $2L_2$	M5,...,M8 M1a, M1b, M2a, M2b	$[1, 4] \times 0.18$	μ m
x_3	W_1	M0, MP1	$[1, 778] \times 0.18$	μ m
x_4	W_2	M1a, M1b, M2a, M2b	$[1, 778] \times 0.18$	μ m
x_5	W_3	M3a, M4a	$[1, 778] \times 0.18$	μ m
x_6	W_4	M3b, M4b	$[1, 778] \times 0.18$	μ m
x_7	W_5 $2W_5$ $4W_5$	M5, M6, MN3, MN4 MN1, MN2, MP4 MP2, MP3	$[1, 778] \times 0.18$	μ m
x_8	W_6	M7, M8	$[1, 778] \times 0.18$	μ m
x_9	W_7	M9, M10	$[1, 778] \times 0.18$	μ m
x_{10}	W_8	M11, M12	$[1, 778] \times 0.18$	μ m

Table 7.10 reproduces the best performances of Guerra-Gómez et al. [49] for each objective using a budget of 50,000 evaluations of the MOP. For instance, p_2, having the decision vector $(2, 1, 474, 400, 83, 39, 59, 100, 43, 59) \times 0.18$, is the best solution for the objectives GBW and ST. On the other hand, Table 7.11 presents the best performances of MOMBI-III. As can be appreciated for the objectives to be maximized, DC gain (67.98 vs. 81.41), GBW (120.9 vs. 176.62), and SR (201.28 vs. 244.9) were considerably improved by our proposal. Concerning the objectives to be minimized, PW (3.25 vs. 3.0), ST (15.64 vs. 3.5), and Noise (2.97 vs. 2.3) were also enhanced by MOMBI-III. Even though objectives PM and Offset could not be improved by MOMBI-III, they are not so far from those of the literature and still satisfy the constraints.

Table 7.10: Best points (in **boldface**) reported by Guerra-Gómez et al. [49].

Feature	**Solution**						
	p_1	p_2	p_3	p_4	p_5	p_6	p_7
L_1	0.36	0.36	0.36	0.36	0.36	0.36	0.36
L_2	0.18	0.18	0.18	0.18	0.18	0.18	0.18
W_1	132.48	85.32	85.86	78.3	125.46	120.06	152.82
W_2	58.5	72	41.94	49.68	45.18	52.2	45.72
W_3	18.9	14.94	17.64	9.9	38.52	42.3	30.96
W_4	9.9	7.02	8.1	4.86	19.62	22.14	16.92
W_5	16.2	10.62	10.8	10.08	15.66	14.76	18.72
W_6	32.76	18	17.1	69.66	61.92	25.92	59.58
W_7	5.22	7.74	8.28	5.22	8.82	6.84	4.5
W_8	24.3	10.62	18	31.14	8.28	2.88	12.96
DC gain	**67.98**	67.52	66.07	67.64	67.81	67.41	67.4
GBW	105.53	**120.9**	99.94	99.9	97.44	107.36	93.66
PM	77.62	76.4	**80.74**	78.64	77.53	75.86	77.26
SR	84.47	92.28	82.37	79.48	84.88	**201.28**	80.67
PW	3.26	3.29	3.3	**3.25**	3.29	3.29	3.26
ST	17.47	**15.64**	18.51	17.95	18.51	16.66	18.88
Offset	19.24	58.52	75.68	0.07	96.61	51.71	**-0.98**
Noise	3.15	3.08	3.06	**2.97**	3.52	3.43	3.25

Table 7.11: Best points (in **boldface**) obtained by MOMBI-III.

Features	**Solution**							
	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
L_1	0.54	0.36	0.36	0.36	0.72	0.36	0.36	0.72
L_2	0.36	0.18	0.18	0.18	0.18	0.18	0.18	0.18
W_1	124.38	125.46	126.36	124.38	113.4	125.46	125.1	127.44
W_2	139.32	134.1	55.98	138.6	125.64	126.18	136.62	132.48
W_3	41.76	50.94	26.64	55.44	42.3	50.94	41.04	30.78
W_4	33.3	31.32	17.1	31.32	32.4	31.32	29.34	20.88
W_5	15.3	15.3	15.48	15.30	14.04	15.3	15.3	15.48
W_6	71.82	43.92	7.56	42.84	14.4	43.56	48.6	133.2
W_7	5.76	7.92	4.32	5.04	4.14	27	3.24	10.44
W_8	2.52	1.44	6.48	2.52	6.66	1.44	1.62	25.02
DC gain	**81.41**	73.14	67.76	69.73	73.82	71.47	71.58	65.74
GBW	99.38	**176.62**	94.03	133.90	99.5	175.43	134.5	96.5
PM	50.96	47.24	**78.33**	63.74	61.57	46.62	59.04	56.64
SR	126.61	217.86	87.05	**244.9**	164.69	213.97	230.9	80.55
PW	3.05	3.3	3.15	3.26	**3**	3.3	3.17	3.04
ST	10.61	4.04	19.95	13.93	10.95	**3.5**	12.83	10.06
Offset	20.85	40.97	58.97	10.98	15.12	120.76	**0.01**	106.18
Noise	6.62	5.01	3.05	3.58	3.72	4.59	3.78	**2.3**

Table 7.12 shows the statistics, the median, average, minimum, maximum, and standard deviation for all objectives among the final solution set. In all cases, MOMBI-III achieved the best results. Table 7.13 provides some interesting solutions that do not belong to extreme solutions of the Pareto front.

Table 7.12: Best points (in **boldface**) for the RFC OTA obtained by the state-of-the-art Guerra-Gómez et al. [49] and MOMBI-III.

Objective	**Optimizer**	**Statistic**				
		med	**avg**	**min**	**max**	**std**
DC gain	baseline	-	67.01	66.46	67.83	0.42
	MOMBI-III	71.35	71.56	65.35	**81.41**	3.21
GBW	baseline	-	96.72	94.63	106.52	2.97
	MOMBI-III	121.04	123.41	89.80	**176.62**	20.21
PM	baseline	-	75.96	75.48	77.30	0.45
	MOMBI-III	59.59	59.72	45.00	**78.33**	7.99
SR	baseline	-	77.63	77.09	79.64	0.56
	MOMBI-III	211.22	196.16	78.60	**244.90**	37.74
PW	baseline	-	3.28	3.24	3.30	0.02
	MOMBI-III	3.13	3.14	**3.00**	3.31	0.08
ST	baseline	-	18.25	16.90	18.49	0.36
	MOMBI-III	10.61	11.37	**3.50**	20.05	4.12
Offset	baseline	-	34.84	0.03	96.97	31.96
	MOMBI-III	35.96	49.29	**0.01**	204.16	45.12
Noise	baseline	-	63.51	55.27	66.40	3.27
	MOMBI-III	3.73	3.91	**2.30**	6.88	0.86

Table 7.13: Compromise solutions obtained by MOMBI-III.

Features	**Solution**							
	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
L_1	0.36	0.36	0.36	0.36	0.54	0.54	0.36	0.36
L_2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
W_1	125.46	125.28	125.46	125.1	124.38	124.38	129.24	126.18
W_2	86.04	138.78	136.98	135.36	107.64	108.36	138.24	135.36
W_3	46.8	40.86	46.8	45.72	46.44	44.46	26.82	41.58
W_4	31.32	31.14	31.32	30.96	32.4	32.58	22.86	26.1
W_5	15.3	15.3	15.3	15.3	15.3	15.3	15.84	15.48
W_6	48.6	39.96	48.6	75.24	12.24	34.92	70.92	63.18
W_7	4.5	3.24	4.5	4.14	5.04	4.5	2.16	3.96
W_8	1.44	1.44	1.44	1.44	2.52	2.52	1.44	1.98
DC gain	72.17	73.11	73.08	72.52	73.22	73.87	73.12	70.5
GBW	153.62	150.45	167.66	153.56	130.09	124.44	127.86	130.57
PM	54.51	50.66	46.05	51.13	55	56.44	51.85	62.33
SR	227.45	227.14	226.12	231.8	206.74	206.23	223.88	236.69
PW	3.24	3.17	3.25	3.22	3.12	3.1	3.1	3.22
ST	5.77	6.06	4.77	6.09	8.03	7.52	8.04	13.94
Offset	1.36	0.98	0.12	0.9	0.09	2.64	0.73	0.31
Noise	4.6	4.37	4.77	4.36	4.27	4.32	3.91	3.58

It is worth noticing that in our experiments, we performed only three independent runs for each algorithm since this application is time-consuming. The average time it takes a single optimization run is 6.5 hours using a PC Intel(R) Core(TM) i7 CPU 950 @ 3.07 GHz $\times$ 8 with 3.8 GB of memory. Furthermore, to obtain feasible solutions at the end of generations in C-MOEA/D and NSGA-II, it was necessary to inject a

feasible solution during the initialization of the population. Such solution is extracted from the literature and appears in column p_6 of Table 7.10.

Tables 7.14, 7.15 and 7.16 present the experimental results of MOMBI-III and the compared algorithms. The reference point for the hypervolume indicator was set to $(-64, -6, -44, -75, 4, 21, 207, 8)$.

Regarding the ONVG indicator, MOMBI-III obtained the maximum number of non-dominated solutions. NSGA-II ranked second followed by NSGA-III. Concerning the hypervolume indicator, MOMBI-III significantly outperformed NSGA-II and NSGA-III (denoted by $\uparrow$) at the confidence interval of 95%. With MOEA/D there was a tie (denoted by $=$). MOEA/D ranked second followed by NSGA-II. For the IGD^+, MOMBI-III also significantly outperformed the three other optimizers at the confidence interval of 95%. MOEA/D ranked second followed by NSGA-II. The two set coverage confirms that MOMBI-III performed better than the other algorithms. Finally, Figure 7.8 shows the parallel coordinates of the final sets. As can be observed, NSGA-II and MOMBI-III covered much more area than the other two algorithms.

Table 7.14: Detailed information of the performance indicators for the RFC OTA. The best results are presented in **boldface.**

Execution Number	**Optimizer**			
	NSGA-II	NSGA-III	MOEA/D	MOMBI-III
ONVG				
1	249	234	212	**250**
2	**250**	231	210	**250**
3	248	234	211	**249**
all	730	688	621	**733**
Hypervolume				
1	9.053320e+08	6.132072e+07	9.670065e+08	**1.138170e+09**
2	9.065817e+08	4.584645e+08	8.749099e+08	**9.135469e+08**
3	8.390591e+08	4.209148e+07	8.808672e+08	**1.116637e+09**
all	1.036827e+09	4.687917e+08	1.059399e+09	**1.253497e+09**
IGD^+				
1	2.477912e+00	7.896030e+01	2.130124e+00	**7.269151e-01**
2	2.604847e+00	1.051483e+01	2.223723e+00	**1.633110e+00**
3	4.008368e+00	7.935186e+01	2.849773e+00	**8.939437e-01**
all	1.795547e+00	1.024330e+01	1.595704e+00	**3.918300e-01**

Table 7.15: Median and standard deviation of the performance indicators for the RFC OTA.

Optimizer	Indicator								
	ONVG		Hypervolume			IGD^+		
NSGA-II	249	8.16e-01	9.053320e+08	3.15e+07	$\uparrow$	2.604847e+00	6.93e-01	$\uparrow$
NSGA-III	234	1.41e+0	6.132072e+07	1.92e+08	$\uparrow$	7.896030e+01	3.24e+01	$\uparrow$
MOEA/D	211	8.16e-01	8.808672e+08	4.21e+07	$=$	2.223723e+00	3.19e-01	$\uparrow$
MOMBI-III	**250**	4.71e-01	**1.116637e+09**	1.01e+08	-	**8.939437e-01**	3.94e-01	-

Table 7.16: Two-set coverage of the optimizers for the RFC OTA.

Optimizer	NSGA-II	NSGA-III	C-MOEA/D	MOMBI-III
NSGA-II	-	0.116	0.00	0.001333
NSGA-III	0.001333	-	0.00	0.00
C-MOEA/D	0.086667	0.318667	-	**0.013333**
MOMBI-III	**0.154667**	**0.324**	**0.008**	-

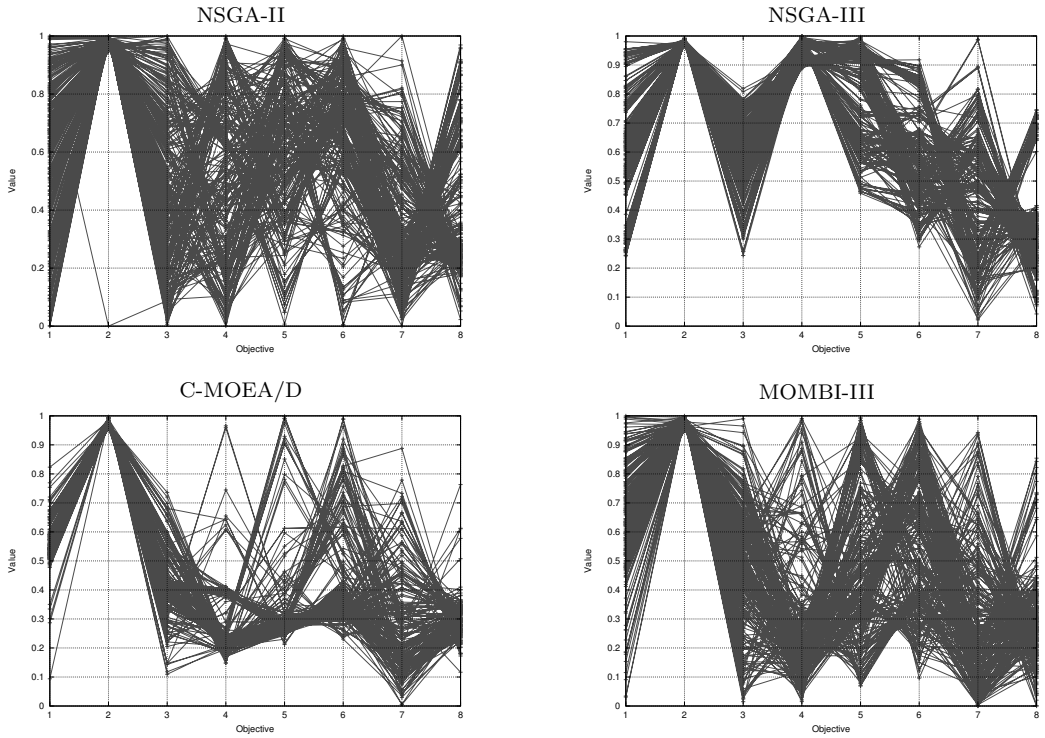


Figure 7.8: Normalized parallel coordinates of the Pareto front produced by optimizers corresponding to the median hypervolume.

7.7 Summary

In order to show the successful application on real-world problems of one of our proposals, we incorporated to it, a mechanism to handle constraints. This mechanism does not require extra parameters and does not increase the computational complexity of the multi-objective evolutionary algorithm. The chapter also described the adopted constraint-handling techniques of the state-of-the-art optimizers used to compare our results. MOMBI-III was compared to C-MOEA/D, NSGA-II and NSGA-III on three well-known test problems having two and three objectives, as well as in three real-world applications having up to eight objectives. In all cases, our proposed approach obtained the best results, proving to be a powerful tool.

Chapter 8

Conclusions and Future Work

Most real-world problems are multi-objective in nature, and their solution involves finding a set of decision variables that represent the best possible trade-offs among all their objectives. Evolutionary algorithms, as well as other bio-inspired metaheuristics, are powerful search techniques that are suitable to solve such types of problems.

One main concern is that evolutionary algorithms based on Pareto dominance and a niching technique may suffer stagnation or even involution during the search. Thus, several approaches have replaced the niching technique by the hypervolume indicator, which is compliant with Pareto dominance. However, the computational cost of the hypervolume makes it unaffordable for many-objective optimization problems.

In order to address these issues, we have explored alternative techniques for the selection of individuals. One possibility is the development of generic methods, which can produce solutions of acceptable quality using a set of easy-to-implement low-level heuristics. These methods are known as hyper-heuristics and can be seen as high-level methodologies, which automatically produce an adequate combination of single heuristics for solving a broad set of problems. Another possibility is the use of parallel models, like the island, providing additional benefits since not only execution time is reduced but the quality of final solutions is also improved. Our proposals are based on these strategies. A synopsis of their features is shown in Table 8.1.

In the remainder of this chapter, we conclude with the most remarkable results obtained in this thesis, providing some paths for future research.

MOMBI-II adopts the Augmented Achievement Scalarizing Function since near optimal solutions lie on the weight vectors provided by the user. In many-objective problems, this is an important factor that prevents us from having an unexpected distribution in objective space. In [58] MOMBI-II was compared with other algorithms (Δ_p-DDE, NSGA-III, $R2$-MOGA, $R2$-IBEA and MOMBI) on selected problems of the DTLZ and WFG benchmarks. Experimental results indicated that MOMBI-II was able to outperform all the compared algorithms in more than the 96% of the test instances. Moreover, the solutions produced by MOMBI-II are uniformly distributed in objective space, being similar to those generated by NSGA-III. However, MOMBI-II requires much less computational effort and its source code is in the public

domain.

In [106], results suggest that there is not a unique utility function that can solve all the problems effectively. Instead, there is a subset of them that can tackle effectively specific problems. In general, utility functions like the Augmented Achievement Scalarizing Function, Penalty Boundary Intersection, Exponential Weighted Criteria and Vector Angle Distance Scaling obtained excellent results, deserving more research. Nevertheless, in some cases, Penalty Boundary Intersection and Vector Angle Distance Scaling may find dominated solutions. Thus, we recommend that their use should be verified with Pareto compliant scalarizing functions.

In Chapter 3, we observed that MOMBI-III could effectively outperform state-of-the-art algorithms on the ZDT, DTLZ, and WFG test problems. Furthermore, MOMBI-III could deal with inverted test problems, producing a uniform distributions in objective space. This merit has not been achieved by optimizers like MOEA/D or NSGA-III, which suffer from overspecialization. We also verified that MOMBI-III could effectively handle many-objective problems with constraints, evidencing its applicability in Chapter 7 for solving real-world problems.

In [60], MOVAP was compared with other algorithms (NSGA-III, HyPE, SPEA2 and NSGA-II) on the ZDT and WFG test problems. MOVAP showed a significant gain in more than 35% of the test instances, producing a much better diversity of solutions, and exploring more regions of the search space in high-dimensionality. Conversely, in low dimensionality, MOVAP was competitive.

In [59], S-PAMICRO was compared with HypE, the serial version of SMS-EMOA and the standard island version of SMS-EMOA without the handling of the external archive. In summary, we observed that S-PAMICRO could achieve much better results than SMS-EMOA and HypE in high dimensionality, spending much less computational time. In fact, the execution time seemed to be dominated by polynomial

Table 8.1: Summary of the proposals of this work thesis

Proposal	**Chapter**	**Features**
MOMBI-II	3	Ranking of the population using the $R2$ indicator. Diversity is achieved through the set of weight vectors. One scalarizing function (ASF).
MOMBI-III	3	Hyper-heuristic based on MOMBI-II with seven scalarizing functions. Diversity is also achieved through the s-energy indicator.
MOVAP	4	Non-dominated Sorting coupled with a density estimator based on the Parallel Coordinates.
S-PAMICRO	5	Parallel version of SMS-EMOA using the island model, small populations and an archiving technique based on the density estimator of MOVAP.
EMO Project	6	Framework software designated to solve multi-objective optimization problems. Parallelization of MOEAs.

terms and not the exponential terms when using micro-populations. Moreover, we have studied the effects of the migration parameters on S-PAMICRO, finding that: 1) the absence of migration reduces performance, 2) high rates of migration (less than 10 objective function evaluations) are harmful, 3) during migration, 2 or 3 individuals should be considered, 4) there is a trade-off between the number of islands and the budget allowable for performing function evaluations, and 5) the worst behaved replacement scheme is the elitist random.

These experimental results confirm our hypotheses that: 1) MOEAs combining different search techniques perform much better than MOEAs baed on a single search strategy, and 2) Parallel MOEAs can reduce the execution time of their serial counterparts, with the possibility of also improving the quality of solutions.

As a future work, we will track the following research lines:

Regarding MOMBI-III (see Chapter 3 on page 39), we would like to expand the set of heuristics, incorporating more values for the model parameters of the current scalarizing functions. Such values might be automatically adapted. We would also like to explore a probabilistic approach for the heuristic selection process, since this is currently performed in an exhaustive way.

With respect to MOVAP (see Chapter 4 on page 61), we are interested in studying its scalability beyond seven objectives and studying the properties of the proposed density estimator. Finally, although MOVAP does not require a set of reference points as NSGA-III, or a large number of sampling points as HypE, it is worth indicating that it needs a resolution parameter which, however, could be tuned during execution time.

Concerning $\mathcal{S}$-PAMICRO (see Chapter 5 on page 75), further studies are required, adopting more benchmarks and comparing it to other state-of-the-art MOEAs.

EMO Project (see Chapter 6 on page 89) was designed with the aim to provide users with efficient implementations of MOEAs, having support to parallel platforms. Nevertheless, to improve its usability a wide spectrum of optimizers should be added. We are also interested in expanding the parallelization layer of EMO Project to new technologies, such as cloud computing [138]. This platform would allow us to distribute computations asynchronously over several heterogeneous computers, which may be loosely coupled. Thus, the approximation to the Pareto optimal set could be stored in the cloud.

Appendix A

Hypervolume Contribution

The hypervolume indicator has been subject of a lot of research in the last few years, mainly because its maximization yields near-optimal approximations of the Pareto optimal front of a multi-objective problem. This feature has been exploited by several evolutionary optimizers, in spite of the considerable growth in computational cost that it is involved in the computation of the hypervolume as we increase the number of objectives. Recently, the Walking Fish Group implemented a new version of the incremental hypervolume algorithm, named IWFG 1.01. This implementation is the fastest reported to date for determining the solution that contributes the least to the hypervolume of a non-dominated set. Nevertheless, this new version has gone mostly unnoticed by the research community. We believe that this is due to an error in the source code provided by the authors of this algorithm, which appears when coupling it to a multi-objective evolutionary algorithm. In this appendix, we describe this error, and we propose a solution to fix it. Moreover, we illustrate the significant gains in performance produced by IWFG 1.01 in many-objective optimization problems, when integrated into SMS-EMOA.

The rest of this appendix is structured as follows. Section A.1 gives the motivation and background of the hypervolume contribution. Section A.2 outlines the IWFG 1.01 algorithm. Section A.3 provides the description of the error and our proposed solution. Section A.4 presents the validation of our proposed solution. Section A.5 contains a summary of the appendix.

A.1 Motivation

Optimizers, such as IBEA [154], SMS-EMOA [9], MO-CMA-ES [66] and HypE [7], have incorporated the hypervolume indicator (see definition on page 14) in their survival selection mechanism. The most common way is by using the *exclusive hypervolume* (or *hypervolume contribution*) as fitness. The idea is to measure the size of the part of objective space that a solution $\mathbf{p}$ dominates, but is not dominated by any element of a set A:

$$ExcHV(\mathbf{p}, A) := HV(A \cup \{\mathbf{p}\}) - HV(A). \tag{A.1}$$

Algorithm 12 A naive computation of the exclusive hypervolume

Input: $A \subset Z$ set of solutions
Output: Solution **s** that contributes the least to $HV(A)$
1: $t \leftarrow HV(A)$
2: **for all a** in A **do**
3: $\quad fitness[\mathbf{a}] \leftarrow t - HV(A \setminus \{\mathbf{a}\})$
4: $\mathbf{s} \leftarrow \arg\min_{\mathbf{a} \in A} fitness[\mathbf{a}]$
5: **return s**

Therefore, those individuals having the poorest contribution are discarded from the population. In Algorithm 12, we present a naive implementation of this process. Although the computational cost of calculating the exact hypervolume is exponential with the scaling of the objectives, the Walking Fish Group (WFG)[1] has proposed clever implementations where, in practice, the real performance is unrelated to this worst case complexity [143]. Of our particular interest is the Incremental Hypervolume Algorithm (IWFG) [142, 23], designed for determining which point in a set contributes least to the hypervolume. This algorithm uses several ideas to provide a substantial speed up. The most recent implementation is the IWFG 1.01 [23], which was released in November 2015. This version reported outstanding performance for even large fronts, being significantly faster than previous approaches in many-objective optimization problems [10]. However, this important version has gone unnoticed by the research community. Popular frameworks of evolutionary multi-objective optimization, such as jMetal[2] or MOEA Framework[3] do not have this update, and still, rely on the naive implementation of Algorithm 12. We believe that this omission is because of the occurrence of an error, which is triggered when integrating IWFG 1.01 into a MOEA.

A.2 IWFG 1.01 Algorithm

In Algorithm 13, we reproduce the pseudocode of IWFG 1.01 [23]. This improved version consists of two phases: the slicing process (lines 1 to 6), and the full computation of the exclusive hypervolume of the least-contributing solution (lines 7 to 11). Here, *head* an *tail* are list functions.[4] In the first phase, *Rank heuristic* imposes the order in which objectives will be processed (in a worsening sequence). The overall hypervolume is then processed in "slices" made by cuts along the corresponding objective. Those slices related to a solution **a** are stored in the list $S[\mathbf{a}]$ (line 3). The elements of this list are assumed to be ordered by size from the largest to the smallest. In lines 4 and 5, the biggest slice of a solution **a** is successively divided by making

[1] `http://www.wfg.csse.uwa.edu.au/hypervolume`
[2] `http://jmetal.github.io/jMetal`
[3] `http://moeaframework.org`
[4] $\text{head}([a, b, c, d]) := a$ and $\text{tail}([a, b, c, d]) := [b, c, d]$.

Algorithm 13 Incremental Hypervolume IWFG 1.01

Input: $A \subset Z$ set of solutions, depth $k \in \mathbb{N}$
Output: Solution $\mathbf{s}$ that contributes the least to $HV(A)$

```
1: for all a in A do
2:    Sort the objectives of a according to Rank heuristic
3:    S[a] ← the slices for a at the top level m
4:    for d = 1 to k − 1 do
5:       S[a] ← slice(head(S[a]), m − d) ∪ tail(S[a])
6:    p[a] ← ExcHV(a, head(S[a]))
7: s ← arg min_{a∈A} p[a]
8: while S[s] ≠ [ ] do
9:    p[s] ← p[s] + ExcHV(s, head(S[s]))
10:   S[s] ← tail(S[s])
11:   s ← arg min_{a∈A} p[a]
12: return s
```

$k-1$ cuts along the remaining objectives. These sub-slices are reinserted into the list $S[\mathbf{a}]$. In line 6, for each solution, the partial exclusive hypervolume relative to the biggest slice is determined. In the second phase, a greedy approach is adopted, named "best-first" queuing mechanism. The idea is to process at each iteration the solution $\mathbf{s}$ with the smallest partial hypervolume until its list of slices has been completely processed. Moreover, instead of using the expression (A.1) for calculating the exclusive hypervolume, IWFG 1.01 uses a more efficient mechanism [11]:

$$ExcHV(\mathbf{p}, A) := IncHV(\mathbf{p}) - HV(\text{NDS}(B)), \tag{A.2}$$

where

$$B := \{\text{limit}(\mathbf{p}, \mathbf{a}) \mid \mathbf{a} \in A\}, \tag{A.3}$$

$$\begin{aligned} &\text{limit}(< p_1, \ldots, p_m >, < a_1, \ldots, a_m >) \\ &\quad := < \text{worse}(p_1, a_1), \ldots, \text{worse}(p_m, a_m) >, \end{aligned} \tag{A.4}$$

and $IncHV$ is the *inclusive hypervolume*, which is used to denote the size of the part of objective space dominated by a solution $\mathbf{p}$ alone, that is:

$$IncHV(\mathbf{p}) := HV(\{\mathbf{p}\}). \tag{A.5}$$

In Figure A.1, we illustrate the two different ways to compute the exclusive hypervolume. It is worth noticing that expression (A.2) calculates the hypervolume of only two solutions, whereas expression (A.1) considers six solutions. This computational effort reduction is because of the filtering of non-dominated solutions.

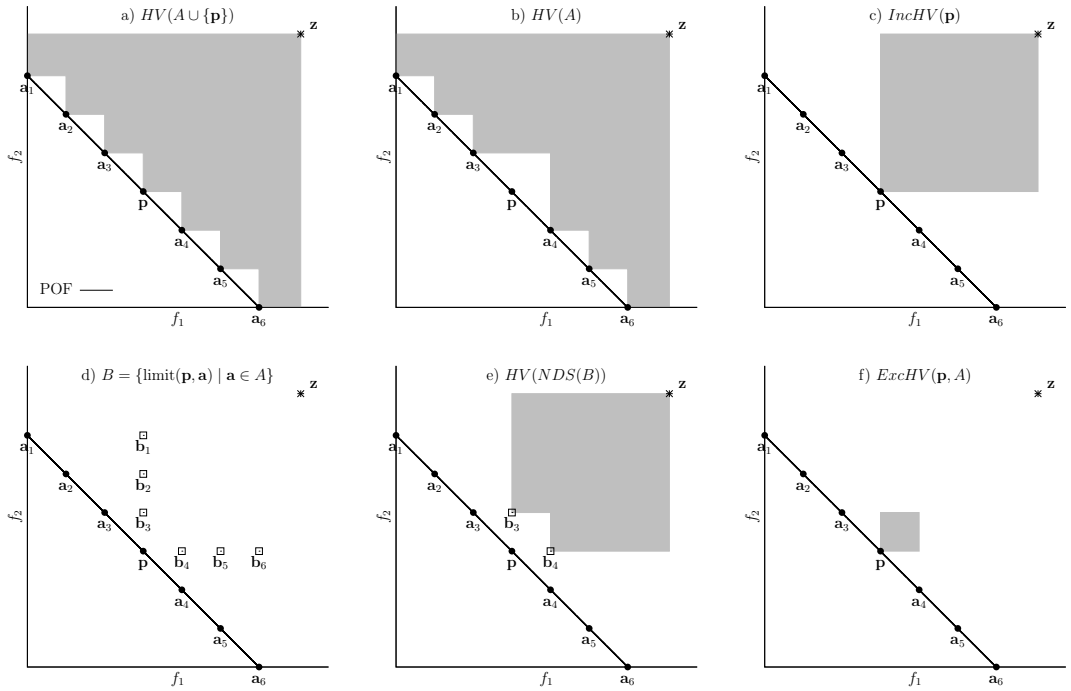


Figure A.1: Steps for the calculation of the exclusive hypervolume using the naive way $HV(A \cup \{\mathbf{p}\}) - HV(A)$ and the efficient way $IncHV(\mathbf{p}) - HV(NDS(B))$, where $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $B = \{b_1, b_2, b_3, b_4, b_5, b_6\}$.

```
> cat sample.dat
#
0.00 0.00 0.00 0.00 1.00
0.51 0.46 0.73 0.00 0.00
0.47 0.43 0.46 0.45 0.42
0.00 0.47 0.54 0.70 0.00
0.51 0.46 0.73 0.00 0.00
0.00 0.81 0.00 0.58 0.00
0.93 0.00 0.36 0.00 0.00
#
> ./iwfg sample.dat 1.1 1.1 1.1 1.1 1.1
Segmentation fault (core dumped)
```

Figure A.2: Error reproduction in the IWFG 1.01 component.

A.3 The Error

In Figure A.2, we reproduce a common error of Algorithm IWFG 1.01 using data produced by MOEAs. The program receives as input the file "sample.dat", which contains one front separated by #, and a reference point with five objectives. As can be noticed, the component throws a fatal error causing an abnormal termination. In this case, the segmentation fault is raised by hardware, which has memory protection, notifying the operating system that the program `iwfg` attempts to access a memory location that is not allowed.

In order to track the source of this error, we relied on the debugging tools Valgrind[5] and gdb[6]. We found that the problem lies in the `binarySearch` function of Figure A.3 since it does not contemplate the situation of identical solutions, as it is the case for the objective vector $(0.51, 0.46, 0.73, 0.00, 0.00)$ from our example in Figure A.2. In evolutionary multi-objective optimization, these copies are known as *indifferent solutions* [21, p. 244], and are occasionally present in a population when variation operators are not applied, so the offspring become clones of the parents. According to expression (2.5) on page 11, indifferent solutions are considered non-dominated to each other, so the requirement of the IWFG 1.0 to accept only fronts with non-dominated solutions is still fulfilled. The purpose of the function `int binarySearch(POINT p,int d)` is to locate the index `i` at which the solution `p` resides in the array of memory addresses `fsorted[d].points[i]`, assuming that elements are already sorted by the given objective `d` from the highest to the lowest value. The ordering relation is achieved by the function `int greaterorder(&p,&q)`, which numerically compares two solutions. This function returns `-1` if the d^{th} objective of `p` is greater than the d^{th} objective of `q`. In the opposite case, it returns `1`, and if they have the same value, the remainder objectives are inspected in the same way using the order imposed by the Rank heuristic. In the case of indifferent solutions, `greaterorder` returns `0`.

During the search, the error originates when the first occurrence of a repeated solution does not match with the memory address of `p`. Thus, the binary search focuses on the upper half of the array in lieu of examining adjacent elements. So, if the solution is not found in this half, the function returns `-1`, which is an invalid index. It is important to mention that the error happens only from three objectives onwards.[7] The `binarySearch` function is invoked by the Rank heuristic, which stores the returning misinformation. The slicing process accesses the indexes, and it is until then when the fault occurs.

One possible solution to this issue is to include the case when `greaterorder` recognizes two identical solutions. In Figure A.4, we present the source code of our proposed correction, named `ourBinarySearch`. Once a duplicated objective vector is found, in lines 16 to 32, adjacent memory locations are inspected until there is a match with the address of `p`. In Figure A.5, we show the right output of our previous example using the proposed function. It is worth mentioning that one of the duplicated solutions is suggested for removal.

The computational complexity of `binarySearch` is $O(m \lg |P|)$, where m represents the number of objectives and $|P|$ is the number of non-dominated solutions in the front. For `ourBinarySearch` is $O(m(\lg |P| + k))$, where k denotes the number of indifferent solutions. Here, the worst case occurs when all elements are repeated, so the computational complexity is $O(m|P|)$. However, this is very unlikely, in the average case $k << |P|$. Thus, the complexity of our proposed function remains as the origi-

[5] `http://valgrind.org`

[6] `https://www.gnu.org/software/gdb`

[7] For two objectives the exclusive hypervolume is computed.

```
1int binarySearch(POINT p, int d) {
2  int min = 0;
3  int max = fsorted[d].nPoints-1;
4  gorder = torder[d];
5
6  while(min <= max) {
7    int mid = (max+min)/2;
8    if(p.objectives==fsorted[d].points[mid].objectives)
9      return mid;
10   else if(greaterorder(&p,&fsorted[d].points[mid])==-1)
11     max = mid-1;
12   else
13     min = mid+1;
14 }
15 return -1;
16}
```

Figure A.3: Source code of the original binarySearch function.

```
1int ourBinarySearch(POINT p, int d) {
2  int i, r;
3  int min = 0;
4  int max = fsorted[d].nPoints-1;
5  gorder = torder[d];
6
7  while(min <= max) {
8    int mid = (max+min)/2;
9
10   if(p.objectives==fsorted[d].points[mid].objectives)
11     return mid;
12   else if((r = greaterorder(&p, &fsorted[d].points[mid])) == -1)
13     max = mid-1;
14   else if(r == 1)
15     min = mid+1;
16   else { /* (r = 0) duplicated solutions */
17     /* check solutions on the left of mid */
18     i = mid - 1;
19     while(i >= min && greaterorder(&p, &fsorted[d].points[i]) == 0) {
20       if(p.objectives == fsorted[d].points[i].objectives)
21         return i;
22       i--;
23     }
24     /* check solutions on the right of mid */
25     i = mid + 1;
26     while(i <= max && greaterorder(&p, &fsorted[d].points[i]) == 0) {
27       if(p.objectives == fsorted[d].points[i].objectives)
28         return i;
29       i++;
30     }
31   }
32 }
33 return -1;
34}
```

Figure A.4: Source code of our proposed binarySearch function.

nal one.

```
> ./iwfg sample.dat 1.1 1.1 1.1 1.1 1.1
mehv(1) = 0.0000000000000000
Smallest: 0.5100000000 0.4600000000 0.7300000000 0.0000000000 0.0000000000
Total time = 0.000000 (s)
```

Figure A.5: Output of the IWFG 1.01 component using the function ourBinarySearch.

A.4 Experimental Results

We compared the performance of IWFG 1.01 versus the naive approach of Algorithm 12. This latter version is denoted here as IWFG 1.00. Both methods calculate the hypervolume using the method described in [143]. These versions were coupled to SMS-EMOA, considering the DTLZ1, DTLZ2, and DTLZ7 test problems (see Section B.3 on page 156). The variation operators were Polynomial-based mutation and Simulated Binary Crossover (SBX). For the mutation operator, its probability and distribution index were set to $1/n$ and 20, respectively. For the crossover operator, these parameters varied according to the number of objectives: for two objectives we adopted 0.9 and 20, whereas for higher dimensionality we adopted 1.0 and 30. In all cases, the population size was set to 100 individuals. The maximum number of evaluations (1×10^3) was set to 40, 60, 70, 80, 80, 90 for 2 to 7 objectives, respectively.

For the performance assessment, we relied on the hypervolume indicator using the reference point $(2, 2, \ldots)$ for DTLZ1,2 and $(2, 2, \ldots, 2m + 1)$ for DTLZ7. In all experiments, we performed 10 independent runs. We applied the Wilcoxon rank sum test (two-tailed) to the mean hypervolume indicator values, in order to determine if the distributions of both variants were identical or different at the confidence interval of 99%. Finally, executions have been done over the GNU/Linux Xiuhcoatl Cluster[8] of 72 nodes with 252 GB of RAM and InfiniBand interconnection network. Each processor is a 32-core AMD Opteron(TM) Processor 6274 1.36 GHz. Algorithms were implemented in C language and compiled with gcc 4.4.7 -O3.

In Figure A.6, we show the median execution time of the two variants. As can be observed, SMS-EMOA IWFG 1.01 spent much less computational time than SMS-EMOA IWFG 1.00. Furthermore, time reduction becomes more significant as the number of objectives increases. Regarding the quality of the solutions, it is worth mentioning that both versions produce slightly different Pareto front approximations even though we used the same random seeds. This occurs because during the survival selection process, several individuals may have the same hypervolume contribution. Thus, the choice of the methods depends on the way in which the population is sorted. In spite of this, there is no significant difference in quality, as shown in Table A.1.

[8] `http://clusterhibrido.cinvestav.mx`

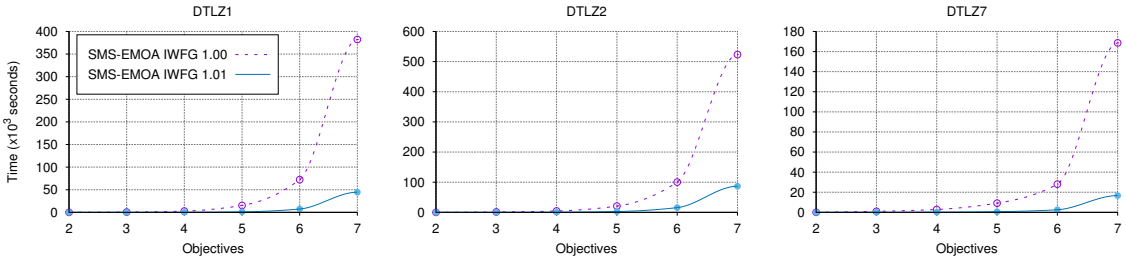


Figure A.6: Execution time of the two versions on some instances of the DTLZ benchmark.

Table A.1: Median and standard deviation of the hypervolume indicator. If p-value > 0.01, then it means that the two samples are equivalent.

m	**SMS-EMOA IWFG 1.00**		**SMS-EMOA IWFG 1.01**		**Statistical Test (p-value)**
			DTLZ1		
2	3.873652e+00	1.81e-04	3.873610e+00	1.47e-04	9.10e-01
3	7.974010e+00	4.77e-05	7.974043e+00	8.25e-05	7.05e-01
4	1.599436e+01	3.06e-05	1.599437e+01	1.15e-05	2.39e-01
5	3.199857e+01	6.11e-05	3.199859e+01	5.11e-05	4.93e-01
6	6.399957e+01	2.06e-05	6.399956e+01	2.53e-05	5.16e-01
7	1.279999e+02	4.22e-05	1.279999e+02	4.82e-05	6.51e-01
			DTLZ2		
2	3.211015e+00	1.86e-05	3.211003e+00	2.64e-05	6.50e-01
3	7.427029e+00	5.39e-05	7.427018e+00	5.76e-05	8.50e-01
4	1.558050e+01	7.71e-05	1.558050e+01	7.71e-05	1.00e+00
5	3.168567e+01	7.59e-05	3.168567e+01	7.59e-05	1.00e+00
6	6.375871e+01	1.10e-04	6.375871e+01	1.10e-04	1.00e+00
7	1.278103e+02	1.43e-04	1.278103e+02	1.43e-04	1.00e+00
			DTLZ7		
2	4.418199e+00	1.60e-05	4.418197e+00	1.23e-05	8.80e-01
3	1.351650e+01	1.51e+00	1.351650e+01	1.51e+00	1.00e+00
4	3.455925e+01	4.64e+00	3.455925e+01	4.62e+00	9.40e-01
5	6.993100e+01	5.14e+00	6.982480e+01	5.14e+00	9.40e-01
6	1.242848e+02	1.55e+01	1.242848e+02	1.55e+01	1.00e+00
7	2.469422e+02	4.84e+01	2.469422e+02	4.84e+01	1.00e+00

A.5 Summary

Recently, an optimized version of the incremental hypervolume algorithm of the Walking Fish Group was proposed. This algorithm determines the solution that contributes the least to the hypervolume of a non-dominated set. However, its use has been limited due to a bug in its implementation. We observed that this error occurs during the slicing process, specifically in the function `binarySearch`, where duplicated solutions are not considered for problems with more than two objectives. In this appendix, we have proposed a corrected version of such function, which has an average-case complexity of $O(m \lg |P|)$, where m denotes the number of objectives and $|P|$ the population size. Clearly, there are other possible solutions to this issue, such as to remove duplicated solutions before calculating the incremental hypervolume. However,

we have presented the one that we believe is the easiest to update in the component while keeping a low computational cost. The source code of all the algorithms were developed in EMO Project (see Chapter 6 on page 89), being the IWFG modules thread-safe.

Appendix B

Test Problems

With the aim of having a better understanding of the working principles of optimizers, there are several benchmarks that have been suggested in the field of multi-objective optimization. These artificial test problems examine the ability to control difficulties in both converging to the true Pareto optimal front and in maintaining a widely distributed set of solutions. Moreover, they offer many advantages over real-world problems, such as scalability, knowledge of the exact shape and location of the resulting Pareto optimal front, fast execution time, as well as ease of understanding, implementation and visualization.

In this Appendix, we define the test problems for real-value encoding that were adopted in our experiments. All of them were implemented in EMO Project. Section B.1 starts with their characterization. Section B.2 is dedicated to the Zitzler-Deb-Thiele (ZDT) benchmark [153]. Section B.3 describes the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [31]. Section B.4 presents the Walking-Fish-Group benchmark (WFG) [64]. Section B.5 includes the inverted problems of the DTLZ and WFG benchmarks [73]. Section B.6 presents some selected test problems with inequality constraints. Finally, Section B.7 concludes this appendix.

B.1 Difficulties in Multi-Objective Optimization

This section provides fundamental concepts about the features that may be present in MOPs. Some of them represent a real challenge for optimizers. For a comprehensive review on the topic, readers can consult the works of Huband et al. [64] and Deb [25].

In the following, we refer to *fitness landscape* as the mapping from the decision space ($\mathcal{X}$) to the objective space ($\mathcal{Z}$). As illustrated in Figure B.1(a), the fitness landscape can be *one-to-one* or *many-to-one*. In the former case, for each decision variable in $\mathcal{X}$, there is a different objective vector in $\mathcal{Z}$. In the second case, two or more decision variables in $\mathcal{X}$ are associated to the same objective vector in $\mathcal{Z}$. The many-to-one mapping presents more difficulties to optimizers since choices must be made among the decision vectors. This becomes critical in *flat regions*, where small perturbations of the decision variables do not change the objective values (see Figure B.1b).

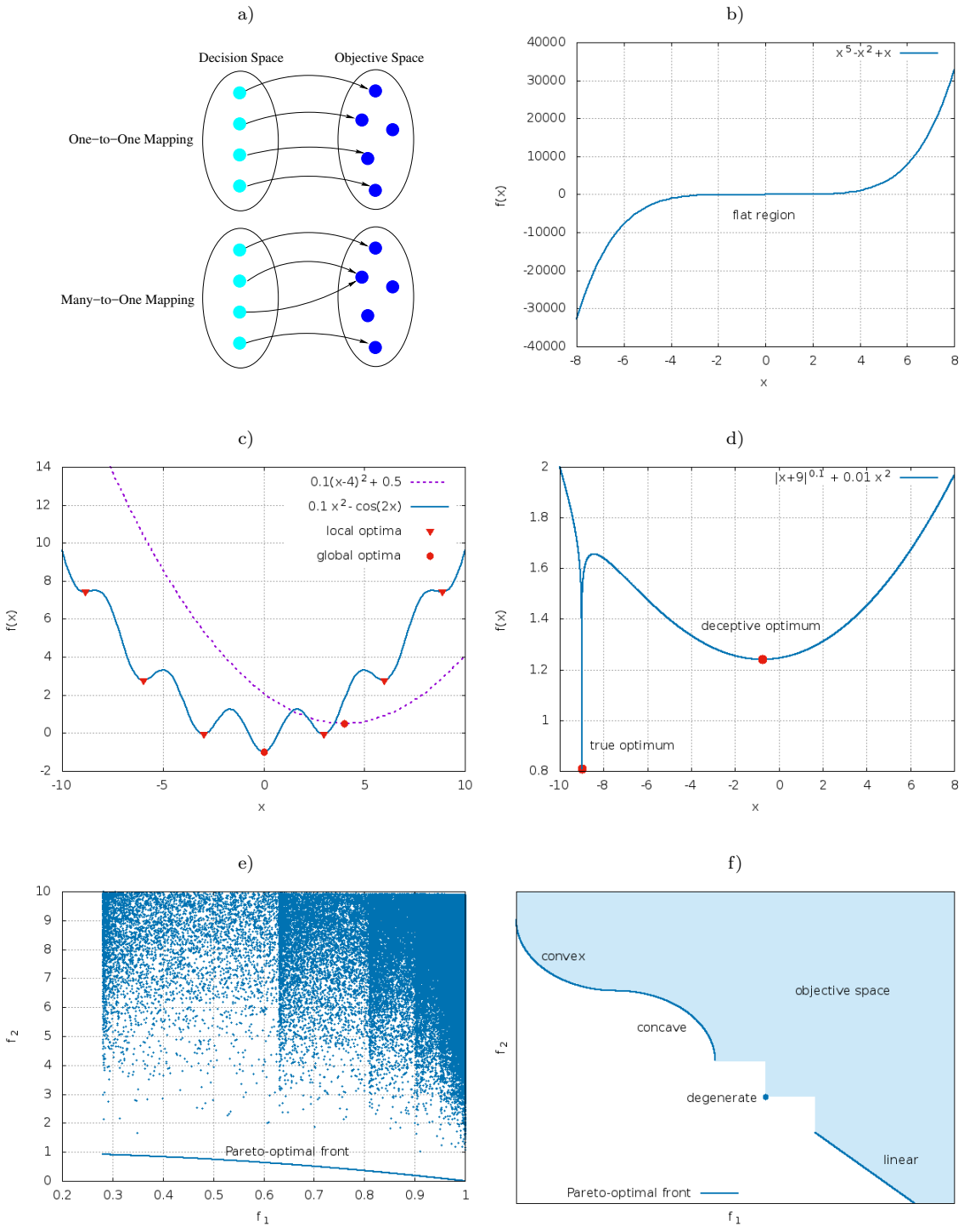


Figure B.1: Some features of multi-objective optimization problems: a) mapping between $\mathcal{X}$ and $\mathcal{Z}$, b) illustration of a flat function, c) examples of uni-modal and multi-modal functions, d) a deceptive function, e) 100,000 random decision vectors from the biased ZDT1 test problem, and f) an hypothetical disconnected and mixed Pareto front.

Another feature of fitness landscapes is modality as depicted in Figure B.1c. In a *uni-modal* MOP, every objective function has a single optimum. In a *multi-modal* MOP, at least one objective function has multiple local optima. A special case of multi-modality is a *deceptive* MOP, where one objective function has at least two optima, a true optimum and a deceptive optimum, but the majority of the decision space favors the deceptive optimum (see Figure B.1d). Thus, the global optimum is in an unlikely place to explore. In general, multi-modal problems are difficult to solve since optimizers can get stuck in local optima.

An important aspect of the fitness landscape is its distribution. We would expect that an evenly distributed sample of the decision vectors in $\mathcal{X}$ maps to an evenly distributed set of objective vectors in $\mathcal{Z}$. However, in practice, this does not occur. A MOP is said to be *biased* when there is a significant density variation between the Pareto optimal set and the Pareto optimal front (see Figure B.1e). Bias has a natural impact on the search process because most optimizers attempt to achieve an even spread of solutions in objective space.

A decision variable x_i is *separable* with respect to an objective function if the Pareto optimal set is the same as such variable changes. Otherwise, x_i, is *non-separable*. If all the variables are independent, then the objective is said to be separable. Thus, each decision variable can be optimized independently. Similarly, a MOP is non-separable if at least one objective function is non-separable. Otherwise, the MOP is separable. In general, separable MOPs are relatively easy to solve, when compared with their non-separable version.

Decision variables can also be categorized regarding their relationship with the fitness landscape. A decision variable x_i is called *distance-related parameter* if it only controls convergence to the Pareto optimal front[1]. A decision variable x_i is called *position-related parameter* if it controls diversity of the current Pareto front[2]. All variables that are neither position nor distance related parameters are *mixed parameters*. Thus, modifying mixed parameters on their own can result in a change in position or distance.

The Pareto optimal front can have a wide variety of geometries. Recall that a set is *convex* if and only if it covers its convex hull. Conversely, it is *concave* if and only if it is covered by its convex hull. A set is *strictly convex* (respectively, *strictly concave*) if it is convex (respectively, concave) and not concave (respectively, convex). A *linear* set is one that is both concave and convex. A *mixed* front is one with connected subsets that are each strictly convex, strictly concave, or linear, but not all of the same type.

A *degenerate* Pareto front is a front that is of lower dimension than the objective space in which it is embedded, less one. For example, a front that is a line segment in a three objective problem is degenerate. Conversely, a two-dimensional front in a three objective problem is not degenerate. Degenerate fronts can cause problems for some algorithms. For example, methods employed to encourage an even spread

[1]Modifying x_i may result in a solution that dominates or is dominated by the original one.

[2]Perturbing x_i may result in a solution that is incomparable with respect to the original one.

Table B.1: Properties of the benchmarks.

Problem	**Separability**	**Modality**	**Geometry**	**Bias**
ZDT1	separable	uni	convex	no
ZDT2	separable	uni	concave	no
ZDT3	separable	multi	disconnected	no
ZDT4	separable	multi	convex	no
ZDT6	separable	multi	concave	polynomial
DTLZ1	separable	multi	linear	no
DTLZ2	separable	uni	concave	no
DTLZ3	separable	multi	concave	no
DTLZ4	separable	uni	concave	polynomial
DTLZ5	unknown	uni	arc, degenerated	parameter dependent
DTLZ6	unknown	uni	arc, degenerated	parameter dependent
DTLZ7	$f_{1:m-1}$ not applicable f_m separable	$f_{1:m-1}$ uni f_m multi	disconnected, mixed	no
WFG1	separable	uni	$f_{1:m-1}$ convex f_m mixed	polynomial, flat
WFG2	non-separable	$f_{1:m-1}$ uni f_m multi	convex, disconnected	no
WFG3	non-separable	uni	linear, degenerated	no
WFG4	separable	multi	concave	no
WFG5	separable	deceptive	concave	no
WFG6	non-separable	uni	concave	no
WFG7	separable	uni	concave	parameter dependent
WFG8	non-separable	uni	concave	parameter dependent
WFG9	non-separable	multi, deceptive	concave	parameter dependent

of solutions across the Pareto optimal front might operate differently if the front effectively employs fewer dimensions than expected.

We are also interested in whether a front is a *connected* set. In the literature, a front that is a disconnected set is often referred to as *discontinuous*. Disconnected fronts will test an algorithm's ability to maintain subpopulations in different Pareto optimal regions.

Examples of different Pareto front geometries are shown in Figure B.1f. Finally, in Table B.1, we summarize the features of the benchmarks that are described in the following sections.

B.2 Zitzler-Deb-Thiele-Laumanns Test Problems

The Zitzler-Deb-Thiele (ZDT) benchmark [153] has six bi-objective functions. Here, we only consider the five continuous problems of the benchmark. The range of the decision variables is $x_1 \in [0, 1]$ and $x_2, \ldots, x_{10} \in [-5, 5]$ for ZDT4; $x_1, \ldots, x_{30} \in [0, 1]$ for ZDT1-3; and $x_1, \ldots, x_{10} \in [0, 1]$ for ZDT6. The Pareto optimal fronts, shown in Figures B.2 and B.3, are formed with $g(\mathbf{x}) = 1$.

ZDT1

$$f_1(\mathbf{x}) = x_1,$$
$$f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \sqrt{f_1/g(\mathbf{x})}\right)$$
$$g(\mathbf{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

ZDT2

$$f_1(\mathbf{x}) = x_1$$
$$f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \left(f_1/g(\mathbf{x})\right)^2\right)$$
$$g(\mathbf{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

ZDT3

$$f_1(\mathbf{x}) = x_1$$
$$f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \sqrt{\frac{f_1}{g(\mathbf{x})}} - \frac{f_1}{g(\mathbf{x})}\sin(10\pi f_1)\right)$$
$$g(\mathbf{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

ZDT4

$$f_1(\mathbf{x}) = x_1$$
$$f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \sqrt{\frac{f_1}{g(\mathbf{x})}}\right)$$
$$g(\mathbf{x}) = 10n - 9 + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right)$$

ZDT6

$$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$
$$f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \left(\frac{f_1}{g(\mathbf{x})}\right)^2\right)$$
$$g(\mathbf{x}) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{9}\right)^{0.25}$$

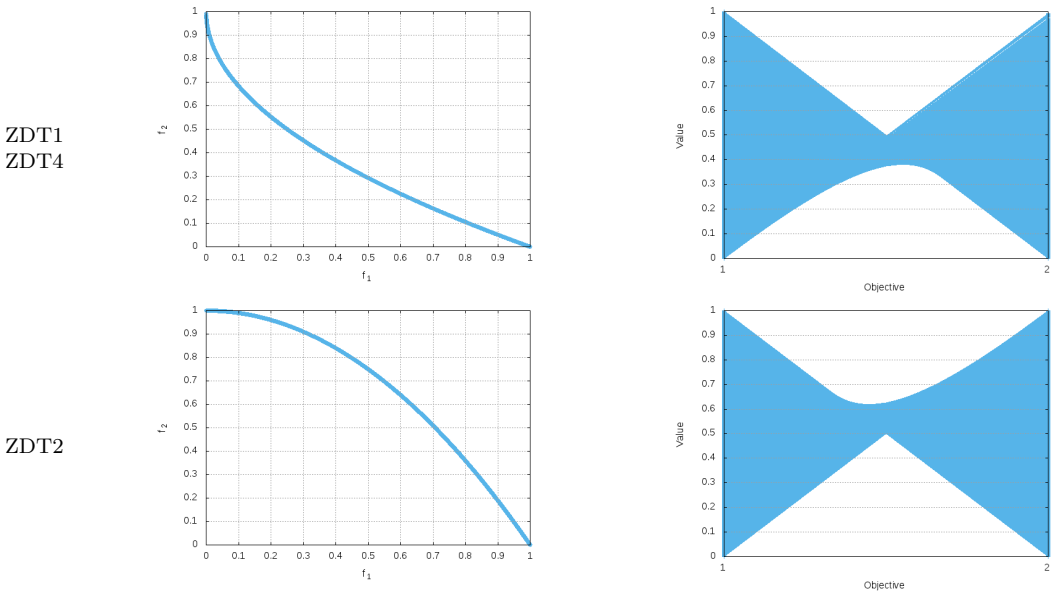
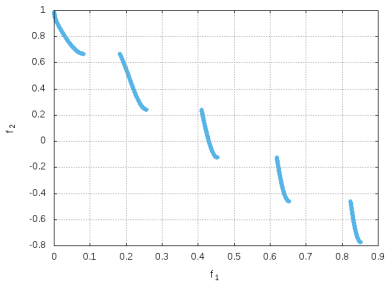
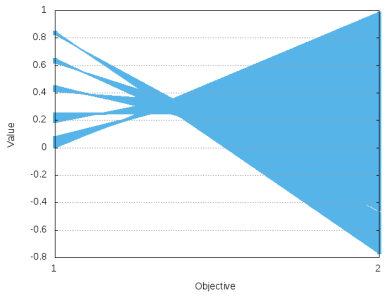


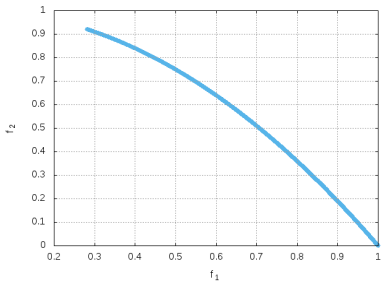
Figure B.2: Pareto optimal fronts and parallel coordinates of the ZDT benchmark.

ZDT3





ZDT6



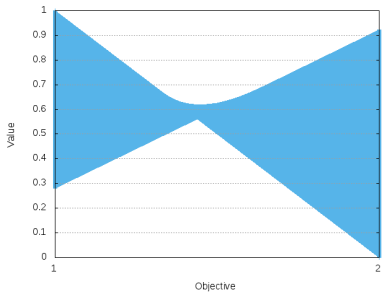


Figure B.3: ZDT benchmark (cont'd).

B.3 Deb-Thiele-Laumanns-Zitzler Test Problems

The Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [31] includes nine representative test problems, which are scalable to any number of decision variables and objectives. Here, we only consider the seven unconstrained problems. In all cases, the range of the decision variables is $[0, 1]$. The number of decision variables is given by $n = m+k-1$, where m represents the number of objectives and k is the number of distance-related parameters. The distance vector of k entries is defined as $\mathbf{y} = \{x_m, x_{m+1}, \ldots, x_n\}$, considering the decision vector $\mathbf{x} = \{x_1, \ldots, x_{m-1}, x_m, \ldots, x_n\}$. In [31], authors suggest k-values of 5 for DTLZ1, 10 for DTLZ2-6, and 20 for DTLZ7. In the case of DTLZ4, $\alpha = 100$ is recommended.

DTLZ1	**DTLZ2**
$f_1(\mathbf{x}) = 0.5\,(1 + g(\mathbf{y}))\prod_{i=1}^{m-1} x_i$	$f_1(\mathbf{x}) = (1 + g(\mathbf{y}))\prod_{i=1}^{m-1} \cos(x_i\pi/2)$
$f_{j=2:m-1}(\mathbf{x}) = 0.5\,(1 + g(\mathbf{y}))\,(1 - x_{m-j+1})\prod_{i=1}^{m-j} x_i$	$f_{j=2:m-1}(\mathbf{x}) = (1 + g(\mathbf{y}))\left(\prod_{i=1}^{m-j} \cos(x_i\pi/2)\right)\sin(x_{m-j+1}\pi/2)$
$f_m(\mathbf{x}) = 0.5\,(1 + g(\mathbf{y}))\,(1 - x_1)$	$f_m(\mathbf{x}) = (1 + g(\mathbf{y}))\sin(x_1\pi/2)$
$g(\mathbf{y}) = 100\left(k + \sum_{i=1}^{k}(y_i - 0.5)^2 - \cos(20\pi(y_i - 0.5))\right)$	$g(\mathbf{y}) = \sum_{i=1}^{k}(y_i - 0.5)^2$

The Pareto optimal solution corresponds to $\mathbf{y} = (0.5, 0.5, \ldots)^T$ for DTLZ1-5 and $\mathbf{y} = (0, 0, \ldots)^T$ for DTLZ6-7. Moreover, it is fullfilled that $\sum_{i=1}^{m} f_i = 0.5$ for DTLZ1, and $\sum_{i=1}^{m} (f_i)^2 = 1$ for DTLZ2-4. DTLZ5-6 were originally proposed as many-objective test problems with degenerate Pareto fronts. However, their Pareto fronts are not degenerate when they have four or more objectives [64]. The Pareto optimal fronts of this benchmark are shown in Figure B.4.

DTLZ3

$$f_1(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \prod_{i=1}^{m-1} \cos(x_i \pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \left(\prod_{i=1}^{m-j} \cos(x_i \pi/2) \right) \sin(x_{m-j+1} \pi/2)$$

$$f_m(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \sin(x_1 \pi/2)$$

$$g(\mathbf{y}) = 100 \left(k + \sum_{i=1}^{k} (y_i - 0.5)^2 - \cos(20\pi(y_i - 0.5)) \right)$$

DTLZ4

$$f_1(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \prod_{i=1}^{m-1} \cos(x_i^{\alpha} \pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \left(\prod_{i=1}^{m-j} \cos(x_i^{\alpha} \pi/2) \right) \sin(x_{m-j+1}^{\alpha} \pi/2)$$

$$f_m(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \sin(x_1^{\alpha} \pi/2)$$

$$g(\mathbf{y}) = \sum_{i=1}^{k} (y_i - 0.5)^2$$

DTLZ5

$$f_1(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \prod_{i=1}^{m-1} \cos(\theta_i \pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \left(\prod_{i=1}^{m-j} \cos(\theta_i \pi/2) \right) \sin(\theta_{m-j+1} \pi/2)$$

$$f_m(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \sin(\theta_1 \pi/2)$$

$$\theta_i = \begin{cases} x_i, & \text{if } i = 1 \\ \frac{1+2g(\mathbf{y})}{2(1+g(\mathbf{y}))} x_i, & \text{otherwise} \end{cases}$$

$$g(\mathbf{y}) = \sum_{i=1}^{k} (y_i - 0.5)^2$$

DTLZ6

$$f_1(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \prod_{i=1}^{m-1} \cos(\theta_i \pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \left(\prod_{i=1}^{m-j} \cos(\theta_i \pi/2) \right) \sin(\theta_{m-j+1} \pi/2)$$

$$f_m(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \sin(\theta_1 \pi/2)$$

$$\theta_i = \begin{cases} x_i, & \text{if } i = 1 \\ \frac{1+2g(\mathbf{y})}{2(1+g(\mathbf{y}))} x_i, & \text{otherwise} \end{cases}$$

$$g(\mathbf{y}) = \sum_{i=1}^{k} y_i^{0.1}$$

DTLZ7

$$f_{j=1:m-1}(\mathbf{x}) = x_j$$

$$f_m(\mathbf{x}) = \big(1 + g(\mathbf{y})\big) \left(m - \sum_{i=1}^{m-1} \left[\frac{f_i}{1 + g(\mathbf{y})} \big(1 + \sin(3\pi f_i)\big) \right] \right)$$

$$g(\mathbf{y}) = 1 + \frac{9}{k} \sum_{i=1}^{k} y_i$$

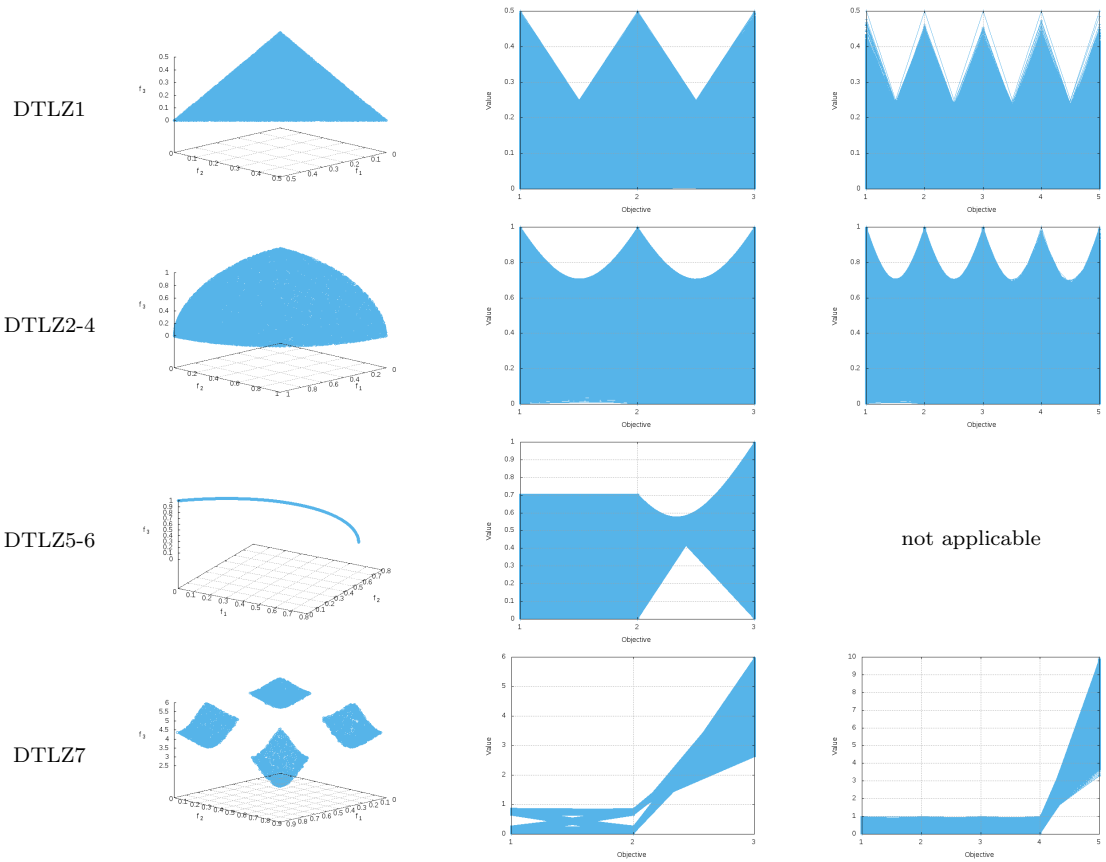


Figure B.4: Pareto optimal fronts and parallel coordinates of the DTLZ benchmark.

B.4 Walking Fish Group Test Problems

The Walking-Fish-Group (WFG) benchmark, proposed by Huband et al. [64], is composed of nine multi-objective test problems that are scalable with respect to both objectives and variables (see Figure B.5). Each problem is defined in terms of an underlying vector of parameters $\mathbf{x} \in \mathbb{R}^m$ that defines the objective space (m represents the number of objectives). All $x_i \in \mathbf{x}$ have the domain $[0, 1]$. The vector $\mathbf{x}$ is derived via a series of transition vectors from a decision vector $\mathbf{z} = \{z_1, \ldots, z_k, z_{k+1}, \ldots, z_n\}$. The domain of all $z_i \in \mathbf{z}$ is $[0, 2i]$. The number of decision variables is given by $n = k + l$, where it should be satisfied that $n \geq m$. The first $k \in \{m-1, 2(m-1), 3(m-1), \ldots\}$ decision variables are the position-related parameters and the last $l \in \{1, 2, \ldots\}$ decision variables are the distance-related parameters. The optimizer directly manipulates $\mathbf{z}$, through which $\mathbf{x}$ is indirectly manipulated. For WFG1-WFG7, a solution is Pareto optimal if $z_{i=k+1:n} = (2i)0.35$. For WFG8, it is required that all of:

$$z_{i=k+1:n} = (2i)0.35^{\left(0.02+49.98\left(\frac{0.98}{49.98}-(1-2u)\left|\lfloor 0.5-u\rfloor+\frac{0.98}{49.98}\right|\right)\right)^{-1}},$$
$$u = \text{r\_sum}(\{z_1, \ldots, z_{i-1}\}, \{1, \ldots, 1\}).$$

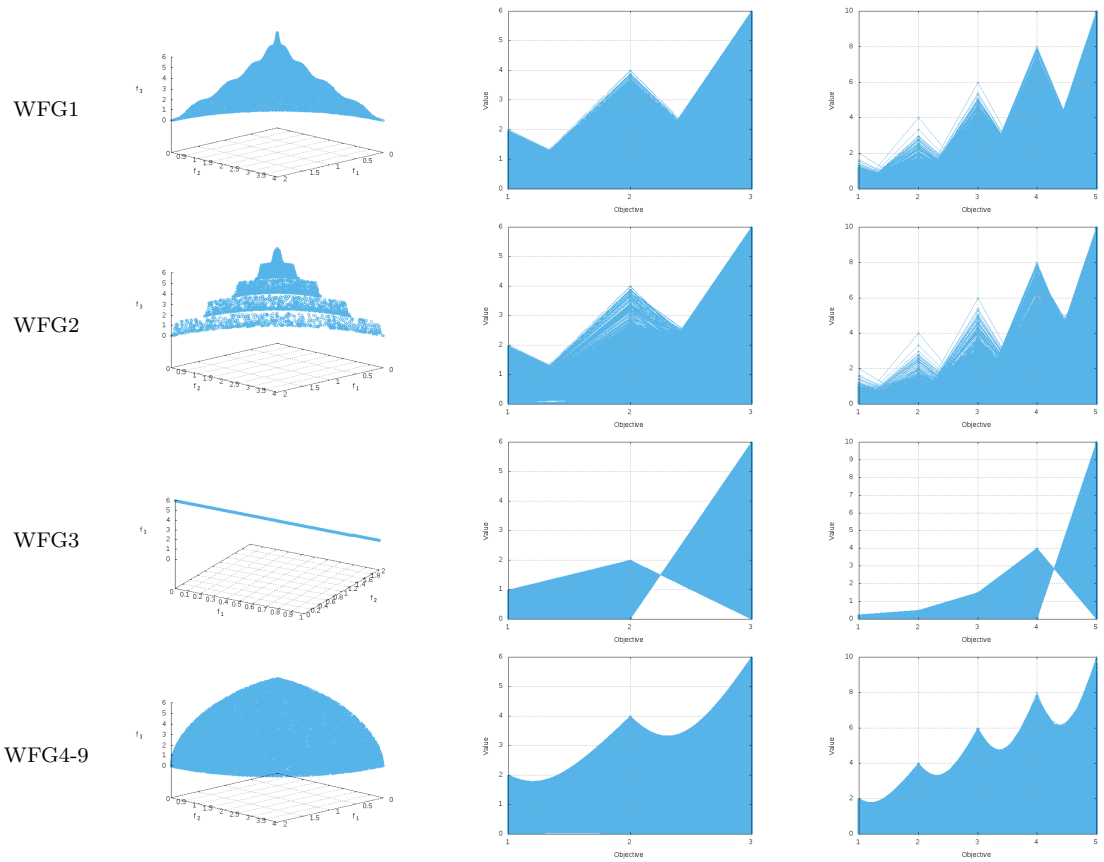


Figure B.5: Pareto optimal fronts and parallel coordinates of the WFG benchmark.

To obtain a Pareto optimal solution, the position should first be determined by setting $z_{1:k}$ appropriately. The required distance-related parameter values can then be calculated by first determining z_{k+1}, then z_{k+2}, and so on, until z_n has been calculated.

In the case of WFG9, for a solution to be Pareto optimal, it is required that all of:

$$z_{i=k+1:n} = (2i) \begin{cases} 0.35^{\left(0.02+1.96\ \text{r\_sum}\left(\{z_{i+1},\ldots,z_n\},\{1,\ldots,1\}\right)\right)^{-1}}, & i \neq n \\ 0.35, & i = n, \end{cases}$$

which can be found by first determining z_n, then z_{n-1}, and so on, until the required value for z_{k+1} is determined. Once the optimal values for $z_{k+1:n}$ are determined, the position-related parameters can be varied arbitrarily to obtain different Pareto optimal solutions.

The transformation functions, which map parameters with domain $[0, 1]$ onto the range $[0, 1]$, add complexity to the problem. There are three types of transformation functions: *bias, shift and reduction functions*. Bias and shift functions only employ one parameter, whereas reduction functions can employ many. Bias transformations have a natural impact on the search process by biasing the fitness landscape. Shift transformations move the location of optimal values, and are used to apply a linear

shift, or to produce deceptive and multi-modal problems. Reduction transformations are used to produce non-separability of the problem. In the following, we define such transformation functions.

Bias: Polynomial

When $\alpha > 1$ or when $\alpha < 1$, y is biased towards zero or towards one, respectively.

$$\text{b\_poly}(y, \alpha) = y^{\alpha}, \tag{B.1}$$

where $\alpha > 0$ and $\alpha \neq 1$.

Bias: Flat Region

Values of y between B and C (the area of the flat region) are mapped to the value A.

$$\begin{aligned} \text{b\_flat}(y, A, B, C) = A &+ \min(0, \lfloor y - B \rfloor) \frac{A(B - y)}{B} \\ &- \min(0, \lfloor C - y \rfloor) \frac{(1 - A)(y - C)}{1 - C}, \end{aligned} \tag{B.2}$$

where $A, B, C \in [0, 1]$, $B < C$, $B = 0 \Rightarrow A = 0 \wedge C \neq 1$, and $C = 1 \Rightarrow A = 1 \wedge B \neq 0$.

Bias: Parameter Dependent

A, B, C, the parameter vector $\mathbf{w} \in [0, 1]^{|\mathbf{w}|}$, and the reduction function u together determine the degree to which y is biased by being raised to an associated power: values of $u(\mathbf{w}) \in [0, 0.5]$ are mapped linearly onto $[B, B + (C - B)A]$, and values of $u(\mathbf{w}) \in [0.5, 1]$ are mapped linearly onto $[B + (C - B)A, C]$.

$$\text{b\_param}(y, u(\mathbf{w}), A, B, C) = y^{B + (C - B)\left(A - (1 - 2u(\mathbf{w}))\left|\lfloor 0.5 - u(\mathbf{w}) \rfloor + A\right|\right)}, \tag{B.3}$$

where $A \in (0, 1)$, and $0 < B < C$.

Shift: Linear

$A \in (0, 1)$ is the value for which y is mapped to zero.

$$\text{s\_linear}(y, A) = \frac{|y - A|}{|\lfloor A - y \rfloor + A|}. \tag{B.4}$$

Shift: Deceptive

A is the value at which y is mapped to zero, and the global minimum of the transformation. B is the "aperture" size of the well/basin leading to the global minimum

at A, and C is the value of the deceptive minima (there are always two deceptive minima).

$$\text{s\_decept}(y, A, B, C) = 1 + \left(|y - A| - B\right)\left(\frac{\lfloor y - A + B \rfloor \left(1 - C + \frac{A-B}{B}\right)}{A - B} + \frac{\lfloor A + B - y \rfloor \left(1 - C + \frac{1-A-B}{B}\right)}{1 - A - B} + \frac{1}{B}\right), \tag{B.5}$$

where $A \in (0, 1)$, $0 < B \ll 1$, $0 < C \ll 1$, $A - B > 0$, and $A + B < 1$.

Shift: Multi-modal

A controls the number of minima, B controls the magnitude of the "hill sizes" of the multi-modality, and C is the value for which y is mapped to zero. When $B = 0$, $2A + 1$ values of y (one at C) are mapped to zero, and when $B \neq 0$, there are $2A$ local minima, and one global minimum at C. Larger values of A and smaller values of B create more difficult problems.

$$\text{s\_multy}(y, A, B, C) = \frac{1 + \cos\left[(4A + 2)\,\pi\left(0.5 - \frac{|y-C|}{2(\lfloor C-y \rfloor + C)}\right)\right] + 4B\left(\frac{|y-C|}{2(\lfloor C-y \rfloor + C)}\right)^2}{B + 2}, \tag{B.6}$$

where $A \in \{1, 2, \ldots\}$, $B \geq 0$, $(4A + 2)\pi > 4B$, and $C \in (0, 1)$.

Reduction: Weighted Sum

By varying the constants of the weight vector $\mathbf{w}$, optimizers can be forced to treat parameters differently.

$$\text{r\_sum}(\mathbf{y}, \mathbf{w}) = \frac{\left(\sum_{i=1}^{|\mathbf{y}|} w_i y_i\right)}{\sum_{i=1}^{|\mathbf{y}|} w_i}, \tag{B.7}$$

where $|w| = |y|$, and $w_1, \ldots, w_{|\mathbf{y}|>0}$.

Reduction: Non-separable

A controls the degree of non-separability (noting that $\text{r\_nonsep}(\mathbf{y}, 1) = \text{r\_sum}(\mathbf{y}, \{1, \ldots, \})$).

$$\text{r\_nonsep}(\mathbf{y}, A) = \frac{\sum_{j=1}^{|\mathbf{y}|}\left(y_j + \sum_{k=0}^{A-2}|y_j - y_{1+(j+k)mod|\mathbf{y}|}|\right)}{\frac{|\mathbf{y}|}{A}\lceil A/2 \rceil\left(1 + 2A - 2\lceil A/2 \rceil\right)}, \tag{B.8}$$

where $A \in \left\{1, \ldots, |\mathbf{y}|\right\}$, and $|\mathbf{y}| \text{mod} A = 0$.

WFG1

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\Big(1-\cos(x_i\pi/2)\Big)$$
$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\Big(1-\cos(x_i\pi/2)\Big)\Big)\Big(1-\sin(x_{m-j+1}\pi/2)\Big)$$
$$f_m(\mathbf{x}) = x_m + 2m\Big(1-x_1-\frac{\cos(10\pi x_1+\pi/2)}{10\pi}\Big)$$
$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{2(i-1)k/(m-1)+1,\ldots,2ik/(m-1)\Big\}\Big)$$
$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},\Big\{2(k+1),\ldots,2n\Big\}\Big)$$
$$y_{i=1:n} = \text{b\_poly}(y'_i, 0.02)$$
$$y'_{i=1:k} = y''_i$$
$$y'_{i=k+1:n} = \text{b\_flat}(y''_i, 0.8, 0.75, 0.85)$$
$$y''_{i=1:k} = z_i/(2i) \qquad y''_{i=k+1:n} = \text{s\_linear}(z_i/(2i), 0.35)$$

WFG2

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\Big(1-\cos(x_i\pi/2)\Big)$$
$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\Big(1-\cos(x_i\pi/2)\Big)\Big)\Big(1-\sin(x_{m-j+1}\pi/2)\Big)$$
$$f_m(\mathbf{x}) = x_m + 2m\Big(1-x_1\cos^2(5x_1\pi)\Big)$$
$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_{k+l/2}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$y_{i=1:k} = y'_i$$
$$y_{i=k+1:k+l/2} = \text{r\_nonsep}\Big(\Big\{y'_{k+2(i-k)-1}, y'_{k+2(i-k)}\Big\}, 2\Big)$$
$$y'_{i=1:k} = z_i/(2i) \qquad y'_{i=k+1:n} = \text{s\_linear}(z_i/(2i), 0.35)$$

WFG3

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1} x_i$$
$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j} x_i\Big)\Big(1-x_{m-j+1}\Big)$$
$$f_m(\mathbf{x}) = x_m + 2m\Big(1-x_1\Big)$$
$$x_{i=1} = u_i$$
$$x_{i=2:m-1} = x_m\Big(u_i-0.5\Big)+0.5$$
$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_{k+l/2}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$u_i = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$y_{i=1:k} = y'_i$$
$$y_{i=k+1:k+l/2} = \text{r\_nonsep}\Big(\Big\{y'_{k+2(i-k)-1}, y'_{k+2(i-k)}\Big\}, 2\Big)$$
$$y'_{i=1:k} = z_i/(2i) \qquad y'_{i=k+1:n} = \text{s\_linear}(z_i/(2i), 0.35)$$

WFG4

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)$$
$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)$$
$$f_m(\mathbf{x}) = x_m + 2m\cos(x_1\pi/2)$$
$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},\Big\{1,\ldots,1\Big\}\Big)$$
$$y_{i=1:n} = \text{s\_multi}(z_i/(2i), 30, 10, 0.35)$$

WFG5

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)$$

$$f_m(\mathbf{x}) = x_m + 2m\cos(x_1\pi/2)$$

$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$y_{i=1:n} = \text{s\_decept}(z_i/(2i), 0.35, 0.001, 0.05)$$

WFG6

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)$$

$$f_m(\mathbf{x}) = x_m + 2m\cos(x_1\pi/2)$$

$$x_{i=1:m-1} = \text{r\_nonsep}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},k/(m-1)\Big)$$

$$x_m = \text{r\_nonsep}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},l\Big)$$

$$y_{i=1:k} = z_i/(2i)$$

$$y_{i=k+1:n} = \text{s\_linear}(z_i/(2i), 0.35)$$

WFG7

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)$$

$$f_m(\mathbf{x}) = x_m + 2m\cos(x_1\pi/2)$$

$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$y_{i=1:k} = y'_i$$

$$y_{i=k+1:n} = \text{s\_linear}(y'_i, 0.35)$$

$$y'_{i=1:k} = \text{b\_param}\Big(z_i/(2i), \text{r\_sum}\Big(\Big\{z_{i+1}/(2(i+1)),\ldots,z_n/(2n)\Big\},\Big\{1,\ldots,1\Big\}\Big), \frac{0.98}{49.98}, 0.02, 50\Big)$$

$$y'_{i=k+1:n} = z_i/(2i)$$

WFG8

$$f_1(\mathbf{x}) = x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)$$

$$f_{j=2:m-1}(\mathbf{x}) = x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)$$

$$f_m(\mathbf{x}) = x_m + 2m\cos(x_1\pi/2)$$

$$x_{i=1:m-1} = \text{r\_sum}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1))}\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$x_m = \text{r\_sum}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},\Big\{1,\ldots,1\Big\}\Big)$$

$$y_{i=1:k} = y'_i \qquad y_{i=k+1:n} = \text{s\_linear}(y'_i, 0.35)$$

$$y'_{i=k} = z_i/(2i)$$

$$y'_{i=k+1:n} = \text{b\_param}\Big(z_i/(2i), \text{r\_sum}\Big(\Big\{z_1/2,\ldots,z_{i-1}/(2(i-1))\Big\},\Big\{1,\ldots,1\Big\}\Big), \frac{0.98}{49.98}, 0.02, 50\Big)$$

WFG9

$$
\begin{aligned}
f_1(\mathbf{x}) &= x_m + 2\prod_{i=1}^{m-1}\sin(x_i\pi/2)\\
f_{j=2:m-1}(\mathbf{x}) &= x_m + 2j\Big(\prod_{i=1}^{m-j}\sin(x_i\pi/2)\Big)\cos(x_{m-j+1}\pi/2)\\
f_m(\mathbf{x}) &= x_m + 2m\cos(x_1\pi/2)\\
x_{i=1:m-1} &= \text{r\_nonsep}\Big(\Big\{y_{(i-1)k/(m-1)+1},\ldots,y_{ik/(m-1)}\Big\},k/(m-1)\Big)\\
x_m &= \text{r\_nonsep}\Big(\Big\{y_{k+1},\ldots,y_n\Big\},l\Big)\\
y_{i=1:k} &= \text{s\_decept}(y'_i,0.35,0.001,0.05)\\
y_{i=k+1:n} &= \text{s\_multi}(y'_i,30,95,0.35)\\
y'_{i=1:n-1} &= \text{b\_param}\Big(z_i/(2i),\text{r\_sum}\Big(\Big\{z_{i+1}/(2(i+1)),\ldots,z_n/(2n)\Big\},\\
&\qquad \Big\{1,\ldots,1\Big\}\Big),\frac{0.98}{49.98},0.02,50\Big)\\
y'_n &= z_n/(2n)
\end{aligned}
$$

B.5 Inverted DTLZ and WFG Test Problems

Recently, a slight change in the DTLZ and WFG formulations has been proposed by Ishibuchi et al. [73]. The idea is to generate different Pareto front shapes by modifying their general form to:

$$
\begin{aligned}
\text{Minimize} \quad & \big\{-f_1(\mathbf{x}),-f_2(\mathbf{x}),\ldots,-f_m(\mathbf{x})\big\}\\
\text{subject to} \quad & \mathbf{x}\in\mathcal{X}
\end{aligned}
\tag{B.9}
$$

Therefore, the same objective functions and the same constraint conditions are used except that all objectives are multiplied by -1. The generated problems in (B.9) are referred to as DTLZ^{-1} and WFG^{-1}. The effect is that the Pareto front is inverted, though some other properties of the original problems may also change. Figures B.6 and B.7 show the approximations to the Pareto optimal front of these problems. Note, for example, that DTLZ1^{-1}, DTLZ2^{-1}, WFG1^{-1} and WFG4^{-1} have a rotated shape of the Pareto front.

As pointed out by Ishibuchi et al. [73], multi-objective evolutionary algorithms that rely on weight vectors or reference points, such as MOEA/D or NSGA-III, deteriorate their performance when changing only the shape functions of the DTLZ and WFG test problems. This issue evidences the overspecialization of these algorithms on regular and symmetric Pareto fronts.

B.6 Constraint Problems

The following problems consider inequality constraints (2.2), as well as bounds on the decision variables (2.4) according to the MOP definition on page 9.

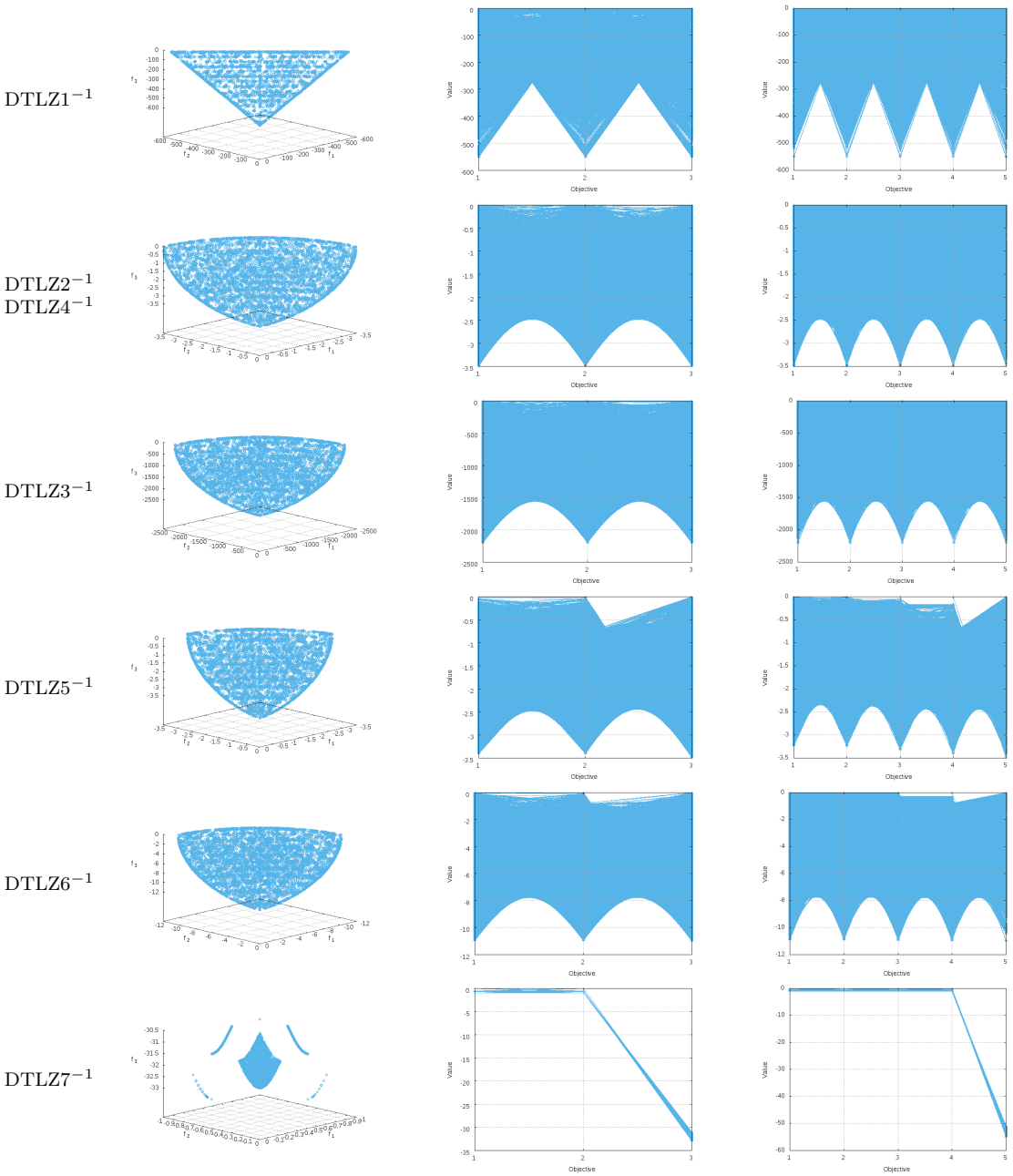


Figure B.6: Approximations to the Pareto optimal front and parallel coordinates of the inverted DTLZ benchmark.

The non-linear constraints of the TNK1 problem, originally proposed by Tanaka et al. [124], make difficult for an optimizer to find the disconnected regions of the Pareto optimal front.

The OSY2 problem, suggested by Osyczka and Kundu [105], contains six constraints that delimit six connected regions in the Pareto optimal front. For optimizers it is very difficult to maintain subpopulations at these regions.

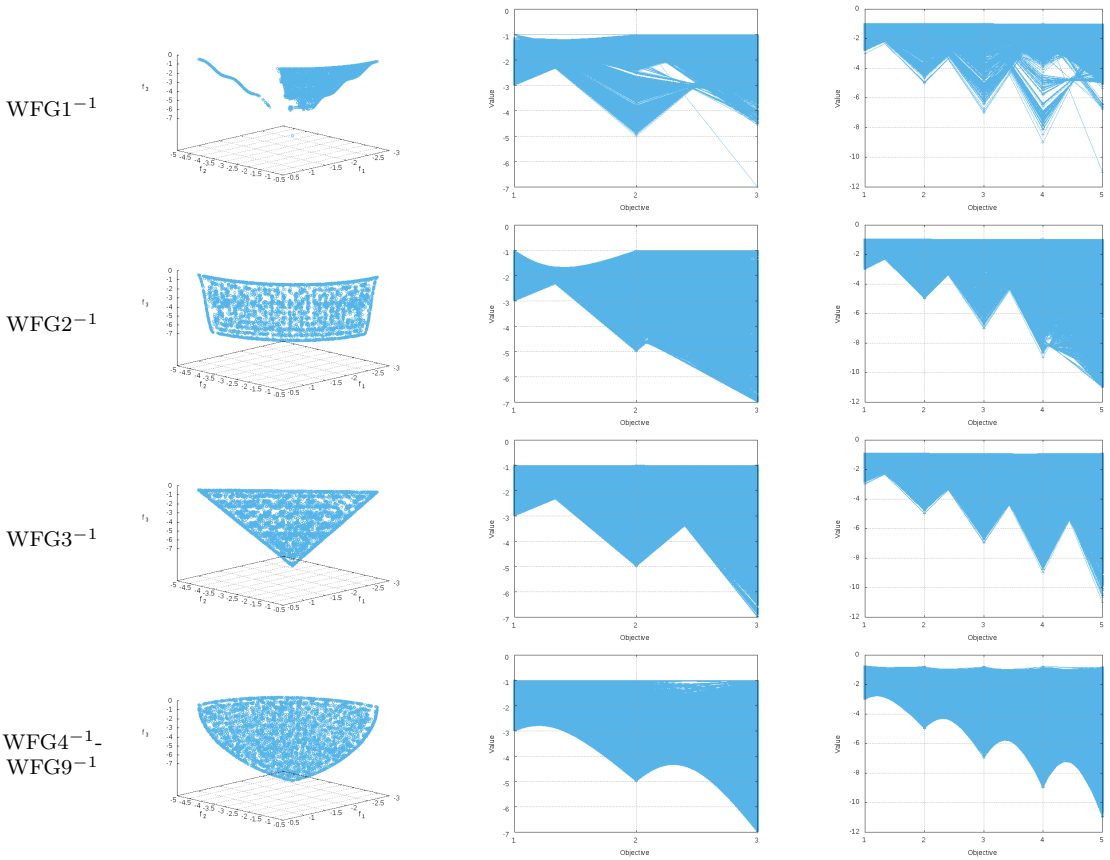


Figure B.7: Approximations to the Pareto optimal front and parallel coordinates of the inverted WFG benchmark.

TNK1

$$
\begin{aligned}
f_1(\mathbf{x}) =& x_1 \\
f_2(\mathbf{x}) =& x_2 \\
g_1(\mathbf{x}) =& x_1^2 + x_2^2 - 1 - 0.1\cos\left(16\arctan\left(\frac{x_1}{x_2}\right)\right) \geq 0 \\
g_2(\mathbf{x}) =& \frac{1}{2} - \left(x_1 - \frac{1}{2}\right)^2 - \left(x_2 - \frac{1}{2}\right)^2 \geq 0 \\
& x_1, x_2 \in (0, \pi]
\end{aligned}
$$

OSY2

$$
\begin{aligned}
f_1(\mathbf{x}) =& -\left(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 \right. \\
& \left. + (x_4-4)^2 + (x_5-1)^2\right) \\
f_2(\mathbf{x}) =& x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \\
& x_1, x_2, x_6 \in [0, 10] \\
& x_3, x_5 \in [1, 5] \\
& x_4 \in [0, 6]
\end{aligned}
\qquad
\begin{aligned}
g_1(\mathbf{x}) =& x_1 + x_2 - 2 \geq 0, \\
g_2(\mathbf{x}) =& 6 - x_1 - x_2 \geq 0, \\
g_3(\mathbf{x}) =& 2 + x_1 - x_2 \geq 0, \\
g_4(\mathbf{x}) =& 2 - x_1 + 3x_2 \geq 0, \\
g_5(\mathbf{x}) =& 4 - (x_3 - 3)^2 - x_4 \geq 0 \\
g_6(\mathbf{x}) =& (x_5 - 3)^2 + x_6 - 4 \geq 0
\end{aligned}
$$

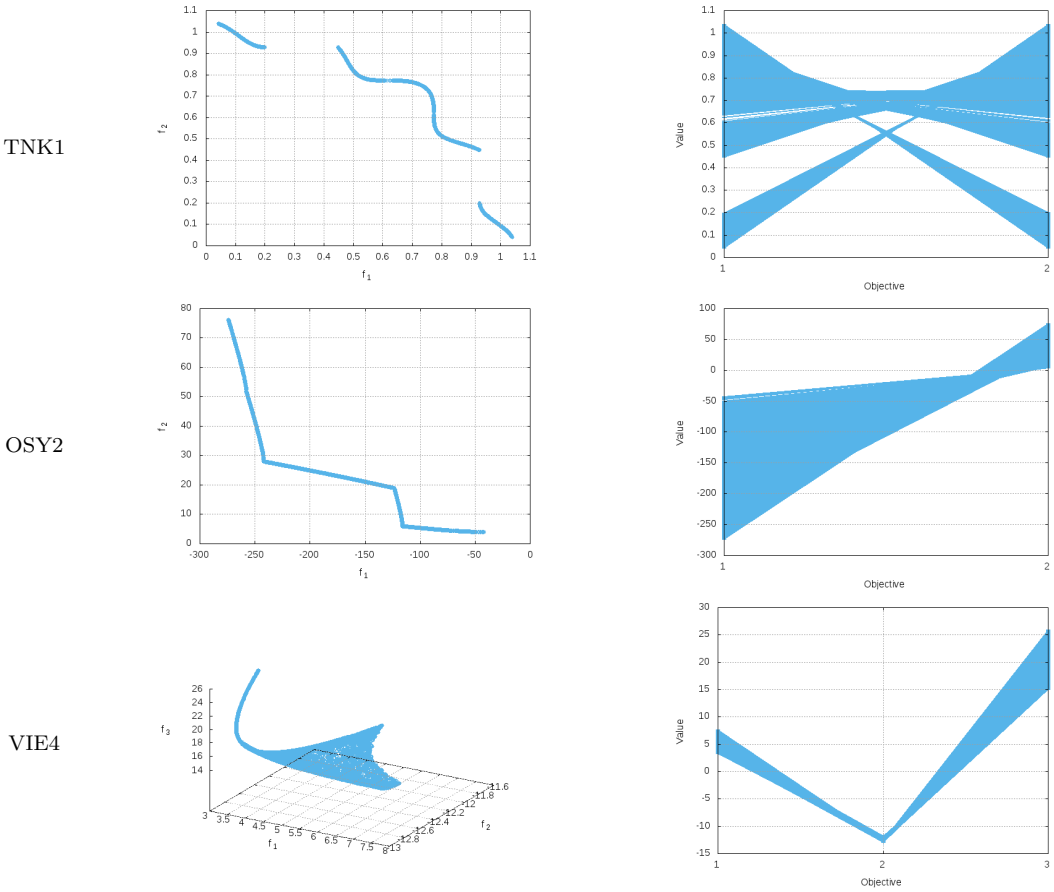


Figure B.8: Pareto optimal fronts and parallel coordinates of the constraint problems.

Finally, VIE4, proposed by Viennet et al. [129, 130], is a three-objective problem with three linear constraints. Optimizers face difficulties in achieving good uniformity of solutions along the curved and asymmetric Pareto front.

VIE4

$$f_1(\mathbf{x}) = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$$

$$f_2(\mathbf{x}) = \frac{(x_1+x_2-3)^2}{175} + \frac{(2x_2-x_1)^2}{17} - 13$$

$$f_3(\mathbf{x}) = \frac{(3x_1-2x_2+4)^2}{8} + \frac{(x_1-x_2+1)^2}{27} + 15$$

$$g_1(\mathbf{x}) = 4 - 4x_1 - x_2 > 0$$

$$g_1(\mathbf{x}) = x_1 + 1 > 0$$

$$g_1(\mathbf{x}) = x_2 - x_1 + 2 > 0$$

$$x_1, x_2 \in [-4, 4]$$

The representations of each MOPs' Pareto optimal front are shown in Figure B.8.

B.7 Summary

When attempting to better understand the strengths and weaknesses of an optimizer, it is important to have a strong understanding of the problem at hand. For this reason, a large set of artificial test problems have been proposed.

In this appendix, we reviewed the most important benchmarks in the field of evolutionary multi-objective optimization: the ZDT benchmark, which includes five continuous problems, the DTLZ test suite, which includes seven unconstrained problems with degenerate and multi-modal Pareto optimal fronts, and the WFG test suite, which includes nine problems with a wide variety of Pareto optimal geometries and fitness landscapes, such as, bias, multi-modality, and non-separability. The DTLZ and WFG benchmarks are scalable with respect to the number of objectives and variables. Moreover, the exact location of the Pareto optimal sets are known. Besides, we have included the inverted versions of the DTLZ and WFG benchmarks, which challenges optimizers to maintain diversity. Finally, we have also considered three constrained problems for two and three objectives.

Bibliography

[1] S. F. Adra and P. J. Fleming, *Diversity Management in Evolutionary Many-Objective Optimization*, IEEE Transactions on Evolutionary Computation, 15 (2011), pp. 183–195.

[2] H. Aguirre, A. Liefooghe, S. Verel, and K. Tanaka, *A Study on Population Size and Selection Lapse in Many-objective Optimization*, in 2013 IEEE Congress on Evolutionary Computation (CEC'2013), Cancún, México, 20-23 June 2013, IEEE Press, pp. 1507–1514. ISBN 978-1-4799-0454-9.

[3] S. G. Akl, *The Design and Analysis of Parallel Algorithms*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[4] J. A. Angelo Jr., *Robotics: A Reference Guide to the New Technology*, Greenwood, 2006.

[5] L. M. Antonio and C. A. Coello Coello, *Use of Cooperative Coevolution for Solving Large Scale Multiobjective Optimization Problems*, in 2013 IEEE Congress on Evolutionary Computation (CEC'2013), Cancún, México, 20-23 June 2013, IEEE Press, pp. 2758–2765. ISBN 978-1-4799-0454-9.

[6] R. S. Assaad and J. Silva-Martínez, *The Recycling Folded Cascode: A General Enhancement of the Folded Cascode Amplifier*, IEEE Journal of Solid-State Circuits, 44 (2009), pp. 2535–2542.

[7] J. Bader and E. Zitzler, *HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization*, Evolutionary Computation, 19 (Spring, 2011), pp. 45–76.

[8] M. P. Basgalupp, R. C. Barros, and V. Podgorelec, *Evolving Decision-tree Induction Algorithms with a Multi-objective Hyper-heuristic*, in Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, New York, NY, USA, 2015, ACM, pp. 110–117.

[9] N. Beume, B. Naujoks, and M. Emmerich, *SMS-EMOA: Multiobjective Selection based on Dominated Hypervolume*, European Journal of Operational Research, 181 (2007), pp. 1653–1669.

[10] L. Bradstreet, L. While, and L. Barone, *A Fast Incremental Hypervolume Algorithm*, IEEE Transactions on Evolutionary Computation, 12 (2008), pp. 714–723.

[11] L. Bradstreet, L. While, and L. Barone, *A New Way of Calculating Exact Exclusive Hypervolumes*, Tech. Rep. UWA-CSSE-09-002, The University of Western Australia, School of Computer Science and Software Engineering, 2009.

[12] J. Bresenham, *A Linear Algorithm for Incremental Digital Display of Circular Arcs*, Communications of the ACM, 20 (1977), pp. 100–106.

[13] K. Bringmann and T. Friedrich, *Don't be Greedy when Calculating Hypervolume Contributions*, in FOGA '09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms, Orlando, Florida, USA, January 2009, ACM, pp. 103–112.

[14] D. Brockhoff, T. Wagner, and H. Trautmann, *On the Properties of the R2 Indicator*, in 2012 Genetic and Evolutionary Computation Conference (GECCO'2012), Philadelphia, USA, July 2012, ACM Press, pp. 465–472. ISBN: 978-1-4503-1177-9.

[15] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, *Hyper-heuristics: A Survey of the State of the Art*, Journal of the Operational Research Society, 64 (2013), pp. 1695–1724.

[16] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, *A Classification of Hyper-heuristic Approaches*, Springer US, Boston, MA, 2010, pp. 449–468.

[17] E. K. Burke, J. D. L. Silva, and E. Soubeiga, *Multi-Objective Hyper-Heuristic Approaches for Space Allocation and Timetabling*, Springer US, Boston, MA, 2005, pp. 129–158.

[18] F. Y. Cheng and X. S. Li, *Generalized Center Method for Multiobjective Engineering Optimization*, Engineering Optimization, 31 (1999), pp. 641–661.

[19] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, *Test Problems for Large-Scale Multiobjective and Many-Objective Optimization*, IEEE Transactions on Cybernetics, PP (2016), pp. 1–14.

[20] C. A. Coello Coello and G. B. Lamont, eds., *Applications of Multi-Objective Evolutionary Algorithms*, World Scientific, Singapore, 2004. ISBN 981-256-106-4.

[21] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, New York, second ed., September 2007. ISBN 978-0-387-33254-3.

[22] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, Addison-Wesley Publishing Company, USA, 5th ed., 2011.

[23] W. Cox and L. While, *Improving the IWFG Algorithm for Calculating Incremental Hypervolume*, in 2016 IEEE Congress on Evolutionary Computation (CEC), July 2016, pp. 3969–3976.

[24] I. Das and J. E. Dennis, *A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto set Generation in Multicriteria Optimization Problems*, Structural optimization, 14 (1997), pp. 63–69.

[25] K. Deb, *Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems*, Evolutionary Computation, 7 (1999), pp. 205–230.

[26] ——, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[27] K. Deb and R. B. Agrawal, *Simulated Binary Crossover for Continuous Search Space*, Complex Systems, 9 (1995), pp. 115–148.

[28] K. Deb and H. Jain, *An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints*, IEEE Transactions on Evolutionary Computation, 18 (2014), pp. 577–601.

[29] K. Deb, M. Mohan, and S. Mishra, *Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions*, Evolutionary Computation, 13 (2005), pp. 501–525.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 182–197.

[31] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable Test Problems for Evolutionary Multiobjective Optimization*, in Evolutionary Multiobjective Optimization, A. Abraham, L. Jain, and R. Goldberg, eds., Advanced Information and Knowledge Processing, Springer London, 2005, pp. 105–145.

[32] J. J. Durillo and A. J. Nebro, *jMetal: A Java Framework for Multi-Objective Optimization*, Advances in Engineering Software, 42 (2011), pp. 760–771.

[33] G. Eichfelder, *An Adaptive Scalarization Method in Multiobjective Optimization*, SIAM Journal on Optimization, 19 (2009), pp. 1694–1718.

[34] M. Emmerich, N. Beume, and B. Naujoks, *An EMO Algorithm Using the Hypervolume Measure as Selection Criterion*, in Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005, C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, eds., Guanajuato, México, March 2005, Springer. Lecture Notes in Computer Science Vol. 3410, pp. 62–76.

[35] M. Fleischer, *The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics*, in Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, eds., Faro, Portugal, April 2003, Springer. Lecture Notes in Computer Science. Volume 2632, pp. 519–533.

[36] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice (2Nd Ed.)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[37] C. M. Fonseca and P. J. Fleming, *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, in Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest, ed., San Mateo, California, 1993, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, pp. 416–423.

[38] F. N. Fritsch and R. E. Carlson, *Monotone Piecewise Cubic Interpolation*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 238–246.

[39] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, Springer Publishing Company, Incorporated, 2nd ed., 2010.

[40] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, *Generalized Decomposition*, in Evolutionary Multi-Criterion Optimization, 7th International Conference, EMO 2013, R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, eds., Springer. Lecture Notes in Computer Science Vol. 7811, Sheffield, UK, March 19-22 2013, pp. 428–442.

[41] T. Glasmachers, *Optimized Approximation Sets for Low-Dimensional Benchmark Pareto Fronts*, in Parallel Problem Solving from Nature - PPSN XIII, 13th International Conference, T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, eds., Springer. Lecture Notes in Computer Science Vol. 8672, Ljubljana, Slovenia, September 13-17 2014, pp. 569–578.

[42] D. E. Goldberg and J. Richardson, *Genetic Algorithms with Sharing for Multimodal Function Optimization*, in Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, Hillsdale, NJ, USA, 1987, L. Erlbaum Associates Inc., pp. 41–49.

[43] J. C. Gomez and H. Terashima-Marín, *Approximating Multi-Objective Hyper-Heuristics for Solving 2D Irregular Cutting Stock Problems*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 349–360.

[44] R. A. Gonçalves, C. P. Almeida, S. M. Venske, J. N. Kuk, L. M. Pavelski, and M. R. Delgado, *A Hyper-Heuristic for the Environmental/Economic Dispatch Optimization Problem*, in Proceedings of the 2015 Brazilian Conference on Intelligent Systems (BRACIS), BRACIS '15, Washington, DC, USA, 2015, IEEE Computer Society, pp. 7–12.

[45] R. A. Gonçalves, J. N. Kuk, C. P. Almeida, and S. M. Venske, *Decomposition Based Multiobjective Hyper Heuristic with Differential Evolution*, Springer International Publishing, Cham, 2015, pp. 129–138.

[46] ——, *MOEA/D-HH: A Hyper-Heuristic for Multi-objective Problems*, in Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, A. Gaspar-Cunha, C. H. Antunes, and C. Coello Coello, eds., Springer. Lecture Notes in Computer Science Vol. 9018, Guimarães, Portugal, March 29 - April 1 2015, pp. 94–108.

[47] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[48] L. Gu, R.-J. Yang, C. Tho, M. Makowskit, O. Faruquet, and Y. Li, *Optimisation and Robustness for Crashworthiness of Side Impact*, 26 (2001).

[49] I. Guerra-Gómez, E. Tlelo-Cuautle, and L. G. De La Fraga, *Richardson Extrapolation-based Sensitivity Analysis in the Multi-objective Optimization of Analog Circuits*, Applied Mathematics and Computation, 222 (2013), pp. 167–176.

[50] G. Guizzo, G. M. Fritsche, S. R. Vergilio, and A. T. R. Pozo, *A Hyper-Heuristic for the Multi-Objective Integration and Test Order Problem*, in Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, New York, NY, USA, 2015, ACM, pp. 1343–1350.

[51] G. Guizzo, S. R. Vergilio, A. T. Pozo, and G. M. Fritsche, *A Multi-Objective and Evolutionary Hyper-Heuristic Applied to the Integration and Test Order Problem*, Applied Soft Computing, 56 (2017), pp. 331–344.

[52] H. J. H. and K. Deb, *An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach*, IEEE Transactions on Evolutionary Computation, 18 (2014), pp. 602–622.

[53] H. Haario, E. Saksman, and J. Tamminen, *An Adaptive Metropolis Algorithm*, Bernoulli, 7 (2001), pp. 223–242.

[54] D. Hardin and E. Saff, *Discretizing Manifolds via Minimum Energy Points*, Notices of the American Mathematical Society, 51 (2004), pp. 1186–1194.

[55] J. Heinrich and D. Weiskopf, *State of the Art of Parallel Coordinates*, in Eurographics 2013 - State of the Art Reports, M. Sbert and L. Szirmay-Kalos, eds., The Eurographics Association, 2013.

[56] R. Hernández Gómez, C. A. C. Coello, and E. Alba, *A Parallel Version of SMS-EMOA for Many-Objective Optimization Problems*, in Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference, J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, eds., Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, September 17-21 2016, pp. 568–577. ISBN 978-3-319-45822-9.

[57] R. Hernández Gómez and C. A. Coello Coello, *MOMBI: A New Metaheuristic for Many-Objective Optimization Based on the R2 Indicator*, in 2013 IEEE Congress on Evolutionary Computation (CEC'2013), Cancún, México, 20-23 June 2013, IEEE Press, pp. 2488–2495. ISBN 978-1-4799-0454-9.

[58] ——, *Improved Metaheuristic Based on the R2 Indicator for Many-Objective Optimization*, in 2015 Genetic and Evolutionary Computation Conference (GECCO 2015), Madrid, Spain, July 11-15 2015, ACM Press, pp. 679–686. ISBN 978-1-4503-3472-3.

[59] R. Hernández Gómez and C. A. Coello Coello, *Parallel SMS-EMOA for Many-Objective Optimization Problems*, in Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO '16 Companion, New York, NY, USA, 2016, ACM, pp. 1011–1014.

[60] R. Hernández Gómez, C. A. Coello Coello, and E. Alba Torres, *A Multi-Objective Evolutionary Algorithm based on Parallel Coordinates*, in 2016 Genetic and Evolutionary Computation Conference (GECCO'2016), Denver, Colorado, USA, 20-24 July 2016, ACM Press, pp. 565–572. ISBN 978-1-4503-4206-3.

[61] J. Horn, N. Nafpliotis, and D. E. Goldberg, *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*, in Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol. 1, Piscataway, New Jersey, June 1994, IEEE Service Center, pp. 82–87.

[62] W. Hu and G. Yen, *Density Estimation for Selecting Leaders and Mantaining Archive in MOPSO*, in IEEE Congress on Evolutionary Computation (CEC), 2013, June 2013, pp. 181–188.

[63] W. Hu and G. G. Yen, *Adaptive Multiobjective Particle Swarm Optimization Based on Parallel Cell Coordinate System*, IEEE Transactions on Evolutionary Computation, 19 (2015), pp. 1–18.

[64] S. Huband, P. Hingston, L. Barone, and L. While, *A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit*, IEEE Transactions on Evolutionary Computation, 10 (2006), pp. 477–506.

[65] E. J. Hughes, *Multiple Single Objective Pareto Sampling*, in Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003), vol. 4, Canberra, Australia, December 2003, IEEE Press, pp. 2678–2684.

[66] C. Igel, N. Hansen, and S. Roth, *Covariance Matrix Adaptation for Multiobjective Optimization*, Evolutionary Computation, 15 (2007), pp. 1–28.

[67] A. Inselberg, *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*, Springer-Verlag New York, Secaucus, NJ, USA, 2009.

[68] H. Ishibuchi, N. Akedo, and Y. Nojima, *Behavior of Multiobjective Evolutionary Algorithms on Many-Objective Knapsack Problems*, IEEE Transactions on Evolutionary Computation, 19 (2015), pp. 264–283.

[69] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, *Modified Distance Calculation in Generational Distance and Inverted Generational Distance*, in Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, A. Gaspar-Cunha, C. H. Antunes, and C. Coello Coello, eds., Springer. Lecture Notes in Computer Science Vol. 9019, Guimarães, Portugal, March 29 - April 1 2015, pp. 110–125.

[70] H. Ishibuchi and T. 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 (1998), pp. 392–403.

[71] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, *Adaptation of Scalarizing Funtions in MOEA/D: An Adaptive Scalarizing Funtion-Based Multiobjective Evolutionary Algorithm*, in Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, eds., Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France, April 2009, pp. 438–452.

[72] ——, *Simultaneous Use of Different Scalarizing Functions in MOEA/D*, in Proceedings of the 12th annual conference on Genetic and Evolutionary Computation (GECCO'2010), Portland, Oregon, USA, July 7–11 2010, ACM Press, pp. 519–526. ISBN 978-1-4503-0072-8.

[73] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, *Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes*, IEEE Transactions on Evolutionary Computation, 21 (2017), pp. 169–190.

[74] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, *Evolutionary Many-Objective Optimization: A Short Review*, in 2008 Congress on Evolutionary Computation (CEC'2008), Hong Kong, June 2008, IEEE Service Center, pp. 2424–2431.

[75] M. A. Jan and Q. Zhang, *MOEA/D for Constrained Multiobjective Optimization: Some Preliminary Experimental Results*, 2010 UK Workshop on Computational Intelligence (UKCI), (2010), pp. 1–6.

[76] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[77] J.-W. Klinkenberg, M. T. Emmerich, A. H. Deutz, O. M. Shir, and T. Bäck, *A Reduced-Cost SMS-EMOA Using Kriging, Self-Adaptation, and Parallelization*, in Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, eds., Springer, Lecture Notes in Economics and Mathematical Systems Vol. 634, Heidelberg, Germany, 2010, pp. 301–311.

[78] J. D. Knowles and D. W. Corne, *Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy*, Evolutionary Computation, 8 (2000), pp. 149–172.

[79] J. D. Knowles, D. W. Corne, and K. Deb, eds., *Multiobjective Problem Solving from Nature. From Concepts to Applications*, Springer, Berlin, 2008. ISBN 978-3-540-72963-1.

[80] J. D. Knowles, D. W. Corne, and M. Fleischer, *Bounded Archiving using the Lebesgue Measure*, in Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003), vol. 4, Canberra, Australia, December 2003, IEEE Press, pp. 2490–2497.

[81] O. Kramer, *A Review of Constraint-handling Techniques for Evolution Strategies*, Applied Computational Intelligence and Soft Computing, 2010 (2010), pp. 3:1–3:19.

[82] ——, *Genetic Algorithm Essentials*, Springer Publishing Company, Incorporated, 1st ed., 2017.

[83] R. Kumar, B. K. Bal, and P. I. Rockett, *Multiobjective Genetic Programming Approach to Evolving Heuristics for the Bounded Diameter Minimum*

Spanning Tree Problem: MOGP for BDMST, in Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, New York, NY, USA, 2009, ACM, pp. 309–316.

[84] R. Kumar, A. H. Joshi, K. K. Banka, and P. I. Rockett, *Evolution of Hyperheuristics for the Biobjective 0/1 Knapsack Problem by Multiobjective Genetic Programming*, in Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08, New York, NY, USA, 2008, ACM, pp. 1227–1234.

[85] A. C. Kumari and K. Srinivas, *Hyper-heuristic Approach for Multi-objective Software Module Clustering*, Journal of Systems and Software, 117 (2016), pp. 384–401.

[86] A. C. Kumari, K. Srinivas, and M. P. Gupta, *Software Module Clustering using a Hyper-Heuristic based Multi-Objective Genetic Algorithm*, in 2013 3rd IEEE International Advance Computing Conference (IACC), Feb 2013, pp. 813–818.

[87] C. León, G. Miranda, E. Segredo, and C. Segura, *Parallel Hypervolume-Guided Hyperheuristic for Adapting the Multi-objective Evolutionary Island Model*, in Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), N. Krasnogor, M. Melián-Batista, J. Pérez, J. Moreno-Vega, and D. Pelta, eds., vol. 236 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2009, pp. 261–272.

[88] C. León, G. Miranda, and C. Segura, *Parallel Hyperheuristic: A Self-Adaptive Island-Based Model for Multi-Objective Optimization*, in 2008 Genetic and Evolutionary Computation Conference (GECCO'2008), Atlanta, USA, July 2008, ACM Press, pp. 757–758. ISBN 978-1-60558-131-6.

[89] C. León, G. Miranda, and C. Segura, *A Parallel Plugin-Based Framework for Multi-objective Optimization*, in International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), J. Corchado, S. Rodríguez, J. Llinas, and J. Molina, eds., vol. 50 of Advances in Soft Computing, Springer Berlin Heidelberg, 2009, pp. 142–151.

[90] ——, *Hyperheuristics for a Dynamic-Mapped Multi-Objective Island-Based Model*, in Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, S. Omatu, M. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, and J. Corchado, eds., vol. 5518 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 41–49.

[91] C. León, G. Miranda, and C. Segura, *METCO: A Parallel Plugin-Based Framework For Multi-Objective Optimization*, International Journal on Artificial Intelligence Tools, 18 (2009), pp. 569–588.

[92] B. Li, J. Li, K. Tang, and X. Yao, *Many-Objective Evolutionary Algorithms: A Survey*, ACM Computing Surveys, 48 (2015).

[93] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, and E.-G. Talbi, *ParadisEO-MOEO: A Software Framework for Evolutionary Multi-Objective Optimization*, in Advances in Multi-Objective Nature Inspired Computing, C. A. Coello Coello, C. Dhaenens, and L. Jourdan, eds., Springer, Studies in Computational Intelligence, Vol. 272, Berlin, Germany, 2010, ch. 5, pp. 87–117. ISBN 978-3-642-11217-1.

[94] F. Luna and E. Alba, *Parallel Multiobjective Evolutionary Algorithms*, in Springer Handbook of Computational Intelligence, J. Kacprzyk and W. Pedrycz, eds., Springer Berlin Heidelberg, 2015, pp. 1017–1031.

[95] G. Luque, E. Alba, and B. Dorronsoro, *Analyzing Parallel Cellular Genetic Algorithms*, Wiley, New Jersey, USA, may 2009, pp. 49–62.

[96] E. Manoatl Lopez, L. Miguel Antonio, and Carlos A. Coello Coello, *A GPU-Based Algorithm for a Faster Hypervolume Contribution Computation*, in Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, A. Gaspar-Cunha, C. H. Antunes, and C. Coello Coello, eds., Springer. Lecture Notes in Computer Science Vol. 9019, Guimarães, Portugal, March 29 - April 1 2015, pp. 80–94.

[97] T. Mariani, G. Guizzo, S. R. Vergilio, and A. T. Pozo, *Grammatical Evolution for the Multi-Objective Integration and Test Order Problem*, in Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, New York, NY, USA, 2016, ACM, pp. 1069–1076.

[98] K. McClymont and E. C. Keedwell, *Markov Chain Hyper-heuristic (MCHH): An Online Selective Hyper-heuristic for Multi-objective Continuous Problems*, in Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, New York, NY, USA, 2011, ACM, pp. 2003–2010.

[99] A. Messac, C. Puemi-Sukam, and E. Melachrinoudis, *Aggregate Objective Functions and Pareto Frontiers: Required Relationships and Practical Implications*, Optimization and Engineering, 1 (2000), pp. 171–188.

[100] E. Mezura-Montes and C. A. Coello Coello, *Constrained Optimization via Multiobjective Evolutionary Algorithms*, in Multi-Objective Problem Solving from Nature: From Concepts to Applications, J. Knowles, D. Corne, and K. Deb, eds., Springer, Berlin, 2008, pp. 53–75. ISBN 978-3-540-72963-1.

[101] Z. Michalewicz, *A Survey of Constraint Handling Techniques in Evolutionary Computation Methods*, in Proceedings of the 4th Annual Conference on Evolutionary Programming, MIT Press, 1995, pp. 135–155.

[102] K. Miettinen, *Survey of Methods to Visualize Alternatives in Multiple Criteria Decision Making Problems*, OR Spectrum, 36 (2014), pp. 3–37.

[103] K. Musselman and J. Talavage, *A Tradeoff Cut Approach to Multiple Objective Optimization*, Operations Research, 28 (1980), pp. 1424–1435.

[104] I. H. Osman and J. P. Kelly, *Meta-Heuristics: An Overview*, Springer US, Boston, MA, 1996, pp. 1–21.

[105] A. Osyczka and S. Kundu, *A New Method to Solve Generalized Multicriteria Optimization Problems using the Simple Genetic Algorithm*, Structural Optimization, 10 (1995), pp. 94–99.

[106] M. Pescador-Rojas, R. Hernández Gómez, E. Montero, N. Rojas-Morales, M.-C. Riff, and C. A. Coello Coello, *An Overview of Weighted and Unconstrained Scalarizing Functions*, in Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings, H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, and C. Grimme, eds., Springer International Publishing, Cham, 2017, pp. 499–513.

[107] C. Qian, K. Tang, and Z.-H. Zhou, *Selection Hyper-heuristics Can Provably Be Helpful in Evolutionary Multi-objective Optimization*, Springer International Publishing, Cham, 2016, pp. 835–846.

[108] K. Rahman, C. Maringanti, M. Beniston, F. Widmer, K. Abbaspour, and A. Lehmann, *Streamflow Modeling in a Highly Managed Mountainous Glacier Watershed Using SWAT: The Upper Rhone River Watershed Case in Switzerland*, Water Resources Management, 27 (2013), pp. 323–339.

[109] T. Ray, T. Kang, and S. K. Chye, *Multiobjective Design Optimization by an Evolutionary Algorithm*, Engineering Optimization, 33 (2001), pp. 399–424.

[110] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution*, Frommann-Holzboog, 1973.

[111] G. Rudolph, *Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets*, in Evolutionary Programming VII, Proceedings of the 7th Annual Conference on Evolutionary Programming, V. Porto, N. Saravanan, D. Waagen, and A. Eiben, eds., Berlin, 1998, Springer, pp. 345–353.

[112] L. V. Santana-Quintero, A. Arias Montaño, and C. A. Coello Coello, *A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization*, in Computational Intelligence in Expensive Optimization Problems, Y. Tenne and C.-K. Goh, eds., Springer, Berlin, Germany, 2010, pp. 29–59. ISBN 978-3-642-10700-9.

[113] A. Santiago, H. J. F. Huacuja, B. Dorronsoro, J. E. Pecero, C. G. Santillan, J. J. G. Barbosa, and J. C. S. Monterrubio, *A Survey of Decomposition Methods for Multi-objective Optimization*, in Recent Advances on Hybrid Approaches for Designing Intelligent Systems, O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk, eds., Springer, 2014, pp. 453–465. ISBN 978-3-319-05170-3.

[114] H. Scheffé, *Experiments with Mixtures*, Journal of the Royal Statistical Society. Series B (Statistical Methodology), 20 (1958), pp. 344–360.

[115] F. Schlottmann and D. Seese, *Financial Applications of Multi-Objective Evolutionary Algorithms: Recent Developments and Future Research Directions*, in Applications of Multi-Objective Evolutionary Algorithms, C. A. Coello Coello and G. B. Lamont, eds., World Scientific, Singapore, 2004, pp. 627–652.

[116] O. Schütze, X. Esquivel, A. Lara, and C. A. Coello Coello, *Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization*, IEEE Transactions on Evolutionary Computation, 16 (2012), pp. 504–522.

[117] C. Segura, A. Cervantes, A. J. Nebro, M. D. Jaraíz-Simón, E. Segredo, S. García, F. Luna, J. A. Gómez-Pulido, G. Miranda, C. Luque, E. Alba, M. Ángel Vega-Rodríguez, C. León, and I. M. Galván, *Optimizing the DFCN Broadcast Protocol with a Parallel Cooperative Strategy of Multi-Objective Evolutionary Algorithms*, in Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, eds., Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France, April 2009, pp. 305–319.

[118] N. Srinivas and K. Deb, *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*, Evolutionary Computation, 2 (1994), pp. 221–248.

[119] T. Stewart, O. Bandte, H. Braun, N. Chakraborti, M. Ehrgott, M. Göbelt, Y. Jin, H. Nakayama, S. Poles, and D. Di Stefano, *Real-World Applications of Multiobjective Optimization*, in Multiobjective Optimization. Interactive and Evolutionary Approaches, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, eds., Springer. Lecture Notes in Computer Science Vol. 5252, Berlin, Germany, 2008, pp. 285–327.

[120] R. Storn and K. Price, *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces*, Journal of Global Optimization, 11 (1997), pp. 341–359.

[121] E.-G. Talbi, *Metaheuristics: From Design to Implementation*, Wiley Publishing, 2009.

[122] E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. Coello Coello, *Parallel Approaches for Multi-objective Optimization*, in Multiobjective Optimization. Interactive and Evolutionary Approaches, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, eds., Springer. Lecture Notes in Computer Science Vol. 5252, Berlin, Germany, 2008, pp. 349–372.

[123] K. Tan, E. Khor, and T. Lee, *Multiobjective Evolutionary Algorithms and Applications*, Springer-Verlag, London, 2005. ISBN 1-85233-836-9.

[124] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, *GA-based Decision Support System for Multicriteria Optimization*, in 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, vol. 2, Oct 1995, pp. 1556–1561 vol.2.

[125] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, *PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization*, IEEE Computational Intelligence Magazine, 12 (2017), pp. 73–87.

[126] T. Tušar and B. Filipič, *Visualization of Pareto Front Approximations in Evolutionary Multiobjective Optimization: A Critical Review and the Prosection Method*, IEEE Transactions on Evolutionary Computation, 19 (2015), pp. 225–245.

[127] D. A. Van Veldhuizen, J. B. Zydallis, and G. B. Lamont, *Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms*, IEEE Transactions on Evolutionary Computation, 7 (2003), pp. 144–173.

[128] J. A. Vázquez-Rodríguez and S. Petrovic, *A New Dispatching Rule based Genetic Algorithm for the Multi-Objective Job Shop Problem*, Journal of Heuristics, 16 (2010), pp. 771–793.

[129] R. Viennet, C. Fontiex, and I. Marc, *New Multicriteria Optimization Method Based on the Use of a Diploid Genetic Algorithm: Example of an Industrial Problem*, in Proceedings of Artificial Evolution (European Conference, selected papers), J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, eds., Brest, France, September 1995, Springer-Verlag. Lecture Notes in Computer Science Vol. 1063, pp. 120–127.

[130] ——, *Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set*, International Journal of Systems Science, 27 (1996), pp. 255–260.

[131] C. von Lücken, B. Baran, and C. Brizuela, *A survey on Multi-Objective Evolutionary Algorithms for Many-Objective Problems*, Computational Optimization and Applications, 58 (2014), pp. 707–756.

[132] J. A. Vrugt, M. P. Clark, C. G. H. Diks, Q. Duan, and B. A. Robinson, *Multi-Objective Calibration of Forecast Ensembles using Bayesian Model Averaging*, Geophysical Research Letters, 33 (2006), pp. 1–6. L19817.

[133] J. A. Vrugt and B. A. Robinson, *Improved Evolutionary Optimization from Genetically Adaptive Multimethod Search*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 708–711.

[134] J. A. Vrugt, P. H. Stauffer, T. Wöhling, B. A. Robinson, and V. V. Vesselinov, *Inverse Modeling of Subsurface Flow and Transport Properties: A Review with New Developments*, Vadose Zone, 7 (2008), pp. 843–864.

[135] T. Wagner, N. Beume, and B. Naujoks, *Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization*, in Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, eds., Matshushima, Japan, March 2007, Springer. Lecture Notes in Computer Science Vol. 4403, pp. 742–756.

[136] D. J. Walker and E. Keedwell, *Multi-objective Optimisation with a Sequence-based Selection Hyper-heuristic*, in Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO '16 Companion, New York, NY, USA, 2016, ACM, pp. 81–82.

[137] D. J. Walker and E. Keedwell, *Towards Many-Objective Optimisation with Hyper-Heuristics: Identifying Good Heuristics with Indicators*, in Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference, J. Handl, E. Hart, P. R. Lewis, M. L'opez-Ib'añez, G. Ochoa, and B. Paechter, eds., Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, September 17-21 2016, pp. 493–502. ISBN 978-3-319-45822-9.

[138] L. Wang, R. Ranjan, J. Chen, and B. Benatallah, *Cloud Computing: Methodology, Systems, and Applications*, CRC Press, Inc., Boca Raton, FL, USA, 1st ed., 2011.

[139] R. Wang, Q. Zhang, and T. Zhang, *Decomposition-Based Algorithms Using Pareto Adaptive Scalarizing Methods*, IEEE Transactions on Evolutionary Computation, 20 (2016), pp. 821–837.

[140] Y. Wang and B. Li, *Multi-Strategy Ensemble Evolutionary Algorithm for Dynamic Multi-Objective Optimization*, Memetic Computing, 2 (2010), pp. 3–24.

[141] S. Wessing, G. Rudolph, and D. A. Menges, *Comparing Asynchronous and Synchronous Parallelization of the SMS-EMOA*, in Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference, J. Handl,

E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, eds., Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, September 17-21 2016, pp. 558–567. ISBN 978-3-319-45822-9.

[142] L. While and L. Bradstreet, *Applying the WFG Algorithm to Calculate Incremental Hypervolumes*, in 2012 IEEE Congress on Evolutionary Computation (CEC'2012), Brisbane, Australia, June 10-15 2012, IEEE Press, pp. 489–496.

[143] L. While, L. Bradstreet, and L. Barone, *A Fast Way of Calculating Exact Hypervolumes*, IEEE Transactions on Evolutionary Computation, 16 (2012), pp. 86–95.

[144] T. Wöhling, J. Vrugt, and G. Barkle, *Comparison of Three Multiobjective Optimization Algorithms for Inverse Modeling of Vadose Zone Hydraulic Properties*, 72 (2008), pp. 305–319.

[145] R. Xu and D. Wunsch, II, *Survey of Clustering Algorithms*, IEEE Transactions on Neural Networks, 16 (2005), pp. 645–678.

[146] Y. yan Tan, Y. chang Jiao, H. Li, and X. kuan Wang, *MOEA/D plus Uniform Eesign: A New Version of MOEA/D for Optimization Problems with Many Objectives*, Computers & Operations Research, 40 (2013), pp. 1648–1660.

[147] S. Yang, S. Jiang, and Y. Jiang, *Improving the Multiobjective Evolutionary Algorithm based on Decomposition with New Penalty Schemes*, Soft Computing, (2016), pp. 1–15.

[148] Q. Zhang and H. Li, *MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition*, IEEE Transactions on Evolutionary Computation, 11 (2007), pp. 712–731.

[149] Q. Zhang, W. Liu, and H. Li, *The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances*, in 2009 IEEE Congress on Evolutionary Computation (CEC'2009), Trondheim, Norway, May 2009, IEEE Press, pp. 203–208.

[150] X. Zhang, P. Beeson, R. Link, D. Manowitz, R. C. Izaurralde, A. Sadeghi, A. M. Thomson, R. Sahajpal, R. Srinivasan, and J. G. Arnold, *Efficient Multi-Objective Calibration of a Computationally Intensive Hydrologic Model with Parallel Computing Software in Python*, Environmental Modelling & Software, 46 (2013), pp. 208 – 218.

[151] X. Zhang, R. Srinivasan, and M. V. Liew, *On the use of Multi-Algorithm, Genetically Adaptive Multi-Objective Method for Multi-Site Calibration of the SWAT Model*, Hydrological Processes, 24 (2010), pp. 955–969.

[152] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[153] E. Zitzler, K. Deb, and L. Thiele, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation, 8 (2000), pp. 173–195.

[154] E. Zitzler and S. Künzli, *Indicator-based Selection in Multiobjective Search*, in Parallel Problem Solving from Nature - PPSN VIII, X. Y. et al., ed., Birmingham, UK, September 2004, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242, pp. 832–842.

[155] E. Zitzler, M. Laumanns, and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[156] E. Zitzler and L. Thiele, *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*, IEEE Transactions on Evolutionary Computation, 3 (1999), pp. 257–271.

[157] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, *Performance Assessment of Multiobjective Optimizers: An Analysis and Review*, IEEE Transactions on Evolutionary Computation, 7 (2003), pp. 117–132.