CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL
DEPARTAMENTO DE COMPUTACIÓN

Using Gradient Based Information to Build Hybrid Multi-objective Evolutionary Algorithms

A dissertation submitted by
Adriana Lara López

For the degree of
Doctor of Computer Science

Supervisors
Dr. Carlos A. Coello Coello
Dr. Oliver Schütze

Mexico City, Mexico. May, 2012.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO
NACIONAL
UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Incorporación de información de gradientes en el diseño de algoritmos evolutivos multi-objetivo híbridos

Tesis que presenta
Adriana Lara López

Para Obtener el Grado de
Doctora en Ciencias

en

Computación

Directores de la Tesis
Dr. Carlos Artemio Coello Coello
Dr. Oliver Steffen Schütze

México, Distrito Federal. Mayo, 2012.

Abstract

Over the last decades evolutionary algorithms have become very popular to solve multi-objective optimization problems (MOPs). Several multi-objective evolutionary algorithms (MOEAs) have been developed to solve MOPs with successfull results. A feature of these algorithms is that they do not exploit concrete information, about continuity or differentiability of the objective functions of the problems—which is considered as information of the problem domain. One question that arises when seeking for more efficient MOEAs, is about the effectiveness of including this mathematical information during the MOEA execution. In particular, we are interested in exploiting the gradient information of the objective functions during the evolutionary search. In this thesis, the inclusion of gradient-based local searchers into MOEAs is presented. An in depth study of the gradient-based search directions is included, as well as the proposal of diverse types of hybridization. This coupling has two aims, one is made in order to improve the performance of these stochastic algorithms, and the second one is to efficiently refine their solution sets. Hybrid gradient-based MOEAs are built and tested, in this work, over widely used benchmark MOPs. The numerical results are analyzed and discussed; also, conclusions and extensions for promising future research paths are included.

Resumen

Durante las últimas décadas los algoritmos evolutivos se han vuelto populares como herramientas para la solución de problemas de optimización multiobjetivo (MOPs). Estos algoritmos evolutivos multiobjetivo, comúnmente conocidos como MOEAs, han demostrado ser una opción muy conveniente y por eso mismo han dado origen al estudio de aplicaciones y mejoras en su eficiencia. Tradicionalmente estos algoritmos no explotan la información específica del dominio de los MOPs; en particular, no utilizan información que tenga que ver con la continuidad ni diferenciabilidad de sus funciones. En esta tesis, estudiamos la manera de explotar información relacionada con los gradientes de las funciones objetivo, como una manera de guiar la búsqueda de los MOEAs y hacerlos más eficientes. También se analiza el uso de esta información como herramienta para refinar la precisión de las soluciones. El trabajo de esta tesis se enfoca en dos objetivos, el primero es construir buscadores locales eficientes que combinen y exploten la información obtenida de los gradientes de las funciones. El segundo objetivo es incorporar estos buscadores locales dentro de los MOEAs para generar algoritmos híbridos. Se proponen varios buscadores locales con características diversas y varias opciones para su integración con MOEAs. Se presentan análisis teóricos y pruebas experimentales de su eficacia y eficiencia en problemas de prueba tradicionales. Se mencionan pautas para hacer eficiente este tipo de integración de heurísticas y se presentan conclusiones.

Para Elisa y Jesús.
Porque los tres somos un solo corazón.

Agradecimientos

Las palabras que pondré en este espacio no harán justicia, para nada, al enorme agradecimiento que tengo por las personas que han estado a mi lado durante estos últimos cuatro años. En primer lugar agradezco al Dr. Oliver Schütze por todo lo que me ha enseñado acerca del trabajo y por ayudarme a redescubrir aquello que me acercó a las matemáticas por vez primera. Le agradezco también por todas las semanas de trabajo intenso y por compartir conmigo los proyectos, y conocimientos, que dieron lugar a esta investigación. Al Dr. Carlos Coello por su paciencia infinita durante los más de 11 años de conocerme; sin su ejemplo y ayuda nunca hubiera empezado, continuado, ni terminado este proyecto. Porque él siempre tiene una respuesta útil a mis preguntas (científicas y no científicas). Mis dos asesores han marcado mi vida profesional en muchos sentidos, delimitando mi actuación como profesora e investigadora, pero también siendo ejemplo del lado humano que acompaña a esta profesión. Agradezco también al Dr. Gerardo De la Fraga, al Dr. Debrup Chakraborti, al Dr. Arturo Hernández y la Dra. Katya Rodríguez por leer esta tesis y por sus valiosos comentarios, no solo al final sino también durante seminarios y otras entregas parciales de este trabajo.

Por supuesto agradezco a Jesús González su amor y dedicación como esposo y padre; por estar incondicionalmente a mi lado, y por literalmente "relevarme" mientras yo escribía esta tesis. También agradezco a mi madre Dulce Ma. López, mi suegra Elsa Espino, mis hermanas Susy y Ana Laura, y todos quienes en algún momento cuidaron de mi y/o de mi bebé en tantos momentos de enfermedad. Sin su ayuda las circunstancias ciertamente me hubiesen vencido, dejando inconcluso este proyecto.

Quiero hacer mención especial de mi amigo José Luis Enriquez, por su amistad, y por siempre resolver mis problemas técnicos (y existenciales). Por su ayuda moral, profesional y "en especie" que fue fundamental para realizar muchos de los experimentos de mi investigación. A mis colegas de la sala de estudiantes: Alfredo Arias, Antonio López, Ma. de Lourdes López, Julio Barrera, Zaúl Zapotecas, Adriana Menchaca, Eduardo Vázquez, Luis V. Santana por estar siempre dispuestos a compartir ideas, risas y hasta café. A todo el personal del Departamento de Computación del CINVESTAV; especialmente a las secretarias Sofía Reza, Felipa Rosas y Erika Ríos por su constante profesionalismo y calidez.

A la memoria de mi padre, cuyo amor y enseñanzas me acompaárán por siempre.

Parte del trabajo de esta tesis fue apoyado por el proyecto CONACyT no. 103570, cuyo investigador principal es el Dr. Carlos A. Coello Coello. Agradezco también al Instituto Politécnico Nacional por la licencia laboral otorgada durante la realización de estos estudios.

Agradezco al CONACyT por la beca económica otorgada para la realización de estos estudios, y el apoyo del proyecto CONACyT no. 128554 bajo la dirección del Dr. Oliver Schüetze.

Products obtained while developing this research

Journal Papers

1. Adriana Lara, Gustavo Sanchez, Carlos A. Coello Coello, and Oliver Schütze. **HCS: A New Local Search Strategy for Memetic Multi-Objective Evolutionary Algorithms**. *IEEE Transactions on Evolutionary Computation*, **14**(1):112–132, February 2010

2. Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **On the influence of the number of objectives on the hardness of a multiobjective optimization problem**. *IEEE Transactions on Evolutionary Computation*, **15**(4):444–455, August 2011

3. Oliver Schütze, Xavier Esquivel, Adriana Lara, and Carlos A. Coello Coello. **Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multi-Objective Optimization**. *IEEE Transactions on Evolutionary Computation. (in press)*, (99):DOI: 10.1109/TEVC.2011.2161872, 2012

4. Oliver Schütze, Adriana Lara, Carlos A. Coello Coello, and Massimiliano Vasile. **On the detection of nearly optimal solutions in the context of single-objective space mission design problems**. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **225**(11):1229–1242, 2011

Book Chapter

Adriana Lara, Oliver Schütze, and Carlos A. Coello Coello. *EVOLVE: A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, chapter On Gradient-based Local Search to Hybridize Multi-objective Evolutionary Algorithms. Studies in Computational Intelligence. Springer (in press)

Technical Reports

Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **The Directed Search Method for**

Multi-Objective Optimization Problems. Technical report, http://delta. cs. cinvestav. mx/~schuetze/technical_reports/index. html, 2009

Papers in Conference Proceedings

1. Adriana Lara, Carlos A. Coello Coello, and Oliver Schütze. **Using Gradient-Based Information to Deal with Scalability in Multi-objective Evolutionary Algorithms**. In *IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 16–23, Trondheim, Norway, May 2009. IEEE Press

2. Adriana Lara, Oliver Schütze, and Carlos A. Coello Coello. **New Challenges for Memetic Algorithms on Continuous Multi-objective Problems**. In *GECCO 2010 Workshop on Theoretical Aspects of Evolutionary Multiobjective Optimization*, pages 1967–1970, Portland, Oregon USA, July 2010. ACM

3. Adriana Lara, Carlos A. Coello Coello, and Oliver Schütze. **A painless gradient-assisted multi-objective memetic mechanism for solving continuous bi-objective optimization problems**. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE Press, 2010

4. Adriana Lara, Carlos A. Coello Coello, and Oliver Schütze. **Using gradient information for multi-objective problems in the evolutionary context**. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO 2010)*, pages 2011–2014. ACM, 2010

5. Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **Evolutionary Continuation Methods for Optimization Problems**. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 651–658, Montreal, Canada, July 8–12 2009. ACM Press. ISBN 978-1-60558-325-9

6. Oliver Schütze, Xavier Equivel, Adriana Lara, and C.A. Coello Coello. **Some Comments on GD and IGD and Relations to the Hausdorff Distance**. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1971–1974. ACM, 2010

7. Oliver Schütze, Adriana Lara, Carlos A. Coello coello, and Massimiliano Vasile. **Computing approximate solutions of scalar optimization problems and applications in**

space mission design. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE

8. Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **The Directed Search Method for Unconstrained Multi-objective Optimization Problems**. In *EVOLVE 2011, A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, Bourglinster Castle, Luxembourg, May 2011

9. Adriana Lara, Sergio Alvarado, Shaul Salomon, Gideon Avigad, Carlos A. Coello Coello, and Oliver Schütze. **The Gradient Free Directed Search Method as Local Search within Multi-Objective Evolutionary Algorithms**. In *EVOLVE 2012, A bridge between Probability, Set Oriented Numerics and Evolutionary Computation (to appear)*, CINVESTAV, Mexico City, Mexico, August 2012

Award

Best Paper and Presentation Award of the Graduate Student Workshop at the Genetic and Evolutionary Computation Conference, GECCO 2010.

Contents

Acronyms

DS	*Directed Search*
EA	*Evolutionary Algorithm*
EC	*Evolutionary Computation*
EMO	*Evolutionary Multi-objective Optimization*
GBMES	*Gradient-based Evolutionary Strategy*
HCS	*Hill Climber with Side Step*
KKT	*Karush-Kuhn-Tucker*
LS	*Local Search*
MO	*Multi-objective Optimization*
MOEA	*Multi-objective Evolutionary Algorithm*
MOP	*Multi-objective Optimization Problem*
NSGA-II	*Elitist Non-dominating Sorting Genetic Algorithm*
SOP	*Single-objective Optimization Problem*
SPEA2	*Strength Pareto Evolutionary Algorithm*

List of Figures

List of Tables

List of Algorithms

"Computer Science is no more about computers than astronomy is about telescopes."

E.W. Dijkstra

1 Introduction

The necessity of finding the best solution, for a particular problem, arises in many areas of engineering and science. Improved solutions are sought in industry, economics, and other human activities which involve planning and limited resources. The area of mathematics which deals with this class of task is known as optimization; and, since the advent of digital computing, optimization algorithms have attracted an increasing interest from researchers. Moreover, when solving many real-life situations, multiple criteria must normally be considered. The task of simultaneously optimizing several objective functions (which are normally in conflict with each other) is called *Multi-objective Optimization* (MO); and the problem that this area deals with is properly called *Multi-objective Optimization Problem* (MOP), also known as a vector optimization problem. Several engineering applications can be modeled by this kind of problems [21, 22, 32, 43, 109, 128].

MOPs have been numerically solved, since their origins, using methods supported by specialized mathematical programming techniques. The use of *Evolutionary Algorithm*s (EAs) for solving MOPs started in the mid-1980s. Since then, the use of the so-called *Multi-objective Evolutionary Algorithm*s (MOEAs) has had a significant growth. EAs are, in general, stochastic search techniques inspired on mimicking the principles of evolution (*i.e.*, the survival of the fittest) in nature. EAs apply a set of variation operators (*e.g.*, crossover, mutation, etc.) to combine different solutions, aiming to produce reasonably good approximations of the optimum. The advantages and issues of these different methods—EAs and mathematical programming techniques—are still under study by several research groups. Solving MOPs is a hard task since there are several basins of attraction that can bias the search. In fact, because of their nature, when solving MOPs it is very convenient to know effective search directions that can speed up the process. Obtaining this knowledge may imply spending an additional computational effort. However, in certain applications, in which the evaluation of the objective functions is computationally expensive, this extra effort is well justified.

This work is focused on coupling these two paradigms—EAs and mathematical programming techniques—in order to exploit their particular advantages when solving MOPs. On smooth problems, gradient information is a powerful tool which is absolutely not exploited by the regular operators of EAs. In a certain way this can be seen as a drawback of these latter methods because of their slow convergence rate, even in problems when differentiable objectives are present. On the other hand, because of their nature, gradient-based procedures have a local scope and are not able to find entire optimal sets by themselves; furthermore, they are usually stuck on local (stationary) optima. Therefore, combining these procedures with a global stochastic strategy—such as EAs—constitutes a promising mechanism to generate good solutions. This mix is done in order to obtain an algorithm which offers, on the one hand, the globality and robustness of the evolutionary approach; but on the other, also an improved overall performance by the inclusion of a well directed *Local Search* (LS) mechanism. There is some evidence [78, 56] that hybridization using gradient information can really speed up convergence of MOEAs on continuous problems. However, the application of the LS procedure some times goes wrong, and for certain problems, the performance of the hybrid is not even as good as the stand-alone MOEA. This opens the door to new research about the way of hybridizing these techniques.

Introducing gradient information of a MOP (when it is available) into a MOEA is a research topic that has attracted a lot of interest in recent years [76, 19]. There are some hybrids of MOEAs and gradient-based methods currently available in the specialized literature [130, 51, 56, 132]. Such approaches normally replace [130] or add [132], existing evolutionary operators such that the available gradient information is used to guide the search. The motivation for this sort of coupling has been, first, to speed up convergence towards the Pareto front [8, 132, 83, 130], and second, to produce more accurate solutions [132]. Nevertheless, the studies about theoretical and practical aspects of this coupling are still scarce, giving rise to a variety of research possibilities, which motivated the work presented in this thesis.

Gradient information provides a MOEA with search directions to perform more accurate movements. However, computing such search directions is also a MOP since each objective provides its own set of (gradient-based) search directions; therefore, all of these directions need to be properly combined into a single one, in order to perform a line search procedure—particularly adapted for the multi-objective case. Early approaches are presented in [42, 107, 53] and [8].

Another major issue arising when incorporating gradient information into a MOEA is how to provide a proper balance between the LS (*i.e.*, operators using the gradient information) and the global search (*i.e.*, the MOEA). In fact, such a balance is problem-dependent [67]. For example, in [51] the use of the LS is proposed to be used at the end of the evolutionary process, and the interleaving between the local and the global search takes place after a certain (fixed) number of objective function evaluations. However, other proposals use the gradient information to overtake solutions towards the Pareto optimal set, and adopt the evolutionary search in a second stage (see for example [56, 83]). In any case, none of these types of strategies is really ideal, since the best would be to have an adaptive mechanism that allows the two types of search to interleave during the run of the MOEA, such that each of them intervenes whenever needed.

LS algorithms move from an initial solution to another one in the search space by applying local changes (typically over particularly defined neighborhoods), until a better solution is found or a time bound is reached. There exist a plethora of LS methods for continuous (and discrete) optimization problems, and some authors have reported successful hybridizations of

LS techniques, in general, with genetic algorithms. However, to the author's best knowledge, there exist basically three crucial questions [54, 133, 79, 63, 89, 7] which remain open in the design of memetic strategies: Where shall a LS process be hybridized with a genetic algorithm? Which individuals should be fine-tuned and how much? And, when shall the local refinement be applied? These questions apply both, to single and to multi-objective problems.

Problem Statement

The main goal of this thesis is to investigate how gradient-based information can effectively help MOEAs during the search, and studying the issues that arise when combining these two procedures. One particular goal is to look deeply into the calculation of multi-objective gradient-based search directions—a basic topic when trying to design hybrid MOEAs, using gradient-based line search. A second particular aim is incorporating these directions into LS engines, as well as devising new possible movement directions. Also, analyzing the cost and differences among them, in order to study some aspects regarding their practical implementation.

Another particular aim of this work is, precisely, to propose an adaptive mechanism that allows the local and global search strategies to interleave. The ideal case is when the algorithm can automatically assign more or less resources either to the local or to the global search procedure, as deemed necessary.

Contributions

The main contributions of this thesis are the following:

- We state a framework for the analysis and the application of gradient-based local searchers in the multi-objective case (Chapter 3). Also, we study the many-objective problem, from the descent cone point of view, and get important results to disprove a common belief about sources of difficulty for MOEAs (Section 3.5).

- For two-objective problems, we present a simple and efficient method to calculate a descent direction; we show the effectiveness of this method and analyze its limitations (Chapter 3). Furthermore, its effectiveness inside a memetic MOEAs is studied (Chapter 6).

- We present two new local searchers, particularly designed to be combined with MOEAs. Both show a lot of potential for hybridization due to their special features. First, the Hill Climber with Side Step which is able to search towards and along the Pareto front—making the switch between the two movements in an automatic way. Second, the Directed Search method which is able to direct and steer the search towards regions of interest (Chapter 4).

- Diverse implementations for coupling our local searchers with MOEAs are presented. The experiments show the benefits of the hybridization and some guidelines for building this type of algorithms (Chapters 5 and 6).
- An adaptive control for the use of LS, inside the memetic MOEA, is proposed and tested. This is based on the dynamics of the size of the non-dominated individual set (Chapter 6).
- The scalability of gradient-based methods is investigated, both, increasing the number of variables and also the number of objectives (Chapters 3 and 5).

Outline of the Thesis

Including this introduction, the thesis consists of seven chapters. Chapter 2 presents basic definitions and concepts as a background for the following chapters of this document.

Chapter 3 is dedicated to the study of the properties induced by gradients when analyzing MOPs. Definitions of descent directions and their importance are presented. We also present, some important geometrical concepts required to understand the rest of the thesis. Finally, we include our insights about the descent cones point of view when dealing with many-objective problems (MOPs with more than three objective functions).

Next, we present our proposals for gradient-based search directions: the simplest one for two-objective problems is introduced in Chapter 3, and in Chapter 4 the Hill Climber with Side-step (HCS) and the Directed Search Method (DS) are presented. The HCS is a novel iterative search procedure which is capable of moving both toward and along the (local) Pareto set, depending on the distance of the current iterate toward this set. It utilizes the geometry of the directional cones present in MOPs and works with or without gradient information. Meanwhile, the Directed Search Method allows to steer the search process in a particular desired direction, established in the objective space. This also counts with a continuation-based procedure, which allows to reach other points over the Pareto set.

Chapter 5 focuses on the hybridization of gradient-based LS and MOEAs. First, a two-stage algorithm named Gradient Based Multi-objective Evolutionary Strategy (GBMES) is presented; this is a hybrid between an elitist MOEA, and a gradient-based descent method. This algorithm requires a low number of objective function evaluations to converge to a few points in the Pareto front; then, the rest of the Pareto front is reconstructed. Emphasis is placed on the effectiveness of this hybrid approach when increasing the number of decision variables, and a study of its scalability is also presented. In this chapter we also show two possible ways to integrate the HCS into a given MOEA, leading to new memetic MOEAs. We use the state-of-the-art algorithms NSGA-II and SPEA2 as baseline MOEAs. Numerical results on some widely used benchmark problems are presented, indicating the benefits of using the proposed local searcher within a MOEA.

In Chapter 6 we present, and investigate, a simple and generic way to add gradient-based information, as a means to improve the search process performed by a MOEA. We present ideas that can be easily incorporated into any MOEA and provide some guidelines regarding the use of our proposed approach. We also propose an adaptive mechanism that allows the local and global search strategies to dynamically interleave. We based the control mechanism on monitoring the evolutionary search efficiency; in this way, the algorithm can automatically assign more or less resources either to the local or to the global search procedure in a dynamic way.

Finally, the conclusions of the thesis, as well as some possible paths for future research, are presented in Chapter 7.

"In mathematics you don't understand things.
You just get used to them."

John von Neumann

2 Preliminary Concepts

2.1 The Multi-objective Optimization Problem

We are interested in solving problems of the type:

$$\text{Minimize } F(x) := [f_1(x), f_2(x), \ldots, f_k(x)]^T \tag{2.1}$$

subject to:

$$g_i(x) \leq 0 \quad i = 1, 2, \ldots, m \tag{2.2}$$

$$h_i(x) = 0 \quad i = 1, 2, \ldots, p \tag{2.3}$$

where $x = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ is the vector of *decision variables* (also known as *decision parameters* vector or *solution* vector), $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, k$ are the *objective functions* (or *objectives,* in short) and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, $j = 1, \ldots, p$ are the constraint functions of the problem. If functions g_i and h_i are not present, we are dealing with an *unconstrained* MOP. Solving the above problem is known as solving a MOP.

If any function in the original problem must be maximized (for example, profit), it is possible to restate it as a minimization approach (using the *duality principle* when multiplying the function by -1). Therefore, the problem can be stated in general as above. Even though MOPs can be defined over other domains—like discrete sets, for example—in this work we are only interested in continuous domains which are contained on $\mathbb{R}^n$. When all the objective functions and the constraint functions are linear, the problem (2.1) is called a *linear* MOP, and there are several techniques to solve it. If at least one of the functions is nonlinear, the problem is then called a *nonlinear* MOP. If all the objective functions are convex, and also the feasible region is convex the problem is known as a *convex* MOP. In this study we are dealing with nonlinear problems, either convex or not. Several conditions, such as differentiability or continuous differentiability, will be assumed for the f, g and h functions during the remainder of this document.

Solving a MOP is very different that solving a *Single-objective Optimization Problem* (SOP). Since some of the f_i are normally "in conflict" with each other,[1] the solution of a MOP is not unique; this is because normally no single solution exists that provides the

[1]For example, one objective may refer to manufacture cost and another to quality of the product.

best possible value for all the objectives (see Figure 2.1). Consequently, solving a MOP implies finding a trade-off among all the objective functions. This involves the generation of a set of possible solutions instead of a single one—like in the single-objective optimization case. The notion of "optimality" that we just informally described, was originally proposed by Francis Ysidro Edgeworth in 1881 [36] and was generalized by Vilfredo Pareto in 1896 [97]. This concept is known today as *Pareto optimality* and will be formally introduced next, after providing some other basic definitions.

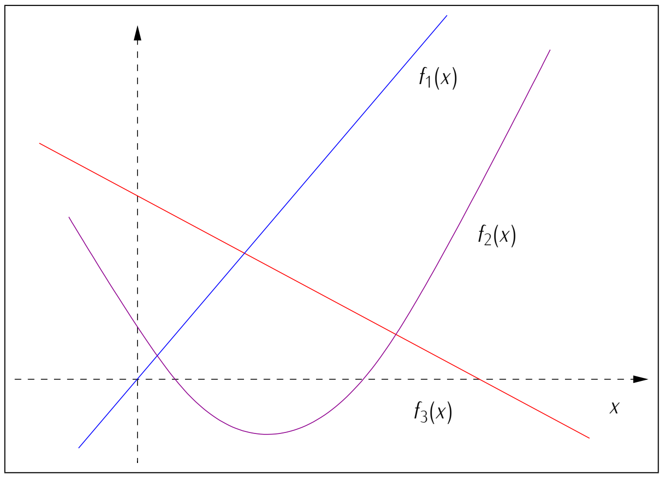


Figure 2.1: This figure shows function f_3 in conflict with functions f_1 and f_2.

Definition 2.1.1 *Given two vectors $x, y \in \mathbb{R}^n$, we say that x* **dominates** *y (denoted by $x \prec y$) if $f_i(x) \leq f_i(y)$ for $i = 1, \ldots, m$, and $F(x) \neq F(y)$.*

Definition 2.1.2 *We say that a vector of decision variables $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is* **non-dominated** *with respect to $\mathcal{X}$ (where $\mathcal{X}$ is the feasible region), if there does not exist another $x' \in \mathcal{X}$ such that $x' \prec x$.*

Definition 2.1.3 *We say that a vector of decision variables $x^* \in \mathcal{X} \subset \mathrm{R}^n$ is* **Pareto optimal** *if it is non-dominated with respect to $\mathcal{X}$.*

Definition 2.1.4 *a) The Pareto Optimal Set or* **Pareto Set** *$\mathcal{P}^*$ is defined by:*

$$\mathcal{P}^* = \{x \in \mathcal{X} \mid x \text{ is Pareto optimal}\}.$$

b) The **Pareto Front** *$\mathcal{PF}^*$ is defined by:*

$$\mathcal{PF}^* = \{F(x) \in \mathbb{R}^k \mid x \in \mathcal{P}^*\}.$$

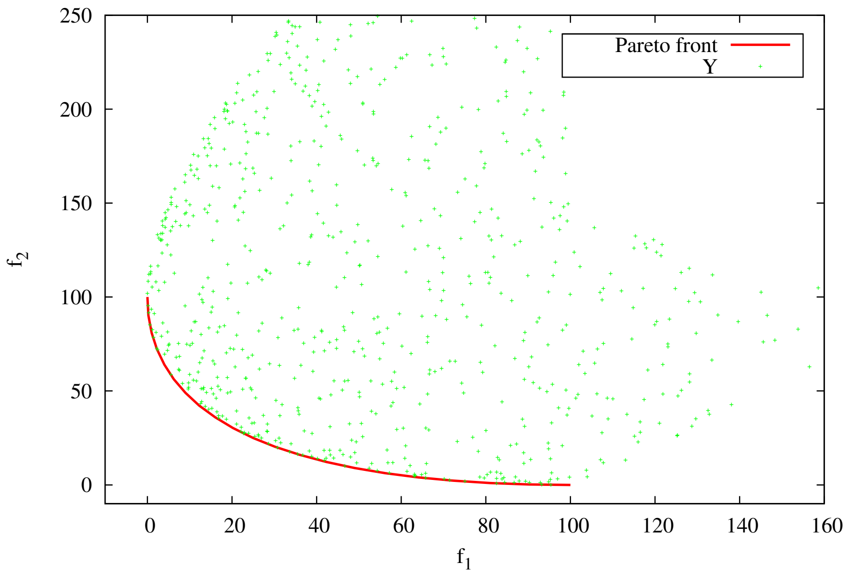


Figure 2.2: This figure emphasizes the Pareto front for Example 2.1.1. The axes represent the value regarding each function—what is known like the *objective space*. The points of Y are the images of randomly generated vectors from the domain.

We thus wish to determine the Pareto optimal set from the set $\mathcal{X}$ of all the decision variable vectors that satisfy the constraints of the problem. Note however that in practice, just a finite representation of the Pareto optimal set is normally achievable.

Assuming x^* as a Pareto point of (2.1), there exist [81] a vector $\alpha \in \mathbb{R}^k$, with $0 \leq \alpha_i, i = 1, \ldots, k$ and $\sum_{i=1}^{k} = 1$ such that

$$\sum_{i}^{k} \alpha_i \nabla f_i(x^*) = 0. \tag{2.4}$$

A point x^* that satisfies (2.4) is called a *Karush-Kuhn-Tucker* (KKT) point.

Example 2.1.1 *Consider the following unconstrained MOP:*

$$\text{minimize } F(x, y) := \left[x^2 + y^2\, ,\, (x-10)^2 + y^2\right]^T, \tag{2.5}$$

with $x, y \in \mathbb{R}$.

The Pareto set of this problem is the line segment $[(0,0),(10,0)] \in \mathbb{R}^2$, *and the Pareto front is shown, as a continuous line, in Figure 2.2.*

2.2 Solving a MOP

The most common procedures to solve (2.1) are classified [91, 92] in four classes: *no-preference methods, a posteriori methods, a priori methods and interactive methods,* according to the stage, an level, in which the decision maker intervenes. In this work, we focus our attention on the *a posteriori* methods, in which the goal is to obtain the best approximation of the entire set of optima, which will be presented *a posteriori* to the decision maker—who will then select the most suitable solution out of it. When tackling the *a posteriori* approach, the use of EAs is a natural choice. This is due to the fact that no previous knowledge regarding the MOP is necessary for the algorithm; and also because, at the end of the run, the population-based algorithm throws an entire set of (ideally) well distributed solutions. We present in the following, a brief review of some classical techniques in order to mention important aspects for comparison against EAs.

2.2.1 Classical mathematical techniques

Operations Research (OR), is a branch of mathematics within which a variety of techniques have been developed to deal with MOPs. These approaches developed within OR for solving MOPs are known as *classical methods*. Up to 1980 most of the computational methods to solve MOPs consisted of minimizing only one function, either using the other objective functions as constraints of the problem, or simply by taking a combination of all the objectives [94]. The most common way to tackle a MOP is by *scalarization* which means reducing the problem to a SOP. One example of this approach is the following method:

Weighted sum method: This method consists of transforming the vector of function values into a scalar value using an aggregating function over the vector function, getting the following problem:

$$\begin{aligned} \text{Minimize} \quad & g_\omega = \sum_{i=0}^{k} \omega_i f_i(x) \\ \text{subject to} \quad & 0 < \omega_i \text{ for all } i \in \{1, \ldots, k\}; \\ \text{and } & \sum_{i=0}^{k} \omega_i = 1;\ x \in S. \end{aligned}$$

In this way, the solution set consists only of one point for each weight combination. An important drawback of this approach is that controlling the weights does not necessary help us to control the distribution of the points in the parameter space. Besides, there are points that can not be generated as a combination of weights in non convex cases—see [27] for a more in-depth explanation about this. There exist, in general, many scalarization methods which transform the MOP into a 'classical' SOP. It is worth to notice that by choosing a clever sequence of SOPs, a suitable finite size approximation of the entire Pareto set can be obtained (see [26, 74, 40, 37] and references therein). Another approach to approximate the entire Pareto set is to use set oriented methods such as subdivision techniques [33, 68].

ϵ-constraint method: In the *ϵ-constraint method* [48], one of the objectives is chosen for minimization while the rest of the objectives conform a set of constraints limited by user specifyed bounds ϵ_i. *i.e.:*

$$\begin{array}{rl} \text{Minimize} & f_j \\ \text{subject to} & f_i \leq \epsilon_i \;\text{ for all } i \in \{1, \ldots, k\}, i \neq j. \end{array}$$

The ϵ-constraint problem should be solved using multiple different values for ϵ_i, if several Pareto optimal solutions are desired. This method can deal with convex and non convex functions; but, choosing the ϵ_i values is still an issue since there is no warranty that a feasible optimum exists for a specific ϵ_i. An in depth analysis of these method can be found in [91].

Most of the classical methods require the prior fulfillment of certain geometrical or analytical properties imposed on the MOP, such that they can properly work. For instance, in problems with nonlinear functions, some methods (like those in [91]) use *gradient* information. Since the differentiability properties of functions provides local information (*i.e.*, an appropriate search direction over a certain neighborhood), any technique that uses such gradient information can be seen as a LS procedure.

2.2.2 Multi-objective evolutionary algorithms

As an alternative to the classical methods, researchers in the area of *Evolutionary Computation* (EC) [46, 6] have developed new approaches to approximate the solution of MOPs. This area, which is now known as *Evolutionary Multi-objective Optimization* (EMO) has given rise to a wide variety of MOEAs [23, 30] which do not require special features or properties from the objective functions, and rely on stochastic search procedures. The use of EAs to solve MOPs has several advantages. In the first place, a MOEA generates a set of solutions instead of a single solution in one run—which is not the case in most OR methods. Furthermore, MOEAs are general search engines which do not impose special requirements on the type of objective functions that can be solved. Note however, that differentiability is a requirement that we will assume in this thesis, because of the sort of work that we aim to develop.

During the last decade, MOEAs have become very popular in a wide variety of application domains, and are now a very fertile research area [22]. Algorithm 1 describes a generic MOEA. The most common way to perform the selection of individuals (lines 13–14) for survival is according to the Pareto dominance relation. The goal of this class of algorithms is to provide a reasonably accurate approximation of the true Pareto front of the problem. When using MOEAs, approximating the Pareto set involves seeking for two goals: (a) minimize the distance between the output population and the theoretical Pareto set, and (b) make a diverse sample (with a good spread and uniform distribution) of output solutions. It is worth to notice that (a) and (b) represents conflicting aims during the execution of a MOEA.

```
Require: MOEA's particular parameters
Ensure: A set $S$ which approximates the Pareto front of the MOP.
 1: Randomly generate a population of individuals $P_t$, ($t = 0$).
 2: for each individual $p \in P_t$ do
 3:     Evaluate the objective functions.
 4:     Assign the corresponding fitness value.
 5: end for
 6: while temination condition is not reached do
 7:     Select $P'_t \subset P_t$ based on the fitness.
 8:     Produce the offspring population $P''_t$ by recombination and mutation.
 9:     for each individual $p'' \in P''_t$ do
10:         Evaluate the objective functions.
11:         Assign the corresponding fitness value.
12:     end for
13:     Select the individuals to survive from a combination of $P_t$ and
14:             $P''_t$, in order to conform $P_{t+1}$.
15:     $t = t + 1$.
16: end while
17: $S = P_t$.
```

Algorithm 1: Description of a generic MOEA.

One of the current challenges on the design of MOEAs is to find structures and mechanisms that increase their efficiency [20]. Fitness values are normally tightly coupled to the objective function values; therefore, there is a difficulty to reduce the number of fitness function evaluations, since this particular information is required to guide the search. We describe, in the following, two state-of-the art MOEAs which are the most popular in the current literature.

Elitist Non-dominating Sorting Genetic Algorithm: The *Elitist Non-dominating Sorting Genetic Algorithm* (NSGA-II) was proposed by K. Deb *et al.* [31] and, due to its proven robustness and efficacy, it has been widely used as a reference to assess the performance of new MOEAs. It has remarkable differences with its predecessor, the Non-dominating Sorting Genetic Algorithm (NSGA) [135], other than the addition of an elitism mechanism.

Algorithm 2 shows in pseudo-code the main parts of the NSGA-II. Lines 3 and 4, in the algorithm, refers to the process known as *nondominated sorting;* this process consists of clasifying the population into several disjoint layers (non-dominated sets) $F_i's$, such that $P = \cup F_i$. The main feature of these classes F_i is that any two members of the same class are incomparable in the Pareto sense. As a second criterion for ordering, after considering the Pareto rank of each solution, the *crowding* distance value is used (Line 18). Crowding distance is an indicator of the density of individuals around a particular individual p_i inside the population. For this, the average distance of two solutions, on either side of solution p_i, is taken along each objective. To create an offspring population, binary tournament selection followed by recombination and mutation is used. In this case, a modification called *crowded tournament* is employed for selection. In this operator, a solution p_i wins a tournament against other solution p_j if p_i has a better (smaller) Pareto rank or, in the case that both have the same rank value, when p_i has a better crowding distance value. This last condition provides a way to proceed in case of incomparable solutions, and, in terms of the evolutionary process, it helps to mantain diversity

```
 1: procedure NSGA-II($gen$)
 2:     Randomly generate a population of individuals $P_t$, ($t = 0$).
 3:     Sort the population into a set of different domination levels.
 4:     Assign the fitness to each solution as its non-domination level.
 5:     Create an offspring population $Q_t$, of size $N$, from $P_t$ by using
 6:           the crowded tournament selection, crossover and
 7:           mutation operators.
 8:     for $t = 1$ to $gen$ do
 9:         Combine parent and offspring populations to conform
10:           $R_t = P_t \cup Q_t$.
11:         Perform a non-dominating sorting to $R_t$ and identify the
12:           different fronts $F_i$, $i = 1, \ldots,$ etc.
13:         Set new population $P_{t+1} = \emptyset$. Set a counter $i = 0$.
14:         repeat
15:             $P_{t+1} = P_{t+1} \cup F_i$.
16:             $i = i + 1$.
17:         until $|P_{t+1}| + |F_i| < N$
18:         Perform the crowding-sort($F_i$) procedure and include the
19:           most widely spread $N - |P_{t+1}|$ solutions by using the
20:           crowding distance values in the sorted $F_i$ to $P_{t+1}$.
21:         Create an offspring population $Q_t$ from $P_t$ by using the
22:           crowded tournament selection, crossover and mutation
23:           operators.
24:     end for
25: end procedure
```

Algorithm 2: Description of the NSGA-II.

Strength Pareto Evolutionary Algorithm: Another widely used MOEA is the *Strength Pareto Evolutionary Algorithm* (SPEA2) [150]. This is a revised version of the original SPEA proposed by Zitzler and Thiele [148]. The algorithm uses an external archive—of a pre-specified size—in order to store the nondominated solutions produced at each generation.

SPEA2 employs an enhanced fitness assignment strategy compared to its predecessor SPEA as well as new techniques for archive truncation and density-based selection. For the fitness assignment of each individual, two quantities are taken into account: the number of individuals which dominate it, and those which are dominated by the individual to be assigned. The *strength* value for the individual is determined by these two numbers, and it is involved in the final computation of the particular individual fitness. A description of SPEA2 is provided in Algorithm 3. The steps described in lines 6 to 10 conform what is mentioned, by the authors, as the environmental selection. In this part of the algorithm, an individual is removed form the archive either if a solution has been found that dominates it, or when the maximum archive size is reached. Also, in the latter case, the individual is removed when lies on a region of the front with a high density of solutions in the archive.

```
 1: procedure SPEA2($gen$)
 2:     Randomly generate a population of individuals $P_t$, $(t = 0)$.
 3:     Create an empty external set $E = \emptyset$.
 4:     for $t = 1$ to $gen$ do
 5:         Compute fitness of each individual in $P_t$ and $E$
 6:         Select all non-dominated individuals from $P_t \cup E$
 7:           and store them in $E$.
 8:         Use the special truncation mechanism to remove elements
 9:           from $E$ when necessary due to the limited size of the archive.
10:         If there is place in $E$ fill it with dominated individuals from $P_t$.
11:         Conform $P_t'$ using by binary tournament selction with
12:           replacement from $P_t \cup E$.
13:         Apply crossover and mutation to $P_t'$ in order to fill $P_{t+1}$.
14:     end for
15: end procedure
```

Algorithm 3: Description of the SPEA2 algorithm.

2.2.3 Memetic algorithms

Algorithms that combine MOEAs with LS are called *multi-objective memetic algorithms* [93]. The work developed in this thesis consists of building new *hybrid MOEAs* (a general description which includes multi-objective memetic algorithms) that incorporate gradient-based information during the evolutionary process. The main design goal of such an approach will be the efficiency of the algorithm (*i.e.*, the proposed approach should perform a reduced number of objective function evaluations as compared to state-of-the-art MOEAs) on standard test functions.

Even though this work focuses just on certain types of MOPs (*i.e.*, those with differentiable functions), an important number of complex real-world applications exist in which these conditions are fulfilled (see for example some references within [14]). Also, some problems can be modified so that the geometrical conditions required by these types of approaches can be satisfied (see for example [58]).

Hybrid algorithms have been successfully applied to find solutions of SOPs. In the multi-objective case, Knowles gives in [77] some examples of hybrid algorithms (MOEA + LS) working over discrete MOPs. In addition, some efforts have also been done for the continuous case [130, 8, 14, 53] (see Section 2.3 for a wider description). However, an important question is still open: how to hybridize mathematical methods based on geometrical properties of a MOP (e.g., methods using gradient information) with MOEAs, such that the resulting approach is more efficient (in terms of the number of objective function evaluations performed) than state-of-the-art MOEAs while producing results of comparative quality.

One of the main questions to solve during the design of these algorithms is: how to introduce the gradient-based information into the genetic operators of the MOEA? Another possibility is not to apply the LS procedure directly as part of the genetic operators; if that is the case, which should be the right place to introduce the LS procedure? Also, while answering these questions it is necessary to keep in mind the evident trade-off between the computational cost and the benefits of the LS procedure. This leads also to the development of suitable (gradient-based) local searchers. Finally, an important decision is: from which solutions should the local search be started? In the sequel, we address all these questions and present the details of our study. Next, we present a short revision of the previous work existing on this subject.

2.3 Previous Work

Hybridization of MOEAs with LS algorithms has been investigated for more than one decade. One of the first memetic MOEAs for models on discrete domains was presented in [64, 65] as a 'Multi-Objective Genetic Local Search' (MOGLS) approach. We can also mention [70] where Jaszkiewicz proposed the Pareto Memetic Algorithm (PMA). Another important MEMOEA, called M-PAES, was proposed in [75]. Unlike Ishibuchi's and Jaszkiewicz's approaches, M-PAES does not use scalarizing functions, but employs instead a Pareto ranking based selection coupled with a grid-type partition of the objective space. Two archives are used: one that maintains the global non-dominated solutions and another that is used por comparison purposes during the LS phase.

In [95], the authors proposed a LS process with a generalized replacement rule based on the dominance relation. Caponio and Neri [16] proposed the *Cross Dominant Multi-Objective Memetic Algorithm* (CDMOMA), which consists of the NSGA-II [31] combined with two local search engines: a multi-objective implementation of the Rosenbrock algorithm

[104], which performs very small movements, and the Pareto Domination Multi-Objective Simulated Annealing (PDMOSA) [136], which performs a more global exploration. Soliman et al. [134] proposed a memetic version of a co-evolutionary multi-objective differential evolution (CMODE-MEM) approach, which evolves both a population of solutions and promising search directions. In [139, 141, 142, 140], methods are presented which are hybrids of evolutionary search algorithms and multi-agent strategies, where the task of the agents is to perform the LS.

For the continuous case—*i.e.*, continuous objectives defined on a continuous domain—the first attempts started, to the author's best knowledge, with [45], where a neighborhood search was applied to NSGA-II [31]. This is a very simple scheme and the authors found that the added computational work impacted severely the efficiency of the algorithm.

In [59] a gradient-based local algorithm (Sequential Quadratic Programming (SQP)), was used in combination with NSGA-II and SPEA [151] to solve the *Zitzler-Deb-Thiele* (ZDT) benchmark suite [149]. The authors stated that if there are no local Pareto fronts, the hybrid MOEA has faster convergence toward the true Pareto front than the original approach. Furthermore, they found that the hybridization technique does not decrease the solution diversity.

In [1], three different LS techniques were hybridized with MOGA: simulated annealing, hill climbing and tabu search. In [102], the authors proposed a hybrid technique that combines the robustness of MOGA-II [100] with the accuracy and speed of NBI-NLPQLP. In [145], the proposed LS process employs quadratic approximations for all the objective functions.

One successful hybrid approach was proposed in [60]. The authors proposed the algorithm MO-CMA-ES, a multi-objective CMA-ES [49], which combines the strategy parameter adaptation of evolution strategies with a multi-objective selection mechanism based on non-dominated sorting. Also, in [144], a novel EA for constrained optimization problems is presented: the so-called hybrid constrained optimization EA (HCOEA). The algorithm combines a niching genetic algorithm based on tournament selection while the best infeasible individual replacement scheme is used as a LS operator. Finally, the mix of NSGA-II with a reference point method has been proposed and studied in [131]; by this approach, the authors were able to accelerate the convergence of NSGA-II.

Methods Based on Descent Directions and Line Search

Talking specifically about differentiable problems, we focus on those methods that use gradient-based descent directions to perform line search. Almost all traditional optimization techniques for MOPs, exploit previous domain knowledge of the problem. For example, some methods [72] use the linear properties of the functions; also, many methods [91] that solve nonlinear MOPs use differentiability properties of the functions. One drawback of these techniques is precisely that they can not be applied to a wide variety of MOPs. Additionally, they require a starting point for triggering the search, and they normally produce a single solution per run (in fact, different starting points may lead to the same final solution in some cases). Considering this, to generate a set of solutions, Schäffler et al. [107] introduced in 2002 a stochastic mathematical method. The method uses the solution of a stochastic differential equation related to the Karush-Kuhn-Tucker conditions for optimality. Some criteria of Brownian motion are used by the method to finally generate a set of solutions to the MOP. This technique requires the functions to be continuously differentiable, and it only works for unconstrained MOPs.

A method to deal with constrained MOPs is shown by Hillermeier in [58]. He introduces an homotopy approach using differential geometry and parametric-optimization concepts to model the MOP by k objective functions on an n-dimensional space. The Pareto set of the problem can be seen as a $k-1$ dimensional manifold. In this method, all the functions are assumed to be twice continuously differentiable, too. Hillermeier states that this method is scalable to problems of high dimensionality. Even when this method also calculates a set of Pareto optimal points, the method is of local nature, which is a drawback; but it is worth noticing, however, that this drawback can be avoided if the method is complemented with a stochastic technique—like a MOEA. Finally, in [114, 110, 50, 111], hybrids can be found were heuristic methods are coupled with multi-objective continuation methods.

For MOPs where the objective functions are continuously differentiable, the use of gradient information seems to be a natural choice. However, due to the local nature of gradient information, its combination with a global search engine such as a meta-heuristic is an obvious choice. In fact, several researchers [53, 14, 130, 8, 59] share the opinion that the combination of gradient-based methods and MOEAs can boost performance. They have proposed some ideas regarding how to combine these two pieces of information (*i.e.*, the fitness function information of the MOEA and the gradient-based information), but the research in this area is still scarce.

In [8, 9, 14, 42, 53], the idea of moving a particular solution toward a special improvement direction is used. The goal is to find a descent direction in the multi-objective case equivalent to the role played by the gradient of a function in the single-objective case. In order to do this, these authors combine the gradient of each objective function in different ways: Fliege [42] introduced, in 2000, a method called *Steepest Descent Direction* which uses a definition of the Multi-objective Gradient (MOG). Fliege's method implies solving a certain quadratic-programming problem involving the Jacobian matrix of F. This method works for convex Pareto fronts as well as for concave Pareto fronts. Later, Bosman and de Jong stated in [8] that the problem of finding an improvement direction for the multi-objective case is indeed a multi-objective problem as well. Therefore, the solution (*i.e.,* the "multi-objective gradient") must be a set of movement directions in which some of the objective functions, from the original MOP, decrease simultaneously while the others can either decrease or just maintain the same value. In other words, the aim is to find a set of descent directions rather than a single one. After that, Bosman and de Jong proposed an analytical way to calculate this set of directions. Their method only requires a few matrix operations and the solution of a linear optimization problem. Once the set of descent directions is settled, one of such directions is randomly selected. The method mentioned above is called *Combined-objectives Repeated*

Line Search (CORL) in [8]. A later improvement for CORL (by combining it with other criteria) can be found in [9]. In both cases, the hybridization of CORL is made with an *estimation distribution algorithm* (EDA) called $\mathbb{MIDEA}$ [10] in the following manner: At the end of the generational cycle, *i.e.*, after the variation operators have been applied, they choose a specific set of individuals to be moved by CORL. Thus, they obtain an improvement on the fitness for each one of them. The results reported in [8] regarding the best way to choose a set of individuals are inconclusive. Also, CORL has problems, according to Harada et al. [53], when looking for feasible descent directions in certain MOPs, and also when increasing the number of objective functions.

An alternative to the previous approach was proposed by Harada et al. [53]. They used the ideas introduced by Fliege [42] to build what they called the *Pareto Descent Method* (PDM). These researchers proposed PDM as an option to deal with particular constrained MOPs, when the solution lies on the boundary between the feasible and the infeasible regions. In these cases, it is necessary to find a different descent direction. The complexity of PDM is polynomial—as its authors refer—because the basic operations of this method consist of solving systems of linear equations. PDM assumes MOPs with no local Pareto fronts. Harada et al. [53] compare PDM with CORL and with a simple weighted linear aggregating function. They also evaluate a randomized generator of solutions, similar to the mutation mechanism used in the Evolution Strategies, and they conclude that this method is the worst performer. PDM does not show dramatic improvements over the other methods, except in the specific case when CORL has trouble (on a three-objective problem). In this particular situation, PDM does not obtain a set of descent directions—as CORL does—but it offers a good alternative. There is no hybrid algorithm based on PDM; this is left by the authors [53] as future work. In the same paper, the authors stated that their method has scalability issues when more than three objectives are used.

Brown and Smith [14] revised and emphasized the concept of the *descent cone* which is conformed by the intersection of the negative half-spaces (generated by the gradients) over all the objective functions. Brown and Smith proposed that the offspring, in a particular MOEA, must lie inside this cone; however, they do not propose a full algorithm. A procedure to approximate the gradient of the objective functions using neighborhood information is also introduced in [14]. According to its authors, that method reduces the computational cost of calculating the Jacobian matrix in Fliege's method. However, further testing and analysis are missing in order to apply the method.

The main problem of using an improvement direction is that it is impossible to know beforehand for how long a certain direction will be useful. Evidently, it makes sense to follow this promising descent direction as long as it remains as a good search direction. However, in problems with a very irregular geometry in the search space, the 'better' descent direction will be constantly changing. Thus, this issue remains as an important drawback when adopting gradient-based information. A particular drawback in Fliege's methods—on which [14], [53] and [8] are based—is that it has a slow convergence rate [14], and it is susceptible to getting trapped in local (*i.e.*, false) Pareto fronts.

In [130], Shukla introduced the use of two stochastic gradient-based techniques to improve the mutation mechanism of the NSGA-II. These two techniques are Schäffler's stochastic method [107] and Timmel's [137] method. Both hybrid algorithms were competitive in some modified versions of the well-known ZDT test problems, outperforming the plain NSGA-II. The ZDT4 problem, however, could not be properly solved by any of these hybrids. Only the NSGA-II was able to converge to the true Pareto front of ZDT4, since all the hybrids got trapped in local Pareto fronts. It is clear that the hybrids proposed by Shukla are relatively straightforward approaches that could be easily improved, but they also illustrate the local nature of the gradient-based information and its possible limitations. Additionally, Schäffler's method requires a huge number of objective function evaluations, which is an important drawback, if we consider that the main aim of using gradient-based information is precisely to reduce the overall computational cost of a MOEA.

"Where there is matter, there is geometry."
Johannes Kepler

3 The Geometry of Multi-objective Problems: A Gradient-based Approach

Gradient-based information can not be used in a straightforward way when solving a MOP. The main reason for this situation is that the gradients of the objective functions typically point toward different directions—due to the fact that the objective functions were originally in conflict each other. This bring us back to the task of finding a common improving direction into a MOP, again.

In this chapter, we look into the geometry induced by the objective function gradients, on the context of MOPs. The goal is to study descent directions acting simultaneously for all the objectives. Important implications of this point of view are also discussed throughout this chapter.

3.1 Descent Cones and Multi-objective Descent Directions

Descent cones geometry was first used in the multi-objective evolutionary context by Brown and Smith [14, 15]. The implications from their study became the basis of further proposals to obtain movement directions when performing multi-objective LS (see for example [8] and [86]). In a general sense, descent cones constitute the main tool to understand the local behavior of gradient-based methods dealing with MOPs. In the following, we state the main concepts and introduce the notation required for further discussion.

Let $f_1, \dots, f_k : \mathbb{R}^n \to \mathbb{R}$ be continuous and differentiable, and $\langle \cdot, \cdot \rangle$ denote the standard inner product in $\mathbb{R}^n$. Let also

$$\nabla f_i(x) = \left[\frac{\partial f_i(x)}{\partial x_1}, \dots, \frac{\partial f_i(x)}{\partial x_n} \right]$$

be the gradient of the function f_i at x. It is well known, and easy to prove[1] that $-\nabla f_i(x)$ points to the maximal decreasing direction for f_i at x (see Figure 3.1).

[1]Assuming unitary vectors, the directional derivative of f_i at x regarding direction $v \in \mathbb{R}^n$ is given by $\langle \nabla f_i(x), v \rangle = ||\nabla f_i(x)|| \cdot ||v|| \cdot \cos\theta$, and it gets its more negative value when the coplanar angle θ between $\nabla f_i(x)$ and v is Π, *i.e.,* when the vectors point toward opposite directions.

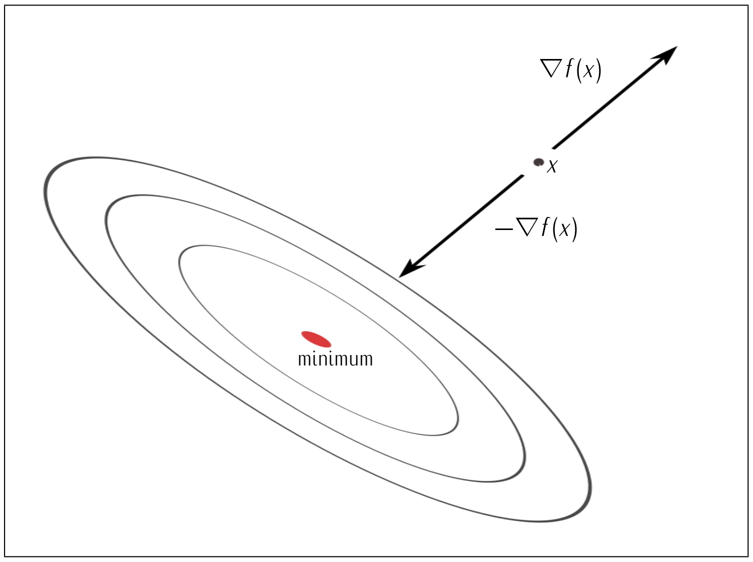


Figure 3.1: The negative of the gradient leads to the maximal decrement direction for a specific function f_i, in a particular point x.

Then, for each $x \in \mathbb{R}^n$, and assuming $\nabla f_i(x) \neq 0$ for $i \in \{1, \dots, k\}$, we define:

$$H_{x,i} = \left\{ v \in \mathbb{R}^n \,:\, \left\langle \nabla f_i(x), v \right\rangle = 0 \right\},$$

$$H^+_{x,i} = \left\{ v \in \mathbb{R}^n \,:\, \left\langle \nabla f_i(x), v \right\rangle \geq 0 \right\},$$

and

$$H^-_{x,i} = \left\{ v \in \mathbb{R}^n \,:\, \left\langle \nabla f_i(x), v \right\rangle \leq 0 \right\}.$$

Since the set $H_{x,i}$ is the orthogonal complement of the vector $\nabla f_i(x)$, it is in general a hyperplane—set of dimension $(n-1)$—of $\mathbb{R}^n$; also, it divides the space in two n–dimensional sets (half spaces) $H^+_{x,i}$ and $H^-_{x,i}$, as it is shown in Figure 3.2).

Definition 3.1.1 *We denote*

$$C_x(-,-,\dots,-) = \bigcap_{i=1}^{k} H^-_{x,i.} \setminus \{0 \,\cup\, \{\cap_{i=1}^{k} H_{x,i}\}\,\}$$

as the descent cone *pointed at x (see Figure 3.3). Similarly, the* ascent cone *is defined as*

$$C_x(+,+,\dots,+) = \bigcap_{i=1}^{k} H^+_{x,i.} \setminus \{0 \,\cup\, \{\cap_{i=1}^{k} H_{x,i}\}\,\},$$

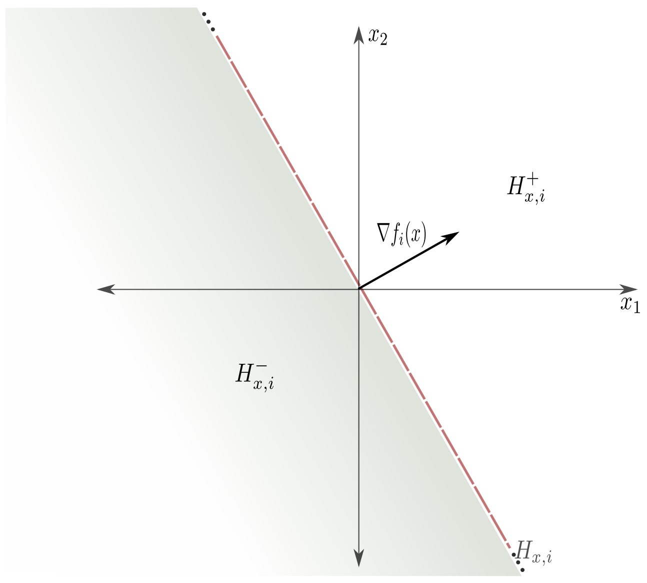


Figure 3.2: This figure shows the space division, into $H^+_{x,i}$ and $H^-_{x,i}$, induced by the gradient of the function f_i at the solution x.

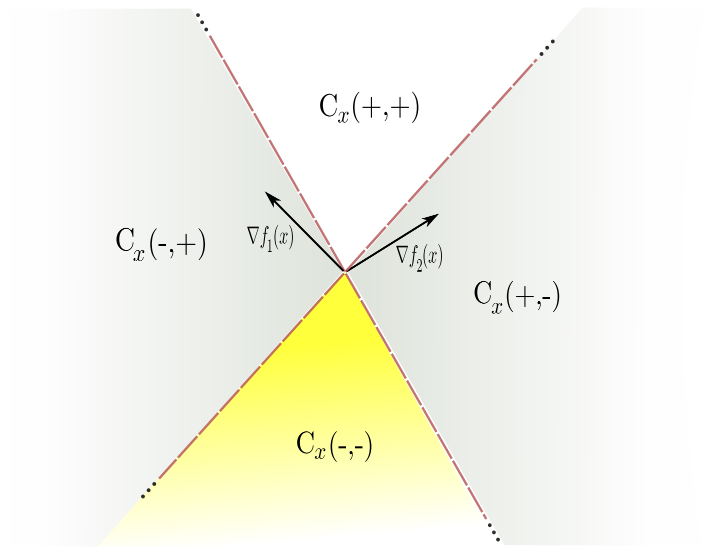


Figure 3.3: This figure shows the ascent cone $C_x(+,+)$, the descent cone $C_x(-,-)$, and the diversity cones $C_x(+,-)$ and $C_x(-,+)$, for a certain point x, on a two-objective problem.

and the diversity cones *are the intersections that contain at least one hyperplane of the form* $H^+_{x,i}$ *and at least one of the form* $H^-_{x,j}$*, for any* $i, j \in \{1, \dots, k\}$.

When having k objective functions of a MOP, each function f_i determines a gradient vector ∇f_i and a hyperplane $H_{x,i}$ for a certain solution x. Then, the presence of these k hyperplanes divides the space $\mathbb{R}^n$ into 2^k, or less, subspaces. Summarizing, for each point x, the search space is divided into one descent cone, one ascent cone and several diversity cones—as they were previously named in [14][2].

The interior set of the descent cone, which is of certain importance in practice, can be expressed as

$$Im(J_{F(x)}) \; \cap \; \mathbb{R}^k_{(-)}, \tag{3.1}$$

where $\mathbb{R}_{(-)}$ is the set of strictly negative real numbers, $Im(J_{F(x)})$ consists of the image of the linear mapping given by $J_{F(x)}$, and $J_{F(x)}$ is the Jacobian matrix of the function F at x described by:

$$J_{F(x)} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}, & \dots, & \frac{\partial f_1}{\partial x_n} \\ \vdots, & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1}, & \dots, & \frac{\partial f_1}{\partial x_n} \end{pmatrix}. \tag{3.2}$$

Descent cones describe the dynamics of the search from a directed-by-gradient point of view. This has several important implications (see for example the results presented in Section 3.5) that are useful when building local search operators for searching algorithms over real numbers. For a particular point x, the descent cone size is determined by the configuration of the objective gradients. To illustrate this, Figure 3.4 shows the different stages of this configuration: when the gradients are mostly aligned, case (e), the size of the cone is almost 50% of the space, being very likely to improve the solution by applying a small perturbation—in this particular stage getting a better point is almost as simple as in the case of minimizing one single function. On the contrary, when the gradients are positioned in opposite directions, case (a), the descent cone size decreases. This latter case occurs, for example, in most of the Pareto set vicinity.

The above analysis could explain why MOEAs have a good performance at the beginning of the search, and a slow convergence rate at latter stages—when points are near to the Pareto front and the chance of generating stochastically better points (near to the parents) is reduced. This observation inspires guidelines for a suitable hybridization; if it is possible to identify when the evolutionary search is no longer producing good results, this is then the time when the gradient-based LS can take part of the process—in order to certainly improve solutions in a deterministic way.

Improving solutions implies performing movements in specific search directions. In MO, as we have noticed, these directions should be able to (at least locally) lead toward better

[2]In [14] the descent cones are defined, for illustrative purposes, by pictures of the corresponding affine hyperplanes described here. We are stating the formal definitions using no affine hyperplanes just to be consistent with our approach.

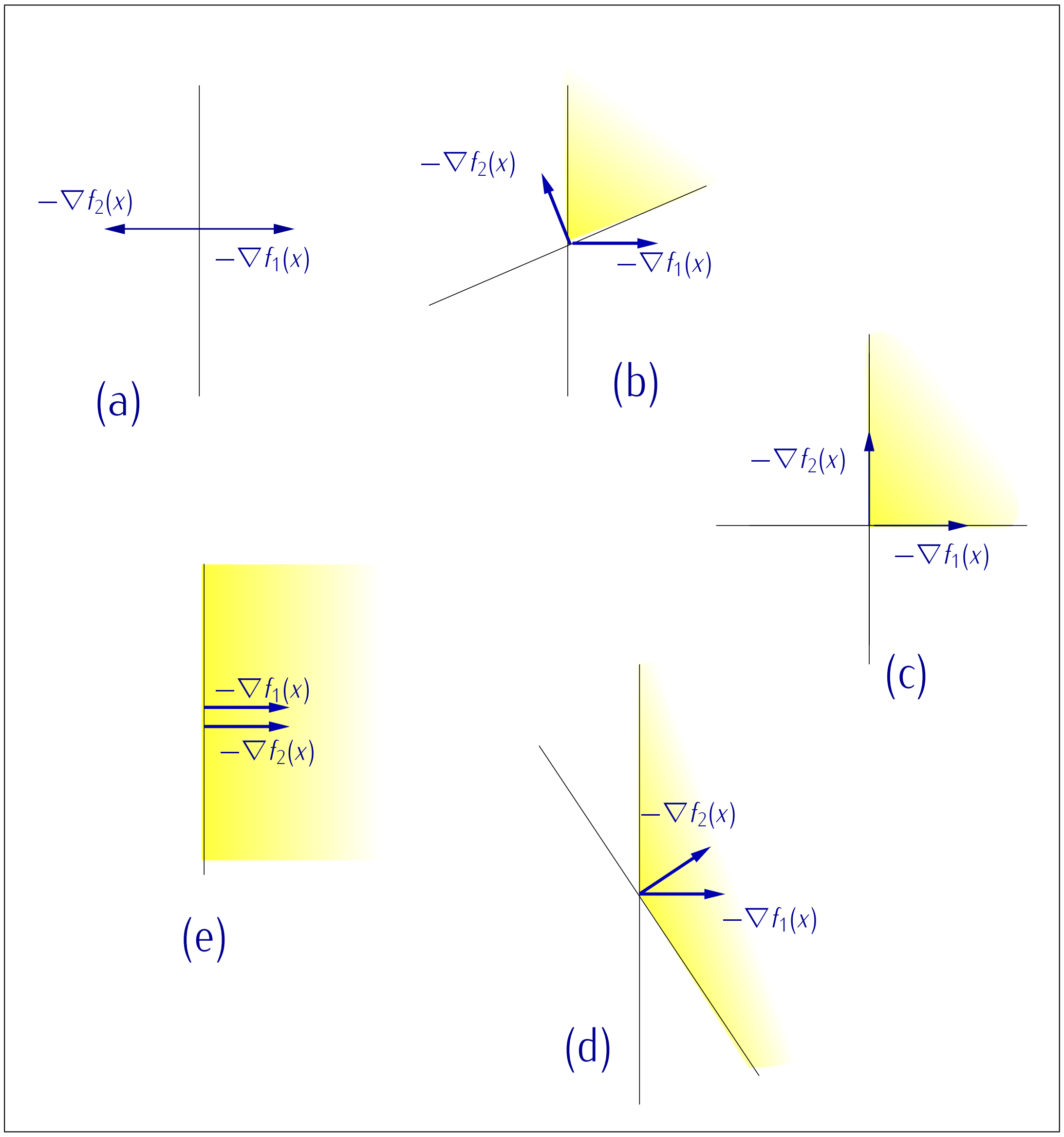


Figure 3.4: This figure shows the dynamics of descent cones induced by the configuration of the gradients, assuming a two-objective problem over $\mathbb{R}^2$.

solutions regarding all the functions simultaneously. *i.e.*, the new point is expected to dominate the original one. This is formally said by the following definition.

Definition 3.1.2 *A vector $v \in \mathbb{R}^n$ is called a multi-objective descent direction of the point $x \in \mathbb{R}^n$ if*

$$v \in C_x(-, -, \dots, -).$$

In other words, a multi-objective descent direction is such that the directional derivatives with respect to v in x are non-positive, *i.e.* $\langle \nabla f_i(x), v \rangle \leq 0$ for all $i \in 1, \dots, k$ without allowing them to be all equal to zero. This means that if we perform a small movement over v, we obtain a local improvement (decrease) simultaneously for all the objective functions. In the following section, we present a result for the construction of a descent direction in the simplest multi-objective case: two objectives.

3.2 A Simple Descent Direction Suitable for Two-objective Problems

The simplest way to combine two gradients, in order to get a common descent direction, is by a vector sum. This fact has been already observed (e.g. [33]) but it has not been exploited in memetic algorithms yet. The following result shows the way this coupling works. Unfortunately, Proposition 3.2.1 cannot be generalized for more than two objective functions—we explain this in detail further in this section; for that case, the application of other approaches (see Section 3.3) is necessary to obtain the descent direction—even when this represents a higher computational cost.

Proposition 3.2.1 *Let $x \in \mathbb{R}^n$, and $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ define a two-objective MOP. If $\nabla f_i(x) \neq 0$, for $i = \{1, 2\}$, then the direction*

$$\overline{\nabla}_x = -\left(\frac{\nabla f_1(x)}{||\nabla f_1(x)||} + \frac{\nabla f_2(x)}{||\nabla f_2(x)||} \right), \tag{3.3}$$

where $||\cdot|| = ||\cdot||_2$, is a descent direction at x for the MOP.

Proof: Let us denote $\nabla_i := \frac{\nabla f_i(x)}{||\nabla f_i(x)||}$ for $i = \{1, 2\}$, and θ be the angle between ∇_1 and ∇_2. Then

$$\begin{aligned} \langle \overline{\nabla}_x, \nabla_1 \rangle &= \langle -(\nabla_1 + \nabla_2), \nabla_1 \rangle \\ &= -1\, (\langle \nabla_1, \nabla_1 \rangle + \langle \nabla_2, \nabla_1 \rangle) \\ &= -1\, (1 + ||\nabla_2||\, ||\nabla_1|| \cos(\theta)) \\ &= -1 - \cos(\theta) \\ &\leq 0. \end{aligned} \tag{3.4}$$

Similarly, $\langle \overline{\nabla}_x, \nabla_2 \rangle \leq 0$; then, $\overline{\nabla}_x$ is a descent direction of the point x for the defined MOP. □

Notice that in case $\nabla f_i(x) = 0$, the point x is a KKT point. It is also important to note that normalizing the gradients is crucial for the result below; see Figure 3.5 for an example.

One of the main issues when using gradient-based tools is how to balance the computational cost of the method with the achieved improvements. The currently available MOEAs that use descent directions as LS engines have two sources of computational cost: one is

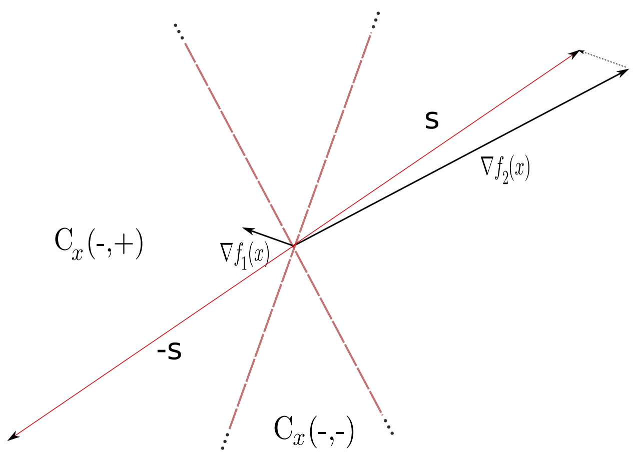


Figure 3.5: This figure illustrates that proposition 3.2.1 does not hold for unnormalized vectors. Here the computed direction $-s$ lies outside the descent cone.

associated to the fitness function evaluations required to estimate the gradients and to perform the line search. The second source is related to the computation of the descent direction itself. This proposal has the advantage of having no cost for the computation of the descent directions—apart from the cost of approximating the gradients. We claim that this procedure is the simplest way to combine the gradients of two functions; but, it can not be generalized to more than two functions since the arithmetic combination of them does not produce descent directions in general. This is shown in the next example:

Example 3.2.1 *Assuming a three-objective MOP such that, for a certain x,*

$$\begin{aligned}\nabla f_1(x) &= (1.000, 1.000, 1.000)\\ \nabla f_2(x) &= (-0.944, 0.970, 0.374)\\ \nabla f_3(x) &= (0.836, -0.177, -0.334),\end{aligned}$$

computing

$$\begin{aligned}\overline{\nabla}_x &= -\left(\frac{\nabla f_1(x)}{||\nabla f_1(x)||}+\frac{\nabla f_2(x)}{||\nabla f_2(x)||}+\frac{\nabla f_3(x)}{||\nabla f_3(x)||}\right) &\boxed{3.5}\\ &= (-0.3826, -0.5262, -0.3730) &\boxed{3.6}\end{aligned}$$

leads to

$$\langle\overline{\nabla}_x, \nabla_1\rangle = -1.2818 < 0.$$

$$\langle \overline{\nabla}_x, \nabla_2 \rangle = 0.4423 > 0.$$
$$\langle \overline{\nabla}_x, \nabla_3 \rangle = -0.2889 < 0.$$

with $\nabla_i := \frac{\nabla f_i(x)}{||\nabla f_i(x)||}$ *for* $i = \{1, 2, 3\}$. *Then* $\overline{\nabla}_x$ *is not a common descent direction.*

Some proposals have recently been made to compute multi-objective descent directions. The greedy approaches [42] and [107] are successful but could lead to bias in presence of unbalanced gradient magnitudes. Since computing a gradient-based descent directions for a MOP leads again to solving a MOP, unbiased computations have been presented based on the calculation of the whole Pareto set of descent directions [8][53]. Both approaches have advantages and drawbacks that are worth studying. All these methods use first order gradient information and require solving a linear optimization problem for each computed direction. The following section covers this subject in detail.

3.3 Current Approaches

When using gradient information within a MOEA, the final hybrid algorithm is able to perform directed accurate movements, towards an improved solution, using multi-objective descent search directions (as it was shown in [42] [107] [53] [8]). Computing such directions is also a MOP [8]. Since each objective provides its own (gradient-based) range of movements for descent, all of these possible directions need to be properly combined into a single one in order to guide a MOEA. The most commonly used proposals for computing multi-objective descent directions are the one from Fliege [42] and the one from Schäffler *et al.* [107]. These methods return a common descent direction for all the objectives after solving a linear optimization problem. These approaches have been already incorporated into multi-objective memetic strategies [129], [83], [86]. Their main drawback is that, being greedy methods used to perform descents, they can present a bias in case of unbalanced magnitudes of the gradients. However, these methods present some advantages, like having an intrinsic stopping criterion and being effective in practice.

As an unbiased alternative, it is possible to use normalized gradients and work with the proposals developed by Bosman and DeJong [8], and by Harada *et al.* [53]. These methods look into the whole Pareto set of descent directions and choose (randomly) one direction within it. The cost of this approach is again related to solving a system of linear equations.

Schäffler, Schultz and Weinzierl's Direction

This method (SSW), presented in [107], consists of two components. The first part generates a better individual, calculated in a deterministic way; while the second part adds a stochastic component to move the individuals along the entire Pareto front, in a certain sense. Some parts of this method have been naively used [130], to replace the mutation operator of a MOEA, generating in this way a new better individual from one element of the population.

In this work, we also pay attention to the first part of the method, in order to generate a new individual—which will be better than another individual from the population. Our idea is that the stochastic component of the process will be fulfiled by the MOEA itself which we are hybridizing. This mentioned method uses the following theorem to obtain a descent direction:

Theorem 3.3.1 *Given a MOP (F) as in (2.1), in page 7, and* $q : \mathbb{R}^n \to \mathbb{R}^n$ *be defined by*

$$q(x) = \sum_{i=1}^{k} \hat{\alpha}_i \nabla f_i(x), \tag{3.7}$$

where $\hat{\alpha}$ *is a solution of*

$$\min_{\alpha \in \mathbb{R}^k} \left\{ || \sum_{i=1}^{k} \alpha_i \nabla f_i(x) ||_2^2 ; \alpha_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \alpha_i = 1 \right\}, \tag{3.8}$$

then either $q(x) = 0$*, or it is the case that* $-q(x)$ *is a descent direction for* F *at* x*.*

Fliege's Approach

Let $J_{F(x)}$ be the Jacobian matrix of F and $\mathbb{R}_{(-)}$ the set of negative real numbers. Then, a necessary condition [42] for the point $x \in \mathbb{R}^n$ to be a local Pareto point is that the following equality

$$\text{range}(J_{F(x)}) \cap \mathbb{R}^m_{(-)} = \emptyset \tag{3.9}$$

holds. The above expression means that if we have a point x which is not dominated by any other point in a certain neighborhood, in a Pareto sense, then it is not possible that a direction v exists, for which the directional derivative of each f_i could be negative—which would turn it into a descent direction.

Setting the above in terms of our purposes, we will assume that a solution x should be improved during the solution of a real-valued minimization MOP. Then, if x is not a critical point for the MOP, it is possible to choose a descent direction v which fulfills

$$J_{F(x)} v \in \mathbb{R}^m_{(-)}.$$

If this is the case, v can be computed using the information provided by the Jacobian of the problem evaluated on x. As Fliege presents in [42], it is necessary to solve the following quadratic programming problem:

$$\text{Minimize} \quad \alpha + \frac{1}{2}||v||^2 \tag{3.10}$$

$$\text{subject to} \quad (J_{F(x)} v)_i \leq \alpha, \quad \text{for all} \quad i \in \{1, \ldots, m\}.$$

The above problem produces a solution

$$(v^*, \alpha^*), \tag{3.11}$$

where the descent direction that we are looking for is v^*. After having[3] $x^1 = x + tv$, we are in condition to repeat the movement by calculating a new descent direction from x^1 or, if this is not possible, we can assume that a critical point has been achieved.

This method, constructed by Fliege, automatically triggers a condition to know if x^1 fulfills condition (3.9), which is the case when $\alpha^* = 0$. In practice, it is necessary to set a tolerance parameter τ, $\tau < 0$ to stop the descent when $\tau \leq \alpha^*$ holds; note that by construction $\alpha^* \leq 0$.

Several ways to calculate descent directions have been proposed [107, 8, 9, 14, 53]. A study of the efficiency of each method is subject of ongoing research. But, descent directions are not the only interesting directions during the search; sometimes it is also necessary to perform movements along the Pareto front, or specifically directed towards a particular region. We will discuss this idea in the following chapters.

3.4 Optimality Conditions

The optimality conditions to decide whether a point could be a Pareto point, of a MOP, have been commonly studied since their introduction in 1951 [81]. In case all the objectives of the MOP are differentiable, the following famous theorem of Kuhn and Tucker [80] states a necessary condition for Pareto optimality for unconstrained MOPs.

Theorem 3.4.1 *Let x^* be a Pareto point of (MOP), then there exists a vector $\alpha \in \mathbb{R}^k$ with $\alpha_i \geq 0, i = 1, \ldots, k$, and $\sum_{i=1}^k \alpha_i = 1$ such that*

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(x^*) = 0. \tag{3.12}$$

According to this theorem, the vector of zeros can be written as a convex combination of the gradients of the objectives at every Pareto point. It is worth to notice that holding Condition (3.12) is not sufficient for claiming Pareto optimality; but, those points satisfying (3.12) are certainly 'Pareto candidates'.

Definition 3.4.1 *A point $x \in \mathbb{R}^n$ is called a* Karush–Kuhn–Tucker point[4] *(KKT–point) if there exist scalars $\alpha_1, \ldots, \alpha_k \geq 0$ such that $\sum_{i=1}^k \alpha_i = 1$ and that equation (3.12) is satisfied.*

[3]With t a suitable step length.

[4]Named after the works of Karush [73] and Kuhn & Tucker [80].

3.5 Descent Cones for MOPs with $K > 3$

This section is included in order to study, from the descent-cones point of view, the role of increasing the number of objectives of a MOP. The term *many-objective problem* [38] refers to a MOP with more than three objective functions. The special interest for investigating Many-objective problems comes from observations about degradation of MOEAs performance when dealing with them.

The following comments are part of the work described in [121] where the influence of the number of objectives is investigated to understand if this is the mere cause of degradation in MOEAs performance for many-objective problems. The main conclusion in that work is that the addition of an objective does not make a problem *per se* harder. In that study, we investigate the influence of the number k of objectives in a MOP on the hardness of the problem when solving it by evolution strategies. For this, we have utilized the descent cones which can be used to measure the probability to improve a solution by the generational operators. Though these considerations are of qualitative nature and can hardly be quantified, they help to a certain extent to understand the behavior of the population's evolution with respect to k. As an example, we have considered a class of uni-modal test functions and have investigated the resulting models qualitatively and empirically. Qualitative studies based on the descent cones led to the conclusion that, on the one hand, the addition of an objective makes the problem indeed harder, but, on the other hand, it can be argued that the difference is not significant, which is, later on, empirically validated. That is, it can be argued that the addition of an objective to a MOP does not make the problem per se harder. In contrast to this, many researchers have so far observed a certain scalability in the hardness of the problem with respect to k, albeit for more complex models. Based on our considerations on the uni-modal models we have tried to identify the challenges which have to be mastered by evolution strategies for general models: the ability to keep "good" solutions in order to pull the population toward the set of interest, the probability to improve an individual, and the multi-modality of the MOP. This together with the qualitative discussions included in [121] can be used to a certain extent to explain recent advances in the field of evolutionary many-objective optimization. The importance of such analysis lies on the possibility that those insights into the geometry of MOPs may help researchers in the field of evolutionary computation for further developments of efficient specialized algorithms, particularly when dealing with many-objective problems.

"Turn away with fear and horror from this lamentable plague of continuous functions that do not have a derivative."

Charles Hermite

4 Gradient-based Local Search for MOPs

4.1 Local Search and Search Directions for MOPs

When used in optimization processes, LS explores the solution space in a fine-grained way. The traditional tool to perform LS for continuous search spaces is the gradient of the objective function (see Figure 3.1). There are several well-known methods available for single-objective optimization, which have been successfully used during the last decades [96, 138, 101]. As we have noticed before, in the presence of several objective functions we have to deal with several gradients—one for each objective. Some gradient-based hybrid MOEAs have been built to perform descent movements alternating the single gradient of each objective function [44, 56], and this could have certain application when preferences management is required. However, when seeking for a general purpose local searcher, for continuous multi-objective problems, our focus lies on using simultaneously all the gradients. In the case of looking for better solutions in the Pareto sense (*i.e.* $x_{i+1} \prec x_i$) descent directions, as described in Chapter 3, are suitable. Improving solutions by descent directions, in LS, has two advantages in the evolutionary multi-objective context: it is a reliable way to refine the final results when used at the end of the search; and second, the possibility of leading the search with "better genes" in order to "pull" the population toward the optimum in a faster manner.

Descent directions are not the only useful movement directions for multi-objective search. Moving along the Pareto set is also possible using gradient-based continuation methods, such as those in [58] [119] [52], or those where estimation of the gradient is not necessary, as in [86]. Furthermore, it is possible to perform directed movements not only towards and along the Pareto front, but also to any desired direction in the objective space. To summarize, alternative search directions are advisable because in multi-objective search the term 'promising region' depends on the goal at the time, *i.e.,* in some moments the proximity to the front is necessary but in others a movement to spread the solutions is advisable.

In the rest of this chapter, we look deeply into the calculation of multi-objective gradient-based search directions. The study of this subject is necessary when hybridizing MOEAs with LS. We analyze the cost and differences of our procedures, and study some aspects regarding the parameters for its practical implementation. We also show some examples to point out the use of these strategies, and present some experimental results for comparison. Some considerations to take into account when hybridizing MOEAs with these operators are presented in Chapter 5.

In the following, we present our different proposals in order to build local searchers with alternative multi-objective search directions: The *Hill Climber with Side-step* in Section 4.2, which combines two kind of movements, and the recently proposed *Directed Search Method* in Section 4.3 (page 52).

4.2 The Hill Climber with Sidestep Operator

The *Hill Climber with Side Step* (HCS) was proposed in [86] as a novel point-wise iterative search procedure, which is designed to perform LS in a given MOP. Being a local searcher, the HCS is intended to be capable of moving both toward and along the set of Pareto points,[1] depending on the location of the current iterate. The HCS operator has an automatic criterion (based on the descent cone size and the KKT conditions) to switch between the two types of movements. Two variants of the HCS have been proposed: A gradient-free version denoted here as HCS1—also presented as an early version in [126]—and one, denoted here as HCS2, that exploits (first and second order) gradient information. Both can be used as standalone algorithms to explore parts of the Pareto set, starting with one single solution, and both are able to handle constraints of the model to some extent. In the following we will explain the two versions of the HCS as standalone algorithms.

4.2.1 HCS1 procedure

The HCS1 assumes that the objective gradients are practically aligned, in particular, when the initial point is far away from the optima. In this case the descent cone is almost equal to the half-spaces associated with each objective;[2] we can say that the descent cone is 'big', since the chance of randomly generating a direction/solution which simultaneously improves all the functions is high (see Section 3.1, page 21). Figure 4.1 shows randomly generated points in these two cases; it illustrates this observation to contrast with the situation when the point x is near to some regions of the Pareto set. In this second case, the gradients are almost linearly dependent, which means that the descent cone shrinks down; then, the probability of randomly generating a better point (*i.e.,* a point inside the descent cone) is low.

According to the comments above, the HCS1 starts with an initial point x_0, and the next iterate x_1 is randomly chosen from a vicinity $B(x_0, r)$ with radius r.[3] If $x_1 \prec x_0$ the movement direction for improvement is set as $v = x_1 - x_0$; if $x_0 \prec x_1$, then the direction is flipped, *i.e.,* $v = x_0 - x_1$.

When a solution is near to a Pareto point, the probability of generating a dominated or a dominating point—like in the case above—is low (see Figure 4.1). So, when $\tilde{x}_1$ is not comparable against x_0, the point is stored, and labeled, as a point which corresponds

[1]Due to the nature of gradients, we can only guarantee local optimality.
[2]This observation was first noticed in [15].
[3]$B(x_0, r) := \{x \in \mathbb{R}^n : x_{0,i} - r_i \leq x_i \leq x_{0,i} + r_i,\ \forall i = 1, .., n\}$.

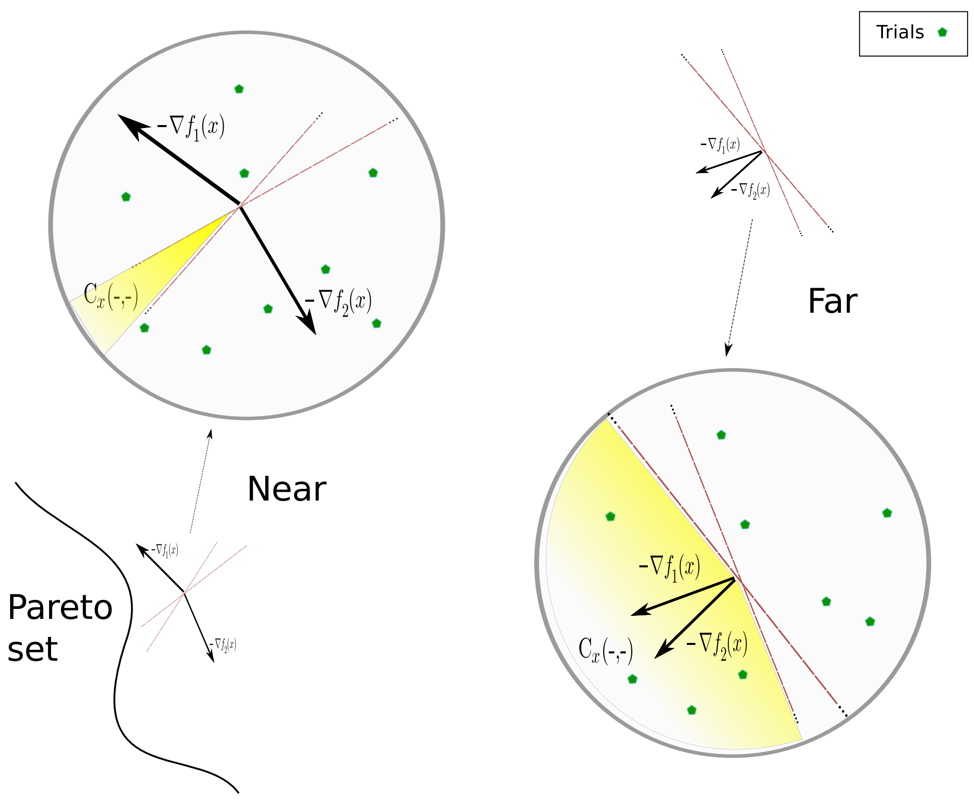


Figure 4.1: The configurations for descent cones at points near and far from the Pareto set are shown in this figure. The trial points illustrate how likely is to reach descent cones according to the proximity to the optima. For illustration purposes, the affine hyperplanes and the translated cones at the solution point are shown.

to a specific diversity cone; then, a new trial point is generated. After N_{nd} trials obtaining mutually non-dominated solutions, the proximity to the optima is assumed and this triggers a 'side step' movement over the Pareto front.

To perform this side step movement, the stored points $\tilde{x}_1, \ldots \tilde{x}_{N_{nd}}$ are used in the following way: Assuming a two-objective problem, for example, if $\tilde{x}_1 - x_0$ is in the cone $C(+,-)$, then $x_0 - \tilde{x}_1$ is in the opposite cone $C(-,+)$ (for the two-objective MOPs, the general k-objective case is analogue). When the limit for unsuccessful trials is reached, a search along $C(-,+)$ is performed; taking advantage of the accumulated information, the following direction is used:

$$v_{acc} = \frac{1}{N_{nd}} \sum_{i=1}^{N_{nd}} s_i \frac{\tilde{x}_i - x_0}{||\tilde{x}_i - x_0||}, \tag{4.1}$$

where

$$s_i = \begin{cases} 1 & \text{if } f_1(\tilde{x}_i) < f_1(x_0). \\ \text{-1} & \text{otherwise.} \end{cases} \tag{4.2}$$

By construction, v_{acc} is in $C(-,+)$, and by averaging the trial search directions we aim to obtain a direction[4] which is ideally 'perpendicular' to the (small) descent cone. Note that in this case v_{acc} is indeed a 'side step' to the upward movement of the hill climbing process as desired, but this search direction does not necessarily have to point along the Pareto set (we present with HCS2 an alternative method with better guidance properties). Also, there is no guarantee that v_{acc} indeed points to a diversity cone, but even if that is not the case, there will be an improvement on the solution, anyway. This means that, even with these two considerations, this side step is still a good option in practice, when working with few objectives or when coupling the operator with evolutionary methods. For more than two objectives the strategy is similar, but we should notice that the possibilities for the side direction increase. For example, for $k = 3$ there are six diversity cones which form three groups by reflection as follows:

$$\begin{array}{lcl} C(+,-,-) & \text{and} & C(-,+,+), \\ C(+,-,+) & \text{and} & C(-,+,-), \\ C(+,+,-) & \text{and} & C(-,+,+). \end{array} \tag{4.3}$$

For a general k it is possible to make $2^{k-1} - 1$ different groups, being less likely to find a perpendicular direction, due to averaging N_{nd} trials, within one of these cones. Alternatively, one can *e.g.,* use the accumulated information by taking the average search direction over *all* search directions only using Formula (4.1) with $s = 1$.

Algorithm 4 shows the pseudocode of the HCS1 algorithm as a standalone process. The value of i_0, chosen at random for simplicity, determines the side step direction (see line 5 and lines 15-20). For illustration purposes, we explain here the two-objective case: in order

[4]This direction has previously been proposed as a local guide for a multi-objective particle swarm algorithm in [13].

```
Require: starting point $x_0 \in Q$, radius $r \in \mathbb{R}^n_+$, number $N_{nd}$ of trials, MOP with $k = 2$
Ensure: sequence $\{x_l\}_{l \in \mathbb{N}}$ of candidate solutions
 1: $a := (0, \dots, 0) \in \mathbb{R}^n$
 2: $nondom := 0$
 3: for $l = 1, 2, \dots$ do
 4:     set $x_l^1 := x_{l-1}^b$ and choose $x_l^2 \in B(x_l^1, r)$ at random
 5:     choose $i_0 \in \{1, 2\}$ at random
 6:     if $x_l^1 \prec x_l^2$ then
 7:         $v_l := x_l^2 - x_l^1$
 8:         compute $t_l \in \mathbb{R}_+$ and set $\tilde{x}_l^n := x_l^2 + t_l v_l$.
 9:         choose $x_l^b \in \{\tilde{x}_l^b, x_l^1\}$ such that $f(x_l^b) = \min(f(\tilde{x}_l^n), f(x_l^1))$
10:         $nondom := 0, \quad a := (0, \dots, 0)$
11:     else if $x_l^2 \prec x_l^1$ then
12:         proceed analogous to case "$x_l^1 \prec x_l^2$" with
13:         $v_l := x_l^1 - x_l^2$ and $\tilde{x}_l^n := x_l^1 + t_l v_l$.
14:     else
15:         if $f_{i_0}(x_l^2) < f_{i_0}(x_l^1)$ then
16:             $s_l := 1$
17:         else
18:             $s_l := -1$
19:         end if
20:         $a := a + \frac{s_l}{N_{nd}} \frac{x_l^2 - x_l^1}{||x_l^2 - x_l^1||}$
21:         $nondom := nondom + 1$
22:         if $nondom = N_{nd}$ then
23:             compute $\tilde{t}_l \in \mathbb{R}_+$ and set $\tilde{x}_l^n := x_l^1 + \tilde{t}_l a$.
24:             $nondom := 0, \quad a := (0, \dots, 0)$
25:         end if
26:     end if
27: end for
```

Algorithm 4: HCS1

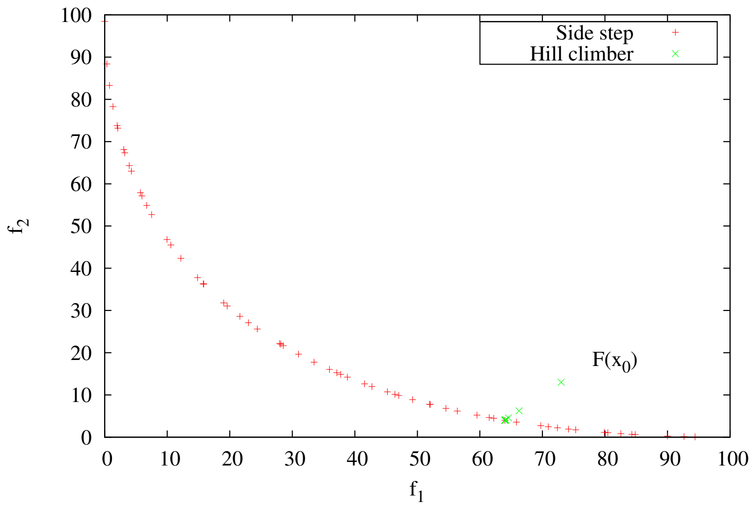


Figure 4.2: This figure shows the performance of HCS as a standalone algorithm on Example 2.1.1. The 'anchor' picture shows an approximation of the Pareto front, built using the steering mechanism previously described.

to introduce an orientation to the search, i_0 is fixed to 1 for the initial iteration steps. When the side step (line 23) has been performed N_s times during the run of an algorithm, this indicates that the current iteration is already near to the (local) Pareto set, and this vector is stored in x_p. If no improvements can be achieved, according to f_1, within a given number N_i of side steps, the HCS 'jumps' back to x_p, and a similar process is started but aiming for improvements according to f_2. That is, i_0 is set to -1 for the following steps (see Figure 4.2). When no improvements can be achieved according to f_2 within another N_i side steps along $C(+,-)$, the process is stopped. It is worth to notice that more attention has to be paid for the computation of t_l (see Section 4.2.3, page 41). For the experiments presented here, a strategy analogous to [34] was used.

Finally, we notice that some parameters are involved in the realization of Algorithm 4; however, only the values for four design parameters have to be chosen—they are shown in Table 4.1. The parameter r defines the neighborhood search of the procedure. This neighborhood is used to find a search direction which is afterwards coupled with a step size control. Because of this, the value of r is not that important, but it should be 'small' to guarantee the locality of the search. N_{nd} is the value which determines the number of directions that have to be averaged, in order to choose the side step direction. In general, a larger value of N_{nd} leads to a 'better' side step (in the sense that the search is performed orthogonal to the upward movement), but will in turn increase the cost of the search. We have experienced that a low value for N_{nd}, in particular $5 \leq N_{nd} \leq 10$, already gives satisfactory results; the 'accuracy' of the search does not seem to influence the performance of the HCS (considering that neither first or second order information is used). The value of ϵ_y is problem

Table 4.1: Design parameters that are required for the realization of HCS1.

Parameter	**Description**
r	Radius for neighborhood search (Algorithm 4).
N_{nd}	Number of trials for the hill climber before the side step is performed (Algorithm 4).
ϵ_y	Desired distance (in image space) for the side step, see Eq. (4.9).
tol	Tolerance value used for the backtracking in Algorithm 7.

dependent but can be given quite easily in a real world application (see discussion above the equation (4.9)). Finally, the tolerance value tol is also problem dependent, and has to be adjusted for constrained MOPs, like in every other algorithm which deals with constraints (see Section 4.2.4, page 45, for the explanation on constraint management).

4.2.2 HCS2 procedure

When gradient information is explicitly available, the search directions for the HCS can be computed more precisely. We describe in the following, a possible realization of the HCS using the descent direction presented in [107]. First, using Theorem 3.3.1, a descent direction $-q(x)$ is computed. If $q(x) = 0$, then the point x should be a KKT–point. This implies that a test for optimality is automatically performed when computing the descent direction for a given point $x \in \mathcal{X}$. In other words, for every point $x \in \mathcal{X}$, which is not a KKT–point, a descent direction can be found by obtaining the vector $\hat{\alpha}$ after solving the quadratic optimization problem (3.8).

For the numerical treatment of this optimality test, we suggest to set a threshold $\epsilon_{\mathcal{P}} \in \mathbb{R}_+$ such that in case

$$||\sum_{i=1}^{k} \hat{\alpha}_i \nabla f_i(x)||_2^2 \geq \epsilon_{\mathcal{P}}, \tag{4.4}$$

the candidate solution x can be considered to be 'away' from $\mathcal{P}$, and thus, it makes sense to seek for a dominating solution. In this case, the descent direction (3.7) can be taken together with a suitable step size control (see Section 4.2.3). If the value of the term in (4.4) is less than $\epsilon_{\mathcal{P}}$, this indicates that x is already in a certain vicinity of $\mathcal{P}$. In that case one can lean elements from (multi-objective) continuation methods [58, 3] to perform a search along $\mathcal{P}$.

To explain the side step procedure we assume, for simplicity, that a KKT–point $\hat{x}$ is given with its corresponding weight vector $\hat{\alpha}$, obtained by solving (3.8). Then the point $(\hat{x}, \hat{\alpha}) \in \mathbb{R}^{n+k}$ is clearly contained in the zero set of the auxiliary function $\tilde{F} : \mathbb{R}^{n+k} \to \mathbb{R}^{n+1}$

Table 4.2: Design parameters that are required for the realization of the HCS algorithm which involves gradient information (HCS2).

Parameter	Description
ϵ_y	Desired distance (in image space) for the side step (4.9)
tol	Tolerance value used for the backtracking, see algorithm 7.
$\epsilon_{\mathcal{P}}$	Threshold for the vicinity test (4.4)

of the given MOP, which is defined as follows:

$$\tilde{F}(x, \alpha) = \begin{pmatrix} \sum_{i=1}^{k} \alpha_i \nabla f_i(x) \\ \sum_{i=1}^{k} \alpha_i - 1 \end{pmatrix}. \tag{4.5}$$

It has been shown in [58], that the zero set $\tilde{F}^{-1}(0)$ can be linearized around $\hat{x}$ by using a QU-factorization of $\tilde{F}'(\hat{x}, \hat{\alpha})^T$, *i.e.*, the transposed of the Jacobian matrix of $\tilde{F}$ at $(\hat{x}, \hat{\alpha})$. To be more precise, given a factorization

$$J_{\tilde{F}}(\hat{x}, \hat{\alpha})^T = QU \in \mathbb{R}^{(n+k)\times(n+k)}, \tag{4.6}$$

where $Q = (Q_N, Q_K) \in \mathbb{R}^{(n+k)\times(n+k)}$ is orthogonal with $Q_N \in \mathbb{R}^{(n+k)\times(n+1)}$ and $Q_K \in \mathbb{R}^{(n+k)\times(k-1)}$, the column vectors of Q_K form—under some mild regularity assumptions on $\tilde{F}^{-1}(0)$ at $(\hat{x}, \hat{\alpha})$, see [58]—an orthonormal basis of the tangent space of $\tilde{F}^{-1}(0)$. Hence, it can be expected that each column vector $q_i \in Q_K$, $i = 1, \dots, k-1$, points (locally) along $\mathcal{P}$ and it is, then, well suited for a side step direction.

Algorithm 5 presents a procedure which is based on the above discussion. Note that this is one possible realization and that there exist certainly other possible ways leading, however, to similar results; for instance, alternatively to the descent direction used in Algorithm 5 the ones proposed in [42] and [10] can be taken. Besides, the movement along $\mathcal{P}$ can be realized by predictor-corrector methods [58, 3] which consists roughly speaking of a repeated application of a predictor step obtained by a linearization of $\tilde{F}^{-1}(0)$ and a corrector step which is done via a Gauss-Newton method. Regarding the design parameters of the algorithm, ϵ_y and tol are as discussed above and N_{nd} and r are not needed due to the accuracy of the gradient-based search; also, the threshold $\epsilon_{\mathcal{P}}$ is used for the vicinity test of a given $\mathcal{P}$. This value is certainly problem dependent, but can be made 'small' due to the convergence properties of the hill climber (*e.g.,* [42]).

```
Require: starting point $x_0 \in Q$
Ensure: sequence $\{x_l\}_{l\in\mathbb{N}}$ of candidate solutions
 1: for $l = 0, 1, 2, \ldots$ do
 2:     compute the solution $\hat{\alpha}$ of (3.8) for $x_l$.
 3:     if $||\sum_{i=1}^{k} \hat{\alpha}_i \nabla f_i(x_l)||_2^2 \geq \epsilon_{\mathcal{P}}$ then
 4:         $v_l := -q(x_l)$
 5:         compute $t_l \in \mathbb{R}_+$ and set $x_{l+1} := x_l + t_l v_l$
 6:     else
 7:         compute $\tilde{F}'(\hat{x}, \hat{\alpha})^T = (Q_N, Q_K)U$ as in (4.6)
 8:         choose a column vector $\tilde{q} \in Q_K$ at random
 9:         compute $\tilde{t}_l \in \mathbb{R}_+$ and set $x_{l+1} := x_l + \tilde{t}_l \tilde{q}$.
10:     end if
11: end for
```

Algorithm 5: HCS2

4.2.3 Step size calculation

Once the movement direction is set, the next step is to determine the step length to advance in this direction. Here we describe the procedure to compute the step sizes of the HCS for both, the hill climber and the side step movement. This procedure is applied for the two presented versions of the HCS[5].

Step size for movements toward the front (hill climber)

When starting the movement we assume having a descent direction v; or alternatively, that two points are given, namely $x_0, x_1 \in \mathbb{R}^n$, such that $x_1 \prec x_0$. In this latter case, there exists a subsequence $\{i_1, \ldots, i_l\} \subset \{1, \ldots, k\}$ with

$$f_{i_j}(x_1) < f_{i_j}(x_0), \quad j = 1, \ldots, l,$$

and thus, $v := x_1 - x_0$ is a descent direction for all f_{i_j}'s at the point x_0.

Some strategies can be used to perform the line search (see, *e.g.,* [34, 127]). Nevertheless, one crucial problem is to find a good initial value t^* for a suitable step size (which is, for example, given by 1 when using Newton's method); in case t^* is not already sufficient, the step size can for instance be fine tuned by backtracking methods [34]. In this particular instance, since x_1 can be very close to x_0, the distance $||x_1 - x_0||$ may not always serve as a good choice; and standard methods to obtain the initial guess do not apply. For this purpose, we propose the following heuristic to compute t^* instead. We begin explaining the

[5]In case of the version with explicit gradients, the step size for the hill climber is easily computed applying as suggested in [42], which ensures convergence of the procedure, but this is not the most efficient choice in practice, because it requires too many evaluations.

common scalar case, *i.e.*, when a function $f : \mathbb{R} \to \mathbb{R}$, and values $t_0, t_1 \in \mathbb{R}$ with $t_0 < t_1$ and $f(t_0) < f(t_1)$ are given. We first define

$$\Delta := t_1 - t_0,$$

$$t_l := t_0,$$
$$t_m := t_1,$$

and

$$t_r := t_0 + 2\Delta$$

in order to check if

$$\frac{f(t_m) - f(t_l)}{t_m - t_l} < \frac{f(t_r) - f(t_m)}{t_r - t_m} \tag{4.7}$$

holds. When the above equation is true, f is approximated by a quadratic polynomial[6] $p(t) = at^2 + bt + c$; where the values of a, b, and c can be explicitly derived by the interpolation conditions

$$p(t_l) = f(t_l), \quad p(t_m) = f(t_m), \quad \text{and} \quad p(t_r) = f(t_r),$$

(see [34]). The expression (4.7) tests the convexity of the function p; this is, if the term in (4.7) is true then p is convex (see Figure 4.3 for an example), and thus, it is guaranteed that the extreme point of p, obtained by

$$t_p^* = -\frac{b}{2a},$$

is a minimizer and takes its value in (t_0, ∞). Hence, t_p^* can be chosen as a guess for the minimizer of f. On the other hand, if (4.7) is false, this quadratic approximation may not yield a useful result—in fact, in that case t_p^* may be negative—and we suggest to check condition (4.7) with the new data

$$t_l := t_m, \quad t_m := t_r,$$

and

$$t_r := t_l + t_l + 4\Delta$$

i.e., doubling the step size for t_r. This process will be repeated until the boundary of the domain ∂Q is reached (in that case, take the maximal step size t_{max} as described below) or (4.7) is true. The process will stop after a few iterations. If t_p^* is too large (*i.e.*, if $f(t_p^*) > f(t_1)$), smaller step sizes can be found via backtracking.

The use of this idea for the multi-objective case is presented in Algorithm 6. Hereby, $f_{v,i}$ denotes the restriction of objective f_i to the line $x_0 + \mathbb{R}v$, *i.e.*,

$$f_{v,i}(t) = f_i(x_0 + tv), \tag{4.8}$$

[6]The idea to approximate f locally by a quadratic polynomial was first proposed by Armijo in [4].

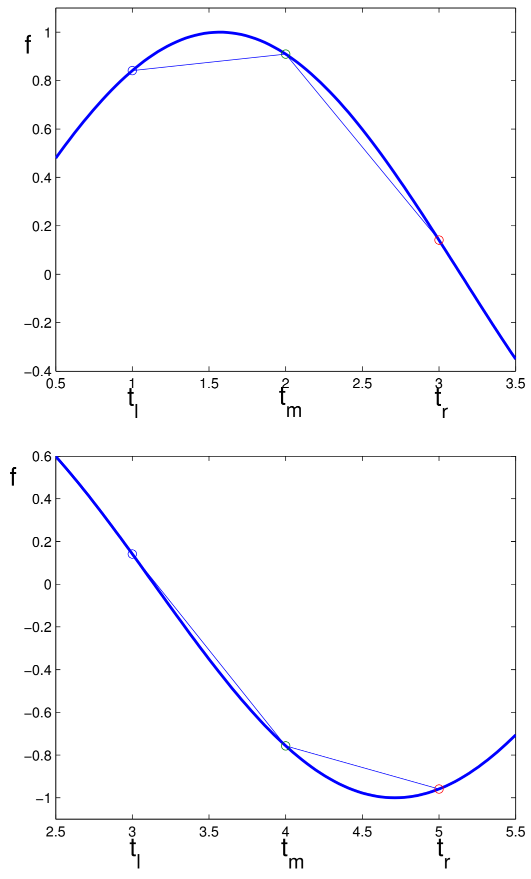


Figure 4.3: The two cases in the quadratic approximation for computing the step length. The expression in Equation (4.7) is false for t_l, t_m, and t_r in the upper figure and true in the lower figure. In the latter case, the quadratic polynomial is convex.

and *quad_approx* is the method to find the minimizer of the quadratic polynomial, as it was described above. Note that this step size control differs from the one presented in [126] since the initial guesses, as they are described in [126], are restricted to the range

$$t^* \in (\ 0\ ,\ 2||x_1 - x_0||_2\ ],$$

which may be too small if x_1 is near to x_0.

Step size for movements along the front (side step)

For this case, we assume to have a point $x_0 \in \mathbb{R}^n$ and a search direction a given by

$$a = \sum_{i=1}^{N_{nd}} \frac{s_i(\tilde{x}_i - x_0)}{||\tilde{x}_i - x_0||}$$

```
Require: $x_0, x_1 \in \mathbb{R}^n$ with $x_1 \prec x_0$, maximal number of trials $N_{max}$
Ensure: step size $t^*$ for the hill climber
 1: $I := \{i \in \{1, \dots, k\} : f_i(x_1) < f_i(x_0)\}$
 2: $v := x_1 - x_0$
 3: $\Delta := ||x_1 - x_0||_2$
 4: $t_l := 0$, $t_m := \Delta$, $t_r := 2\Delta$
 5: for $j = 1, \dots, N_{max}$ do
 6:     if $\exists i \in I$ : (4.7) is true for $t_l, t_m, t_r$ and $f_{v,i}$ then
 7:         for all $i \in I$ do
 8:             if (4.7) is true for $t_l, t_m, t_r$ and $f_{v,i}$ then
 9:                 $t_i^* := quad\_approx(t_l, t_m, t_r, f_{v,i})$
10:             else
11:                 $t_i^* := \infty$
12:             end if
13:         end for
14:         return $t^* := \min_{i=1,\dots,k} t_i^*$
15:     else
16:         $t_l := t_m$, $t_m := l_r$, $t_r := 2 * t_r$
17:     end if
18: end for
19: return $t^* := t_m$
```

Algorithm 6: Step size tracking procedure for the HCS (for its two versions).

(or alternatively the one described in section 4.2.2) with $\tilde{x}_i \in B(x_0, r)$, $i = 1, \ldots, N_{nd}$, and such that $(x_0, \tilde{x}_i)$, $i = 1, \ldots, N_{nd}$, are mutually non-dominating. Then, we propose to proceed analogously to [125], where a step size strategy for multi-objective continuation methods is suggested: given a target value $\epsilon_y \in \mathbb{R}_+$ (*i.e.*, the minimal value which makes two solutions distinguishable from a practical point of view), the task is to compute a new candidate $x_{new} = x_0 + \tilde{t}a$ such that

$$||F(x_0) - F(x_{new})||_\infty \approx \epsilon_y. \tag{4.9}$$

In case F is Lipschitz continuous (see [96]), there exists an $L \geq 0$ such that

$$||F(x) - F(y)|| \leq L||x - y||, \quad \forall x, y \in Q. \tag{4.10}$$

This constant can be estimated around x_0 by

$$L_{x_0} := ||J_{F(x_0)}||_\infty = \max_{i=1,\ldots,k} ||\nabla f_i(x_0)||_1,$$

where $J_{F(x_0)}$ denotes the Jacobian of F at x_0 and $\nabla f_i(x_0)$ denotes the gradient of the i-th objective at x_0.

In case the derivatives of F are not given (which is also considered here) the accumulated information can be used to compute the estimation

$$\tilde{L}_{x_0} := \max_{i=1,\ldots,N_{nd}} \frac{||F(x_0) - F(\tilde{x}_i)||_\infty}{||x_0 - \tilde{x}_i||_\infty},$$

since the $\tilde{x}_i$'s are near to x_0.

Finally, combining (4.9), (4.10) and using the estimation L_{x_0} the step size control is obtained by

$$x_{new} = x_0 + \frac{\epsilon_y}{L_{x_0}} \frac{a}{||a||_\infty}. \tag{4.11}$$

These step-size controls were used for the experiments in this, and the following chapter, obtaining competitive results. However, the step size control in multi-objective optimization is a very complicated problem. This will be later discused in detail in Section 6.1.2, page 115.

4.2.4 Constraints management

During the run of the search algorithm, it can occur that some of the new iterates are not inside the feasible domain Q. That is to say, we are faced with the situation that $x_0 \in Q$ and $x_1 := x_0 + h_0 v \notin Q$, where v is the search direction. In that case, we propose to proceed analogously to the well-known bisection method for root finding in order to backtrack from the current iterate x_1 to the feasible set:
let $in_0 := x_0 \in Q$ and $out_0 := x_1 \notin Q$ and

$$m_0 := in_0 + 0.5(out_0 - in_0) = x_0 + \frac{h_0}{2} v.$$

If $m_0 \in Q$ set $in_1 := m_0$, else $out_1 := m_0$. Proceeding in an analogous way, one obtains a sequence $\{in_i\}_{i \in \mathbb{N}}$ of feasible points which converges linearly to the boundary ∂Q of the feasible set. One can, for example, stop this process with an $i_0 \in \mathbb{N}$ such that

$$||out_{i_0} - in_{i_0}||_\infty \leq \text{tol},$$

obtaining a point in_{i_0} with maximal distance tol to ∂Q.

Algorithm 7 illustrates this described process. Note that by this procedure no function evaluation has to be spent, though a feasibility test may also be of relevant numerical effort in some cases. In case the domain Q is given by box constraints, *i.e.*, if Q can be written as

$$Q = \{x \in \mathbb{R}^n \,:\, l_i \leq x_i \leq u_i,\, i = 1, \ldots, n\}\,, \tag{4.12}$$

where $l, u \in \mathbb{R}^n$ with $l \leq_p u$, the backtracking can be performed in one step: given a point $x_0 \in Q$ and a search direction ν the maximal step size h_{max} such that $x_0 + h_{max}\nu \in Q$ can be computed as shown in Algorithm 8.

```
Require: $x_0 \in Q$, $x_1 = x_0 + h_0\nu \notin Q$, $\text{tol} \in \mathbb{R}_+$
Ensure: $\tilde{x} \in \overline{x_0x_1} \cap Q$ with $\inf_{b \in \partial Q} ||b - \tilde{x}|| < \text{tol}$
 1: $in_0 := x_0$
 2: $out_0 := x_1$
 3: $i := 0$
 4: while $||out_i - in_i|| \geq \text{tol}$ do
 5:     $m_i := in_i + \frac{1}{2}(out_i - in_i)$
 6:     if $m_i \in Q$ then
 7:         $in_{i+1} := m_i$
 8:         $out_{i+1} := out_i$
 9:     else
10:         $in_{i+1} := in_i$
11:         $out_{i+1} := m_i$
12:     end if
13:     $i := i + 1$
14: end while
15: return $\tilde{x} := in_i$
```

Algorithm 7: Backtracking to feasible region when managing constraints in HCS.

Note that the HCS2 is proposed for the unconstrained case. While an extension to the constrained case for the hill climber is possible (see, *e.g.,* [42] for possible modifications) this does not hold for the movement along the Pareto set (*i.e.*, the side step). Though it is possible to extend system (4.5) by equality constraints (*e.g.,* by introducing slack variables to transform the inequality constraints into equality constraints) this could lead to efficiency problems in the numerical treatment [58]. Hence, we restrict ourselves here to the unconstrained case.

```
Require: feasible point $x_0 \in Q$, search direction $v \in \mathbb{R}^n \backslash \{0\}$, lower and upper bounds
    $l, u \in \mathbb{R}^n$
Ensure: maximal step size $h_{max}$ such that $x_0 + h_{max} v \in Q$
 1: for $i = 1, \ldots, n$ do
 2:     if $v_i > 0$ then
 3:         $d_i := (u_i - x_i)/v_i$
 4:     else if $v_i < 0$ then
 5:         $d_i := -(x_i - l_i)/v_i$
 6:     else
 7:         $d_i := \infty$
 8:     end if
 9: end for
10: $h_{max} := \min_{i=1,\ldots,n} d_i$
```

Algorithm 8: Algorithm to deal with box constraints when using HCS.

4.2.5 Numerical results

To illustrate the behavior of the HCS as a standalone procedure, we tested both instances—gradient-based and gradient-free version—on some examples. We use in the first two examples a convex model (*i.e.*, a model which does not contain local minima where the LS can get stuck) and we investigate both the unconstrained and the constrained cases. Finally, we consider a multi-modal and constrained model (ZDT4 [28]). All computations for these examples have been done using the programming language MATLAB[7].

Example 4.2.1 *Consider the MOP defined by minimizing*

$$\begin{aligned} f_1(x) &= (x_1 - 1)^4 + \sum_{i=2}^{n} (x_i - 1)^2 \text{ and} \\ f_2(x) &= \sum_{i=1}^{n} (x_i + 1)^2. \end{aligned}$$

The Pareto set $\mathcal{P}$ of this model is located within $[-1, 1]^n$.

Figure 4.4 shows the results obtained by the modified algorithms HCS1 and HCS2, when applied independently on Example 4.2.1. The MOP was set for dimension $n = 10$ and the experiment was performed over the domain $Q = [-5, 5]^{10}$. In both cases the same starting point x_0 has been chosen. Since $\mathcal{P}$ is located within Q in this example, no constraint handling techniques had to be applied in order to generate the sequence. Analyzing the cost of the procedures, we found in a typical trial that, for HCS1 a total of 1693 function calls had to be

[7] `https://www.mathworks.com`

spent in order to get this result. For HCS2, 207 function calls, 60 evaluations of the gradient and 192 evaluations of the Hessian were required (which are both given analytically); a conversion would lead to 13,095 function calls (We will explain this equivalences later in Section 5.4.2, page 93). It is evident that due to the different requirements of the algorithms a quantitative comparison is hardly possible. However, as a way of comparison, Figure 4.4 shows some qualitative differences as anticipated from the design of the different algorithms. We can see that HCS2 converges faster (in this case four iterates were needed to reach $\mathcal{P}$ while HCS1 needed 23 iterations) and the non-dominated front is better distributed compared to the results obtained by the gradient-free version HCS. Nevertheless, both results are satisfying since both non-dominated fronts represent a good approximation of the Pareto front with reasonable effort.

Example 4.2.2 *We consider now the constrained case of the above example. In this case, we have used dimension $n = 2$ and the domain has been restricted to*

$$Q = [0.5, 1.5] \times [1, 2].$$

The Pareto set, for this constrained case, is given by

$$P_Q = [0.5, 1] \times \{1\}$$

and thus included in the boundary of Q.

Figure 4.5 shows a numerical result from the HCS1 over the above constrained example. The plots show that also in this case the HCS1 is capable of approaching the solution set, and moving along it further on. However, a total of 997 function calls had to be spent in this setting, that is, more in comparison to the unconstrained case (note that the dimension of the model is much lower in the latter case).

Example 4.2.3 *We finally consider, in this example, the problem ZDT4 defined by minimizing*

$$\begin{aligned} f_1(x) &= x_1, \\ f_2(x) &= g(x)(1 - \sqrt{f_1/g(x)}), \end{aligned}$$

with

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10cos(4\pi x_i)),$$
$$0 \leq x_1 \leq 1, \quad -5 \leq x_i \leq 5, \ i = 2, \dots, n.$$

This is a highly nonlinear and multi-modal model.

Figure 4.6 shows two results, in image space, for two different initial solutions $x_0, z_0 \in Q = [0, 1] \times [-5, 5]^9$, when applying the two variants of the HCS. As anticipated, the results for both algorithms and starting points differ significantly; this is because the HCS is a local

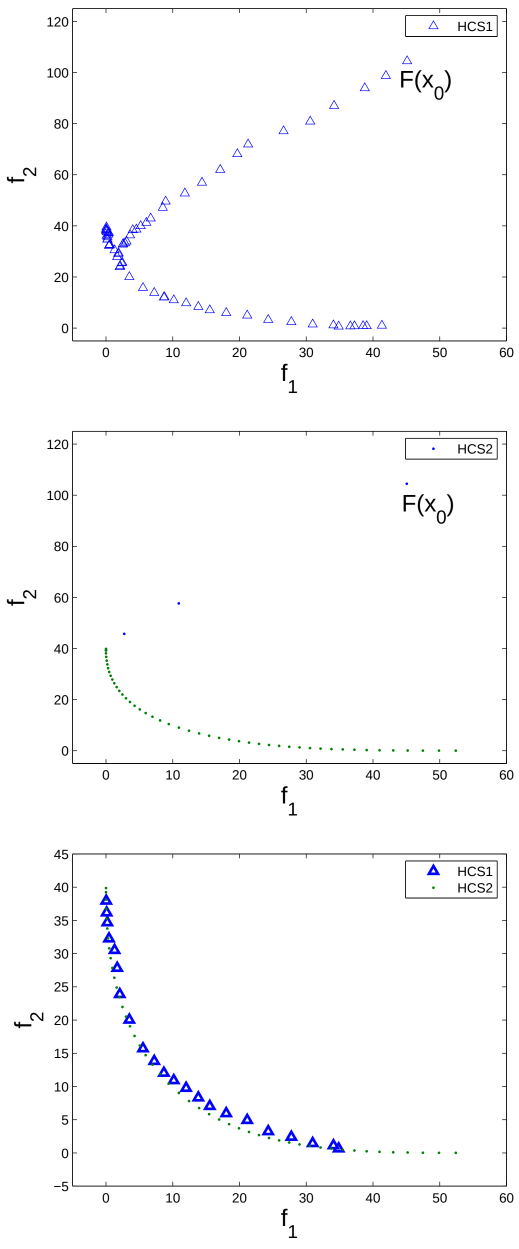
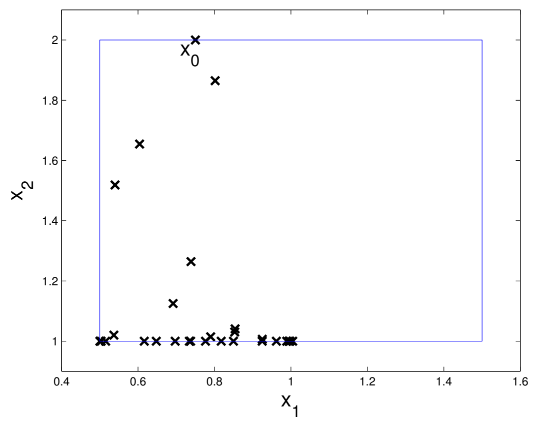
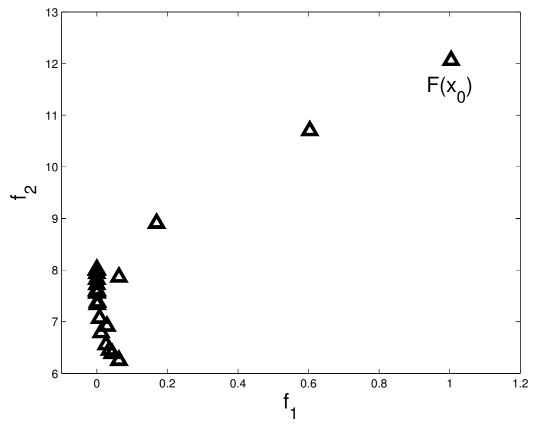


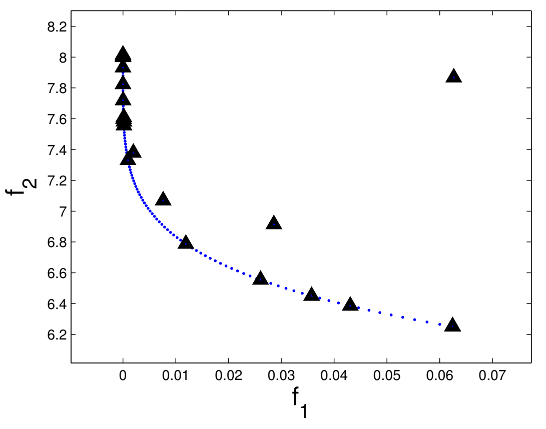
Figure 4.4: Numerical result of HCS for Example 4.2.1, with $Q = [-5, 5]^{10}$, in objective space. The comparison between the two versions is presented in the bottom plot, while the separate behaviors are at the top (HCS1) and middle (HCS2) ones.



(a) Parameter Space

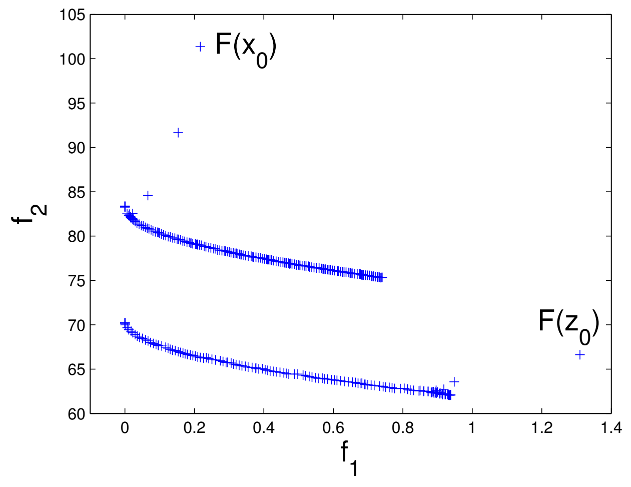


(b) Image Space

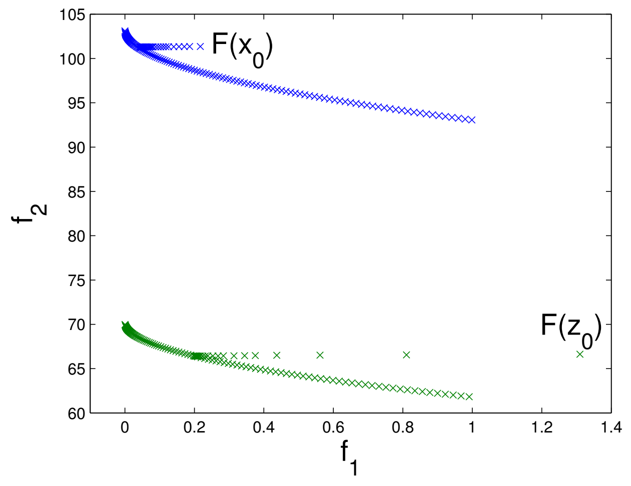


(c) Zoom Image Space and Pareto Front

Figure 4.5: Numerical result of HCS1 for the constrained Example 4.2.2 with $Q = [0.5, 1.5] \times [1, 2]$.



(a) Result HCS1



(b) Result HCS2

Figure 4.6: Numerical result of HCS1 and HCS2 for the multi-objective problem ZDT4, plotted in objective space, for two initial solutions x_0 and z_0.

strategy, and ZDT4 contains many local Pareto fronts. However, both procedures are able to explore a part of the local Pareto front—which is located 'near' to the image of the initial solution.

To conclude this section, it is worth to notice that the performance of the gradient-based HCS—in terms of convergence—is better than its gradient-free version (as it is shown in Figure 4.4); but, this improvement does not come for free: for the descent direction all objectives' gradients have to be available (or approximated), and to perform the linearization of $\mathcal{P}$ even all second derivatives are required. More tests of the HCS are presented later in this work, where this operator is combined with MOEAs and its performance as a hybrid algorithm is analyzed.

4.3 The Directed Search Method

When working with MOPs, performing LS movements towards a particular region is sometimes desired. In this line of thought, we present in this chapter a proposal for the computation of improving movements by the *Directed Search* (DS) method. The complexity of one iteration step is again equivalent to solving a system of linear equations, and only first order gradient information is necessary to use it.

This new approach calculates a gradient-based descent direction using a controlled bias towards interest regions determined in objective space. Hence, we claim that this proposal has many potential applications in the context of designing hybrid MOEAs. This method is based on a weighting approach but it is also able to reach non-convex regions on the front (see Figures 4.7 and 4.8). It can also be used instead of the greedy approaches presented in [42],[107].

The DS method describes the set of descent directions in a similar way to [10]; but, it performs movements inside this cone in a user-preference controlled way. *i.e.,* assuming $x_0 \in \mathbb{R}^n$, when using the Jacobian notation (3.1), in page 24, the descent cone (Definition 3.1.1 in page 22) can be expressed like:

$$J_{F(x_0)} \nu \leq 0 \quad \text{and} \quad J_{F(x_0)} \nu \neq 0. \tag{4.13}$$

After a possible normalization, Equation (4.13) can be re-stated as

$$J_{F(x_0)} \nu = -\alpha, \tag{4.14}$$

where $\alpha \in \mathbb{R}^k$ is a convex weight (*i.e.*, $\alpha_i \geq 0$ and $\sum_{i=1}^{k} \alpha_i = 1$). Hence, the descent cone can be represented as follows:

$$C_{x_0}(-, \dots, -) = \{\nu \in \mathbb{R}^n \backslash \{0\} \ : \ \exists \alpha \in \mathbb{R}^k \backslash \{0\} : \ \alpha_i \geq 0, \ J_{F(x_0)} \nu = -\alpha\} \tag{4.15}$$

and thus, every descent direction $\nu \in D(x_0)$ can be computed by solving the under-determined system of linear equations (4.14) for a given vector $-\alpha$. Note that α has to be determined first, but by this, v gains a physical meaning: by construction, the direction in image space is given by

$$\langle \nabla f_i(x_0), v \rangle = -\alpha_i, \quad i = 1, \dots, k. \tag{4.16}$$

In the following we use the notation $d = -\alpha$ to denote descent direction vectors. It is important to notice that such a search direction v can be computed by solving a system of linear equations. Since typically the number of parameters is (much) higher than the number of objectives in a given MOP, *i.e.*, $n >> k$, system (4.14) is (probably highly) under-determined which implies that its solution is not unique. To prevent this, the estimated solution with the lowest norm can be chosen leading to

$$v = J^+_{F(x_0)} d, \tag{4.17}$$

where $A^+ \in \mathbb{R}^{n \times k}$ denotes the pseudo inverse of a matrix $A \in \mathbb{R}^{k \times n}$, $k \leq n$. In case the rank of A is maximal, the pseudo inverse is given by $A^+ = A^T(AA^T)^{-1}$.

In case the MOP contains m active inequality constraints $g_1, \dots, g_m : \mathbb{R}^n \to \mathbb{R}$ at a point x—which is not within the scope of this work—one has to solve instead of (4.14) the enlarged system

$$\begin{aligned} J_{F(x)} v &= d \\ J_{G(x)} v &\leq 0, \end{aligned} \tag{4.18}$$

where

$$J_G(x) = \begin{pmatrix} \nabla g_1(x)^T \\ \vdots \\ \nabla g_m(x)^T \end{pmatrix} \in \mathbb{R}^{m \times n}. \tag{4.19}$$

For the solution of such systems we refer *e.g.,* to [11]. An analog statement for equality constraints, however, does in general not hold since they typically reduce the dimension of the search space, and then, the feasible choice of d may be restricted. In that case, one might change from the fixed direction d to a 'best fit', for instance by solving the following quadratic optimization problem:

$$\begin{aligned} &\min_v ||J^+_{F(x)} d - v||_2^2 \\ &\text{s.t. } J_{G(x)} v \leq 0, \text{ and} \\ &J_{H(x)} v = 0, \end{aligned} \tag{4.20}$$

where H contains the gradients of the equality constraints. If $d \leq 0$ and $d \neq 0$, *i.e.*, if d is a 'descent direction', then one can add the constraint

$$J_{F(x)} v \leq 0 \tag{4.21}$$

to (4.20) in order to prevent that the function values of subsequent candidate solutions decrease.

4.3.1 Tracing a solution curve

The above result can be used to define a curve of dominating points. Assume that a (not necessarily fixed) convex weight α is given and a search in that direction is desired. Using

$$v_\alpha(x) := -J^+_{F(x)}\alpha, \tag{4.22}$$

one can thus try to solve numerically the following initial value problem:

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x}(t) &= v_\alpha(x(t)), \quad t > 0 \end{aligned} \tag{IVP$_\alpha$}$$

A solution of (IVP$_\alpha$) yields a curve of dominating points, and the proportion of the improvements of the single objectives is given by α as shown in (4.16). Clearly, if all objectives are continuously differentiable and if $\alpha = \alpha(x)$ is continuous in x the solution curve $x(t)$ is also continuously differentiable.

Stopping Criterion

It is important to notice that even if an endpoint x^* of (IVP$_\alpha$) exists—for instance if $F(\mathbb{R}^n)$ is bounded below—this point does not have to be Pareto optimal since the approach depends next to α on the initial point x_0. At least, a stopping criterion for the numerical treatment of (IVP$_\alpha$) can be given to detect if an endpoint of the curve is reached under certain (reasonable) assumptions: if (a) the number of parameters n is at least as large as the number of objectives k and (b) if the gradients of all objectives are linearly independent at x_0, *i.e.*, rank$(J_{F(x_0)}) = k$ (which means that all objectives are indeed in conflict at x_0), then for every point $x(t)$ along the curve the rank of the Jacobian is k, except for the endpoint x^* (in particular $-\alpha$ is not in the image of $J_{F(x^*)}$). The rank of a matrix can of course not be used to detect the endpoint of a curve numerically, but instead the condition number κ_2 of $J_{F(x)}$ can be used: one can *e.g.,* compute

$$\kappa_2(J_{F(x)}) = ||J_{F(x)}||_2||J^+_{F(x)}||_2 = \frac{\sigma_1}{\sigma_k}, \tag{4.23}$$

where σ_1 and σ_k are the largest and smallest singular value of $J_{F(x)}$, respectively, and stop the process if $\kappa_2(J_{F(x_i)}) \geq$ tol, where tol $\in \mathbb{R}_+$ is a given (large) threshold. This can be done since by the above discussion $\kappa_2(J_{F(x(t))}) \to \infty$ for $x(t) \to x^*$.
This discussion shows one potential drawback of the approach, namely that the determination of the search direction by solving (4.14) gets inaccurate for points near the Pareto set due to the high condition number of $J_{F(x)}$. However, our experience has shown that state-of-the-art numerical tools allow to come 'near enough' to the Pareto set even for higher dimensional problems (here we refer to the numerical results presented in Sections 4.3.3 and 4.3.5, in pages 59 and 65 respectively).

To trace the solution curve of (IVP$_\alpha$) one can *e.g.,* choose well-established numerical discretization methods (*e.g.,* [35]); Algorithm 9 shows a generic procedure. Also, we think that Armijo-like step size controls for the choice of h_i such as the ones presented in [5, 41] can be adapted to the current context (whereby (4.16) has to be integrated in a suitable way). However, none of these methods allow to perform a correction back to the solution curve. In the following we propose one possible way to compute the solution curve numerically by predictor corrector (PC) methods (*i.e.*, with a method which allows for such a correction, see *e.g.*,, [2]). To do this recall that for every point x on the solution curve it holds

$$F(x) = F(x_0) + \lambda_y \alpha, \tag{4.24}$$

where $\lambda_y \in \mathbb{R}$. Hence, the curve is contained in the zero set of

$$\begin{aligned} H &: \mathbb{R}^{n+1} \to \mathbb{R}^k \\ H(x, \lambda_y) &= F(x) - F(x_0) - \lambda_y \alpha \end{aligned} \tag{4.25}$$

To comply with the needs of PC methods we introduce an additional parameter λ_x into the solution curve (which is defined in parameter space):

$$\begin{aligned} x(0) &= (x_0, \lambda_{x,0} = 0) \in \mathbb{R}^{n+1} \\ \dot{x}(t) &= \begin{pmatrix} \nu_\alpha(x(t)) \\ 1 \end{pmatrix}, \quad t > 0 \end{aligned} \tag{IVP$_{\alpha,\lambda}$}$$

So far, we are not able to apply PC methods since the parameters λ_x and λ_y parametrize different curves. The following consideration, however, argues that it is reasonable to match the two parameters: Let x_0 be given and $\lambda_{x,0} = 0$, and let $(x_1, \lambda_{x,1})$ be an Euler step of (IVP$_{\alpha,\lambda}$) with a small step size $\Delta\lambda_x$, *i.e.*,

$$x_1 = x_0 + \Delta\lambda_x \nu_\alpha(x_0), \quad \lambda_{x,1} = \Delta\lambda_x. \tag{4.26}$$

By construction of $\nu_\alpha(x_0)$ and since $\Delta\lambda_x$ is small we have

$$\alpha_i \approx \frac{f_i(x_1) - f_i(x_0)}{\Delta\lambda_x}, \quad i = 1, \ldots, k. \tag{4.27}$$

This implies that $F(x_1) - F(x_0) \approx \Delta\lambda_x \alpha$ and hence $H(x_1, \lambda_{x,1}) \approx 0$ which suggests to make the substitution $\lambda_x = \lambda_y$. Using this, (IVP$_{\alpha,\lambda}$) and (4.25) can now be used to perform classical PC methods in order to trace *one* solution curve: starting with the point $(x_0, \lambda_{x,0})$ one can integrate (IVP$_{\alpha,\lambda}$) numerically for a small time step, *e.g.,* via the Euler method as described above leading to a predictor solution $(\tilde{x}_1, \tilde{\lambda}_{x,1})$. In a next step this solution can be corrected to the desired curve (in objective space). That is, starting with $(\tilde{x}_1, \tilde{\lambda}_{x,1})$ and using a (Gauss-) Newton method applied on (4.25) one can try to find a solution $(x_1, \lambda_{x,1})$ with $H(x_1, \lambda_{x,1}) \approx 0$, and so on. For details on the method including the step size control and the properties of the

(Gauss-) Newton method we refer *e.g.,* to [2]. Note that $H^{-1}(0)$ forms locally a $(n-k+1)$-dimensional set and hence no unique solution is defined (but also not needed). However, if the (Gauss-) Newton method is taken for the corrector as described in [2] λ_x can be chosen as the pseudo arc length to trace such a curve.

```
Require: starting point $x_0 \in \mathbb{R}^n$ with rank$(J_{F(x_0)}) = k$, tol $\in \mathbb{R}_+$, convex weight $\alpha_0 \in \mathbb{R}^k$.
 1: $i := 0$
 2: while $\kappa_2(J_{F(x_i)}) <$ tol do
 3:     compute $v_i = -J^+_{F(x_i)}\alpha_i$
 4:     compute $h_i \in \mathbb{R}_+$
 5:     set $x_{i+1} := x_i + h_i v_i$
 6:     choose $\alpha_{i+1} \in \mathbb{R}^k$
 7:     set $i := i + 1$
 8: end while
```

Algorithm 9: Directed Search Method

4.3.2 Directed movements

To illustrate the potential of biasing search movements, we use the example of Goal Programming (GP), where the main task is to find a point such that its image is as close as possible to a given target value $Z \in \mathbb{R}^k$. This leads, for the unconstrained case, to the following optimization problem (see *e.g.,* [17] for more information):

$$\min_{x\in\mathbb{R}^n} d(Z, F(x)), \tag{4.28}$$

where $d(\cdot,\cdot)$ is a particular (chosen) distance in $\mathbb{R}^k$. When choosing the Euclidean distance, the (local) best search direction at a point x_0 is given by

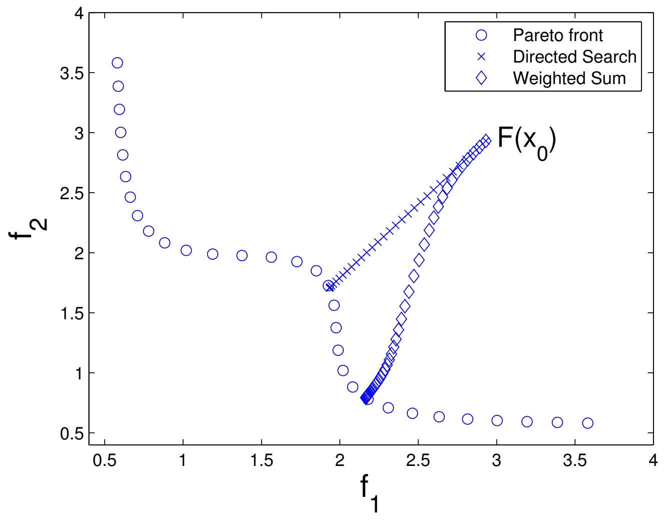
$$\alpha_{Z,2} := \frac{F(x_0) - Z}{||F(x_0) - Z||}, \tag{4.29}$$

where we assume that $Z \leq_p F(x_0)$. To satisfy user preferences, or to be able to reach alternative Pareto optimal points, it is commonly desired to use a weighted metric instead of a fixed metric (*e.g.,* [90]). When using the weighted 2-metric

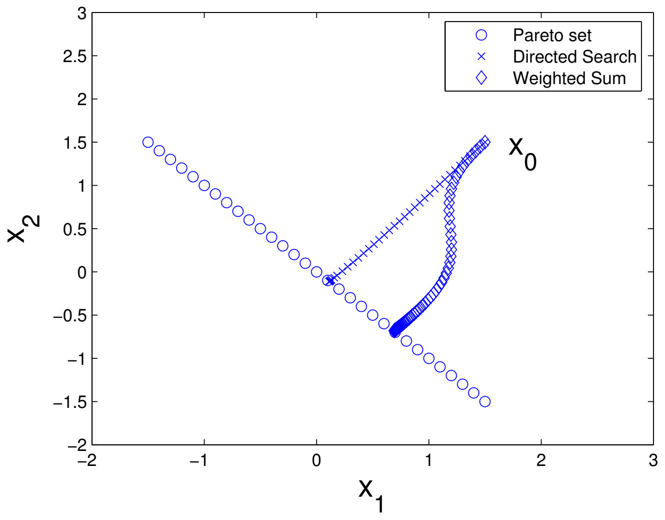
$$d_D(x,y) := \sqrt{(x-y)^T D(x-y)}, \tag{4.30}$$

where D is a diagonal matrix with positive diagonal entries, the greedy direction at x_0 is given by

$$\alpha_{Z,D} := \frac{D(F(x_0) - Z)}{||D(F(x_0) - Z)||}. \tag{4.31}$$

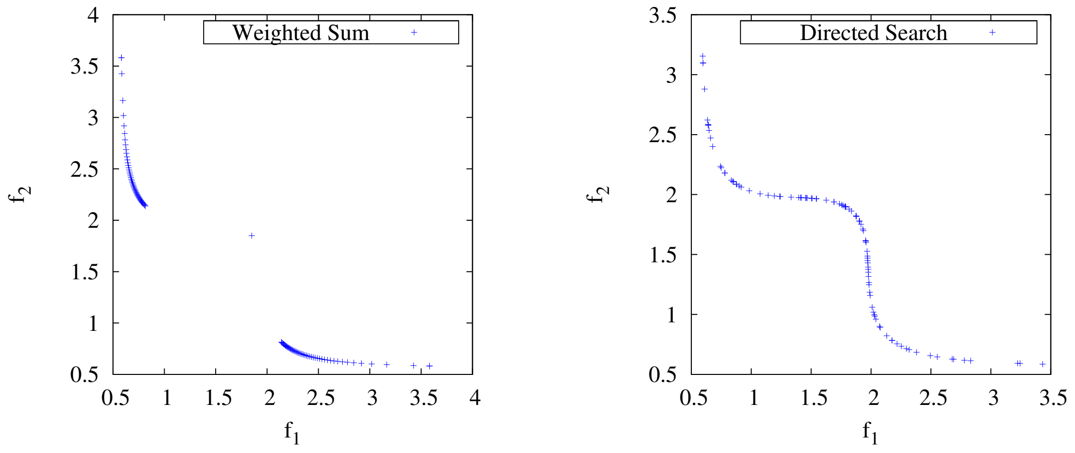


(a) Objective Space

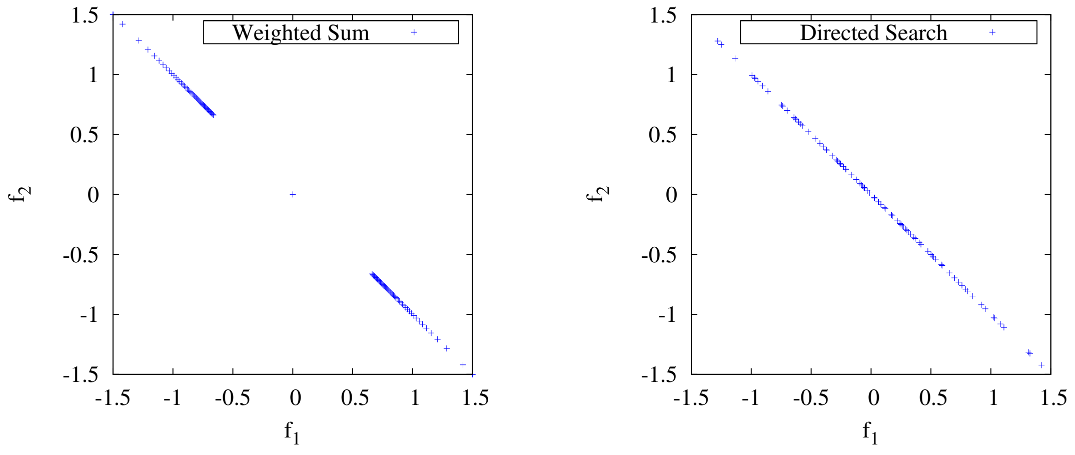


(b) Parameter Space

Figure 4.7: Numerical result for MOP (4.37): comparison of the numerical solution paths of the weighted sum and the directed search approach for $x_0 = (1.5, 1.5)$ and $\alpha = (0.5, 0.5)^T$.



(a) Objective Space



(b) Parameter Space

Figure 4.8: Numerical result for the same MOP as in Figure 4.7. Comparison of the solutions, for $\alpha = (0.5, 0.5)^T$, of the weighted sum (left) and the directed search approach (right) using 100 randomly chosen initial points.

To derive equation (4.31), we define for a given point x_0 and a fixed $\alpha \in \mathbb{R}^k$ the curve $c_\alpha : \mathbb{R} \to \mathbb{R}^k$ by

$$c_\alpha(t) = F(x_0) + t\alpha. \tag{4.32}$$

Also, let $g_\alpha : \mathbb{R} \to \mathbb{R}$ be the square of the weighted 2-metric of Z and $c_\alpha(t)$, *i.e.*,

$$g_\alpha(t) = d_D(Z, c_\alpha(t))^2 = \sum_{i=1}^{k} d_i(Z_i - f_i(x_0) - t\alpha_i)^2, \tag{4.33}$$

where d_i is the i-th diagonal element of D. Then, the derivative is given by

$$g'_\alpha(t) = -\sum_{i=1}^{k} 2d_i\alpha_i(Z_i - f_i(x_0) - t\alpha_i), \tag{4.34}$$

and hence

$$g'_\alpha(0) = -2\sum_{i=1}^{k} d_i\alpha_i(Z_i - f_i(x_0)). \tag{4.35}$$

Finally, using (4.35) we can determine the most greedy choice of α, *i.e.* the steepest descent with respect to α given by

$$-\nabla_\alpha(g'_\alpha(0)) = D(Z - F(x_0)), \tag{4.36}$$

and the claim follows (note that the directional vector is negated in (4.14)). To conclude, the vectors $v(\alpha_{Z,2})$ and $v(\alpha_{Z,D})$ can be viewed as the 'best' LS direction for the Goal Programming (4.28) according to the given metric.

4.3.3 Numerical results for the DS method

Here we present some numerical results to illustrate the Directed Search descent method. All computations, in this section, have been done using MATLAB[8].

Example 4.3.1 *First we consider the following parameter dependent MOP ([146][124]):*

$$\begin{aligned} &f_1, f_2 : \mathbb{R}^2 \to \mathbb{R} \\ &f_1(x,y) = \frac{1}{2}(\sqrt{1+(x+y)^2} + \sqrt{1+(x-y)^2} + x - y) + \lambda \cdot e^{(-x-y)^2} \\ &f_2(x,y) = \frac{1}{2}(\sqrt{1+(x+y)^2} + \sqrt{1+(x-y)^2} - x + y) + \lambda \cdot e^{(-x-y)^2} \end{aligned} \tag{4.37}$$

For $\lambda = 0.85$ the Pareto front contains a dent. To be more precise, the Pareto front is connected and consists of two convex parts and one concave part.

[8] `http://www.mathworks.com`

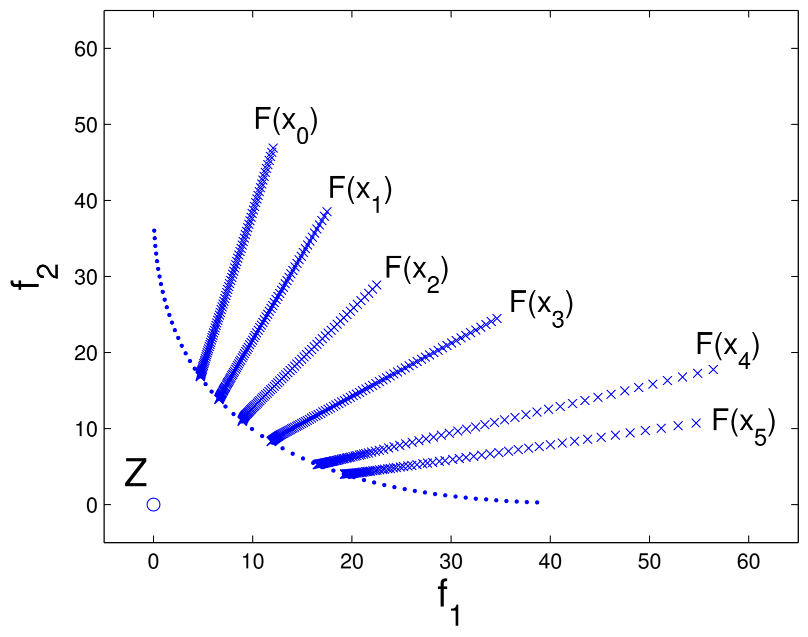


Figure 4.9: Numerical result for MOP (4.38): solution paths for several initial conditions x_i, $i = 0, \dots, 5$, using the 2-metric for d.

Figure 4.7 shows a comparison of the Weighted Sum method against the Directed Search method. Starting point for both methods was $x_0 = (1.5, 1.5)$, and the weight vector was chosen as $\alpha = (0.5, 0.5)^T$. For the stopping criterion (4.23) we have chosen for this as well as for the subsequent examples tol $= 1e8$. To demonstrate the solution curves we have chosen to solve (IVP_α) with the Euler method using very small step sizes. While the solution curve of the Directed Search method steers on a straight line from $F(x_0)$ to the corresponding point on the Pareto front (the same appears in parameter space), the solution curve of the Weighted Sum approach eludes this straight line leading to a (for this method) better solution. We have repeated this for 100 randomly chosen starting points within $Q = [-5, 5]^2$; and the resulting endpoints for both methods are shown in Figure 4.8. While it may be argued that for a single solution the Weighted Sum approach obtained a good solution (Figure 4.7), this does not hold in terms of a possible representation of the entire *Pareto set, since the concave part of the Pareto front is left out nearly completely.*

Example 4.3.2 *Next, we consider the following convex MOP:*

$$f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$$

$$f_i(x) = \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_j - a_j^i)^2 + (x_i - a_i^i)^4, \tag{4.38}$$

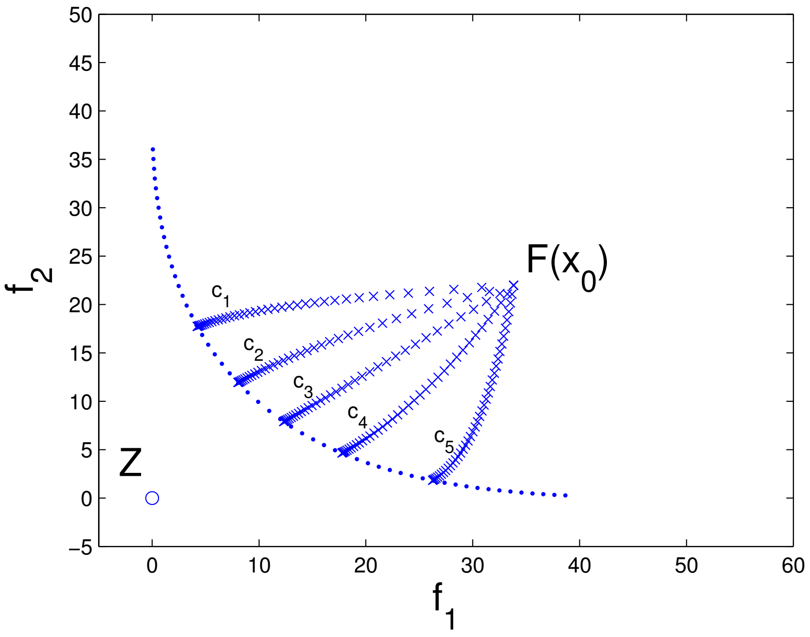


Figure 4.10: Numerical result for MOP (4.38): solution paths for one initial point and for several weighted metrics (see text).

where

$$\begin{aligned} a^1 &= (1, 1, 1, 1, \ldots) & \in \mathbb{R}^n \\ a^2 &= (-1, -1, -1, -1, \ldots) & \in \mathbb{R}^n, \end{aligned}$$

First we turn our attention to the goal programming problem (4.28) where we choose the target value as $Z = (0, 0)$. Figure 4.9 shows some numerical solution curves for several starting points where the Euclidean Distance has been chosen.
In Figure 4.10 some numerical results are shown for several problems where x_0 has been fixed, but the weighed 2-metric has been chosen for different matrices. To be more precise, the curves c_i, $i = 1, \ldots, 5$, belong to the matrices D_i, where

$$\begin{aligned} c_1 &\quad \textit{corresponds to } D_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.1 \end{pmatrix}, \\ c_2 &\quad \textit{corresponds to } D_2 = \begin{pmatrix} 0.7 & 0 \\ 0 & 0.3 \end{pmatrix}, \\ c_3 &\quad \textit{corresponds to } D_3 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}, \\ c_4 &\quad \textit{corresponds to } D_4 = \begin{pmatrix} 0.3 & 0 \\ 0 & 0.7 \end{pmatrix}, \\ c_5 &\quad \textit{corresponds to } D_5 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.9 \end{pmatrix}. \end{aligned}$$

4.3.4 Continuation process for the DS method

In this section we propose a new PC method for the continuation along (local) Pareto sets of a given MOP. The central difference from a classical method is that it is based on the geometry of the Pareto front and realized by the Directed Search Method. This new method—unlike classical PC methods—does not require to compute the Hessian matrices of the objectives; this implies that this method can be used to handle even higher dimensional models ($n >> 1000$), without exploiting the possibly given sparsity of the system.

We concentrate on the two-objective case (*i.e.*, $k = 2$) since a consideration of $k > 2$ requires an additional data structure for the efficient representation of the approximation (for this we refer *e.g.*, to [55, 112, 115]). Apart from that, all subsequent ideas are suitable for models with $k > 2$.

a) Predictor Assume that we are given a Pareto point x_0 with an associated weight α_0 such that

$$\sum_{i=1}^{k} \alpha_{0,i} \nabla f_i(x_0) = 0, \tag{4.39}$$

and further we assume that

$$\text{rank}(DF(x)) = k - 1. \tag{4.40}$$

It is known [57] that in this case α is orthogonal to the Pareto front, i.e.,

$$\alpha \perp T_y \partial F(\mathbb{R}^n), \tag{4.41}$$

where $y = F(x)$ and $\partial F(\mathbb{R}^n)$ denotes the border of the image $F(\mathbb{R}^n)$ (this holds even in the constrained case, see [57]). Thus, a search orthogonal to α (in objective space) could be promising to obtain new predictor points. To use the direct approach (4.14), for instance a QR-factorization of α can be computed, *i.e.,*

$$\alpha = QR, \tag{4.42}$$

where $Q = (q_1, \dots, q_k) \in \mathbb{R}^{k\times k}$ is an orthogonal matrix and $q_i,\ i = 1, \dots, k$, its column vectors, and $R = (r_{11}, 0 \dots, 0)^T \in \mathbb{R}^{k\times 1}$ with $r_{11} \in \mathbb{R}\backslash\{0\}$ (for the computation of such a factorization we refer *e.g.,* to [96]). Since by (4.42) $\alpha = r_{11}q_1$, i.e., $\alpha \in span\{q_1\}$, and Q orthogonal it follows that the column vectors $q_2, \dots, q_k$ build an orthonormal basis of the hyperplane which is orthogonal to α. Thus, a promising well-spread set of search directions v_i may be the ones which satisfy

$$DF(x)v_i = q_i, \quad i = 2, \dots, k. \tag{4.43}$$

Since α is not in the image of $DF(x)$ (else x would not be a Pareto point) and by assumption (4.40) it follows that the vectors $q_2, \dots, q_k$ are in the image of $DF(x)$, i.e., Equation (4.43) can be solved for each $i \in \{2, \dots, k\}$. Note that by this choice of predictor direction no second derivative of the objectives are required.
The predictor direction can—except for its sign—be chosen as described above, *i.e.*, one of the normalized vectors $v := \pm v_2/||v_2||_2$, where v_2 satisfies (4.14) as described above for $d = q_2$. To orientate the curve (*i.e.*, to determine the sign of v) we can not proceed as for 'classical' PC methods since this would require as well the derivative of $\tilde{F}$ as in (4.5). Instead, one can define an orientation in the context of two-objective optimization by the increase (or decrease) of one objective. For this, the signum of the according entry of the direction vector q_2 can be taken. If, for instance, an improvement according to f_2 is sought, then

$$p := x_0 - \mathrm{sgn}(q_{2,2})hv \tag{4.44}$$

can be chosen as predictor, where $q_{2,2}$ denotes the 2nd entry of q_2, and h is the desired step size. Note that when choosing one of the data structures in [55, 112, 115] for models with $k > 2$ no orientation of the solution manifold is required.
To choose the step size h we suggest to proceed as follows: assume we are given x_0 and the search direction v with $||v||_2 = 1$ associated to the direction q in objective space for the predictor, *i.e.*, $p = x_0 + hv$, where $h \in \mathbb{R}_+$ has to be chosen. The underlying idea of the step size control is as follows: to obtain an adequate spread of the solutions the function values $f_j(x)$ and $f_j(p)$ of at least one objective $j \in \{1, \dots, k\}$ differ ideally by a (problem dependent) value ϵ while the difference for all other objectives does not exceed this threshold.

Since this value can differ for each objective the demand on the spread can be stated (after possible renormalization) as follows:

$$d_w(F(p), F(x)) \approx \epsilon \tag{4.45}$$

where d_w is the weighted infinity distance, *i.e.*,

$$d_w(x, y) = \sum_{i=1}^{k} w_i |x_i - y_i|. \tag{4.46}$$

Assuming that each objective is Lipschitz continuous and that the step size h_i for the i-th objective is sufficiently small we obtain for $i = 1, \ldots, k$:

$$\underbrace{|f_i(p) - f_i(x)|}_{\overset{!}{=} \frac{\epsilon}{w_i}} \approx L_{i,x} \underbrace{||p - x||_2}_{=h_i} \tag{4.47}$$

Since $L_{i,x}$ can be approximated by the norm of the directional derivative we obtain for each objective the control

$$h_i = \frac{\epsilon}{w_i |\langle \nabla f_i(x), v \rangle|}, \quad i = 1, \ldots, k, \tag{4.48}$$

and hence for the entire MOP

$$h := \min_{i=1,\ldots,k} h_i. \tag{4.49}$$

By construction, the difference vector $F(p) - F(x)$ is ideally orthogonal[9] to α which can be used to determine if the chosen step size (4.49) is too large. If

$$|\langle \alpha, F(p) - F(x) \rangle| \leq \text{tol}, \tag{4.50}$$

where tol $\in \mathbb{R}_+$ is a given tolerance, the predictor p can be accepted. If (4.50) is not true, then p does probably not serve as a good predictor, and the step size has to be decreased accordingly.
Alternative step size controls for multi-objective continuation can be found *e.g.,* in [57, 113].

b) Corrector Given a predictor p, the subsequent solution along the curve can be computed by solving numerically (IVP$_\alpha$), using p as initial value and choosing α_0, *i.e.*, the weight from the previous solution x_0 leading to a new solution x_1. This together with the step size control (4.49) and the 'quasi-orthogonality' test (4.50) is intended to obtain an even spread of the solutions. In fact, this is the case if the value of ϵ in (4.45) and hence the step size t is sufficiently small. The new associated weight α_1 can be updated by solving the following quadratic optimization problem (3.7).

[9] If the Pareto front around $F(x)$ is convex, this can only be reached asymptotically.

Algorithm 10 shows a possible realization of the continuation method which does not require the second derivatives of the objectives. Hereby, we start with an approximate minimizer of the first objective f_1 and trace the curve seeking for improvements according to the second objective f_2. Hence, a possible stopping criterion is that the associated weight of a candidate solution is approximately $\tilde{\alpha} = (0, 1)$. Other stopping criteria, however, are possible according to the given setting.

```
Require: Initial solution $(x_0, \alpha_0)$ with $\alpha_0 \approx 1$, threshold $\epsilon \in \mathbb{R}_+$, tolerance $\delta \in \mathbb{R}_+$.
Ensure: Set of candidate solutions $x_i$
 1: $i := 0$
 2: while $1 - \alpha_2 > \delta$ do
 3:     compute $q_2$ as in (4.42)
 4:     compute $v$ as in (4.43) (i.e., $v := v_2$)
 5:     compute $h$ as in (4.49)
 6:     $p_i := x_i - sgn(q_{2,2})hv$
 7:     compute $x_{i+1}$ by solving (IVP$_\alpha$) with initial value $p_i$ and using $\alpha_i$.
 8:     compute $\alpha_{i+1}$ as in (3.8)
 9:     set $i := i + 1$
10: end while
```

Algorithm 10: Bi-Objective Continuation

4.3.5 Numerical results for the DS continuation method

Here we present some numerical results to illustrate the novel continuation approach. All computations on this section have been again done using MATLAB.

Example 4.3.3 *First, we are interested in solving MOP (4.38) by continuation methods. Figure 4.11 shows a numerical result for $n = 10$. The figure shows the images of the solutions $F(x_i)$ as well as the images of the predictors $F(p_i)$ which are already near to the solutions.*
We have observed that the corrector step uses approximately three iterations to be near enough to the Pareto set (to be more precise: in the above example we needed, on average, 2.4 iterations). We use this to make a comparison to the classical PC method:
the classical PC method uses in each predictor step one QR-factorization of $(\tilde{F}')^T(x, \alpha)$ which implies one Jacobian call (for all objectives) and one Hessian call. For the corrector at least one Gauss-Newton step has to be performed which implies again one Jacobian and one Hessian call (here we neglect possible additional function calls due to backtracking strategies). In total, we can assume that for the generation of a new candidate solution two Jacobians and two Hessians have to be calculated. The PC method as described above requires one Jacobian call for the predictor and we estimate three Jacobians for the corrector (as observed in this example), which makes in total the computation of four Jacobians to

obtain a new solution. For the sake of comparison, we assume that the gradient information is not available but has to be calculated or approximated, and hence measure the cost to find a new candidate solutions in terms of the number of required function calls. If, for instance, automatic differentiation (AD) is used to compute the derivatives, we can estimate $5 * k$ *function calls for each derivative call and* $k * (4 + 6n)$ *function calls for each calculation of the Hessian ([47]). Then we obtain for our example (*$n = 10$, $k = 2$*):*

$$|fc(PC_{classical})| = 276, \quad |fc(PC_{new})| = 40, \tag{4.51}$$

where $fc(A)$ *denotes the number of required function calls for method A. These values change when using finite differences (FD). If, for instance, the forward difference quotient*

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x_1, \ldots, x_i + \delta_i, \ldots, x_n) - f(x_1, \ldots, x_n)}{\delta_i}, \quad i \in \{1, \ldots, n\} \tag{4.52}$$

where $\delta_i \in \mathbb{R}_+$ *is a small value, is used to estimate the gradient, apparently* n *function calls are required to estimate each gradient. The central difference quotient leads to more accurate approximations, but does in turn require* $2n$ *function calls [47]. A forward difference quotient approximation of the second derivative requires a total of* n^2 *function calls (and* $2n^2$ *or* $4n^2$ *function calls when using the central difference quotient, depending on how the rule is applied). Hence, using the forward difference rule in FD we obtain*

$$|fc(PC_{classical})| = 440, \quad |fc(PC_{new})| = 80. \tag{4.53}$$

In both cases, i.e., AD and FD, the new PC method requires a much smaller number of function calls for the computation of the subsequent solution.

Figure 4.12 shows a numerical result for the same model and the same setting but using $n = 100,000$ *parameters. Also here, the cost to obtain a new candidate solution is approximately given by four Jacobian calls (we omit here a comparison as in (4.51) and (4.53)). Since a naive storage of the Hessian matrix (i.e., without exploiting the sparsity) requires the size of* n^2 *floats, a straightforward implementation of the classical PC method is restricted on a standard computer to approximately* $2,000$ *free parameters, which does not hold for the new method. Since the second derivatives are not needed for the latter approach, we think that this one is—independent of the storage problem—a promising alternative to the classical PC method in particular for higher dimensional problems.*

Example 4.3.4 *Finally, we reconsider MOP (4.37) from Example 1 and turn our attention again to Goal Programming. Though it was shown above, that* α_Z *is* locally *the best search direction, it is not even guaranteed that a solution of (*IVP_α*) leads to a local solution of (4.28). This is due to the fact that the approach depends next to* x_0 *and* Z *also on the shape of the Pareto front which is clearly not known a priori. As a possible remedy it seems possible to combine the two methods proposed in this work to obtain a two-stage algorithm for the detection of local solutions of (4.28): first, one can compute, starting from* x_0*, the endpoint* x^* *those image is on the border of* $F(\mathbb{R}^n)$*. In the second step, a movement along* $\partial F(\mathbb{R}^n)$ *can*

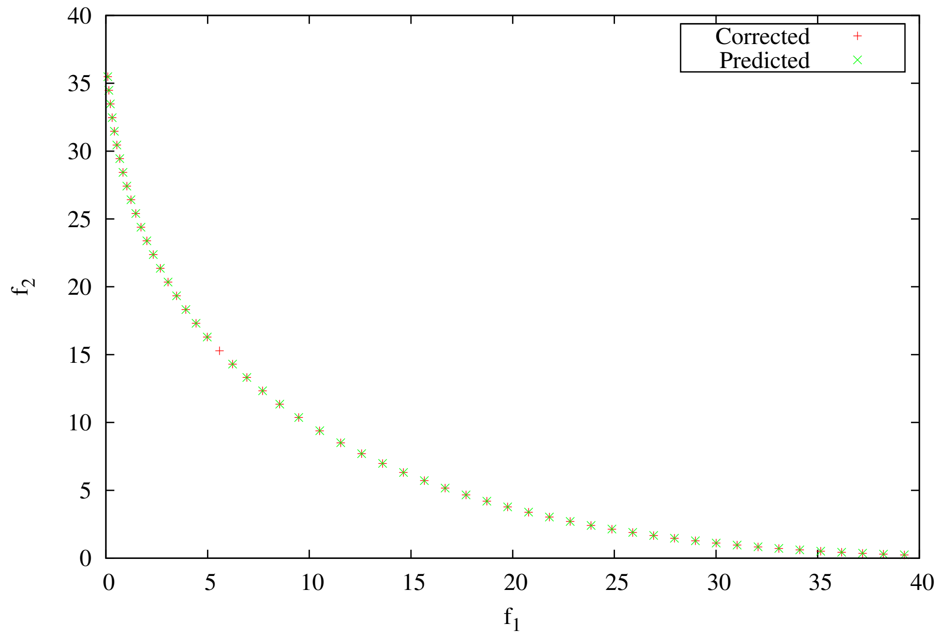


Figure 4.11: Numerical result for MOP (4.38): solution of the continuation algorithm for $n = 10$. Shown are the images of the predictors and the solutions.

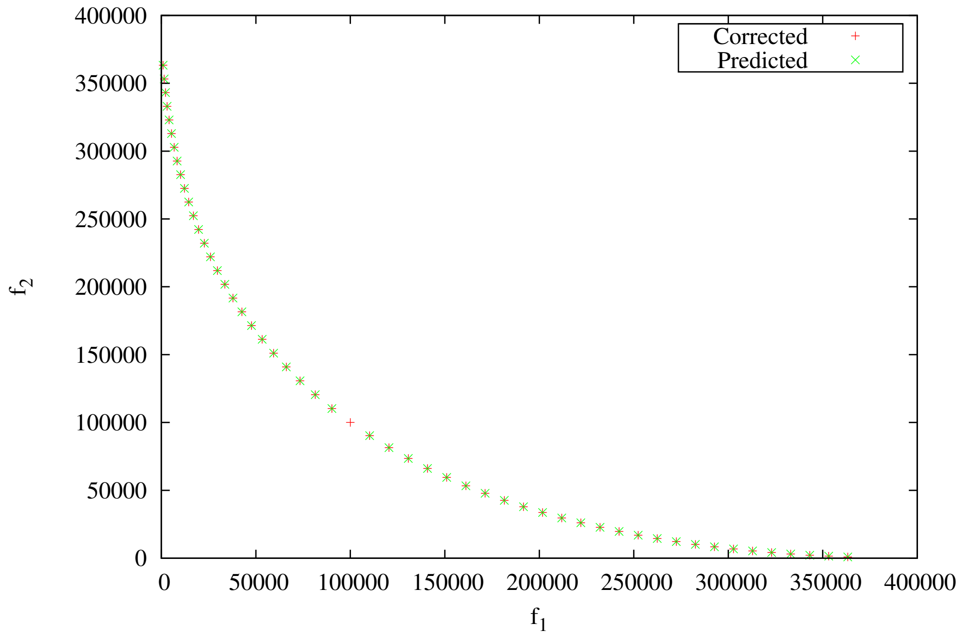


Figure 4.12: Numerical result for MOP (4.38): solution of the continuation algorithm for $n = 100,000$. Shown are the images of the predictors and the solutions.

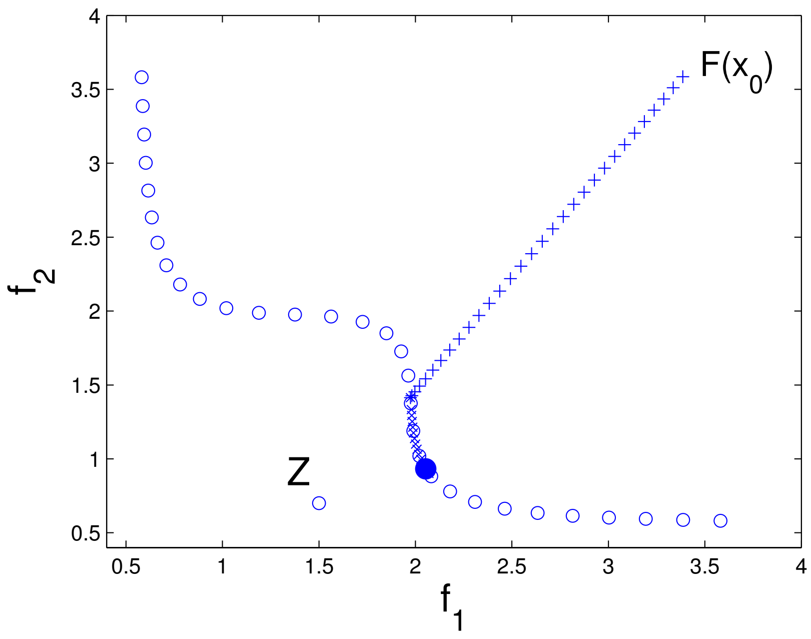


Figure 4.13: Numerical result for MOP (4.38): a combination of the Directed Search descent method (indicated by the stars) and a modified continuation procedure (crosses) leads to the solution of the goal programming problem (filled circle).

be performed seeking for a decrease of the distance toward Z.
Figure 4.13 shows such an example for MOP (4.37). The continuation has been done as described above but we have changed the orientation according to the problem, and stopped the process as soon as the search direction flipped (monitored by the value of q_2). Though the first result is quite promising, we feel that much more investigation has to be done in that direction which we leave for future research.

4.3.6 Discussion

It is worth noticing that the idea to adjust the search direction in image space a priori to obtain a numerical scheme has been previously discussed in the literature [39]. Some consideration that have to be taken are, for example, that the computation of the search vector v requires the solution of a possibly highly under-determined system of linear equations, and the condition number of this system increases as the point x gets nearer to a local solution. However, we have observed that state-of-the-art numerical tools are able to handle such problems, even for high dimensional problems (for instance, in Section 4.3.5 the Pareto set of an MOP with $n = 100,000$ parameters is presented). Another important observation is that it is still unclear how to 'steer' the optimization process, *i.e.*, how to choose α. Here, we set

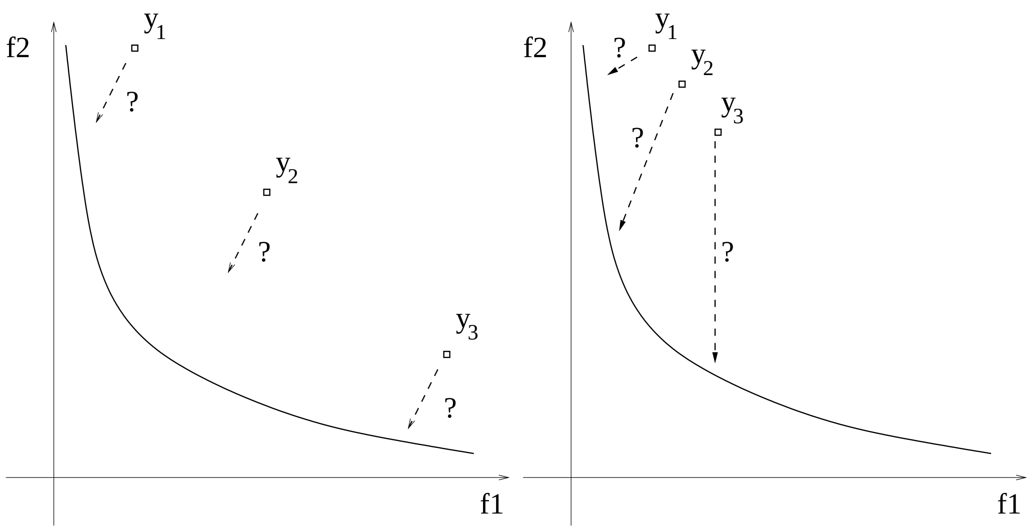


Figure 4.14: In the context of population based optimization, the effective choice of the search directions depend on several factors. One of them is the location of the individuals of the population.

a start and present some choices which are locally optimal in certain cases. However, some 'global' strategies are to be established in the future.
A very similar representation of the descent cone which is the basis for our descent method can be found in [8]. However, the authors of [8] derived their result differently, and did not exploit the 'steering feature' in their work.

To conclude, there exist some options to compute multi-objective search directions instead of the traditional weighted sum; but the question about how to efficiently integrate them into a population-based context—as in the set oriented (MOEAs) algorithms—remains open. It is also worth noting that the suitable choice of the movement direction relies also on the location of the point, and on the location of all the other population individuals (see Figure 4.14); to this end, the directed search method has a lot of potential.

"Thought is only a flash between two long nights, but this flash is everything."

Henri Poincare

5 Gradient-based Memetic Algorithms

Along this chapter, our different proposals for hybridization of gradient-based LS and MOEAs are presented. First, we establish a two-stage algorithm and discuss the advantages and disadvantages of this point of view, followed by numerical examples to study the scalability of the method. Finally, our proposals for hybridization of two state-of-the-art MOEAs is presented and discussed; also, some numerical experiments are presented. In the following section, we state some important definitions about the performance indicators used for the sequel.

5.1 Comparison Methodology

Building memetic algorithms (MA) has the aim of improving the efficiency of the base EA—a MOEA in case of multi-objective problems. To verify this improvement, a fair comparison of performance is necessary; more precisely, we expect to get a better solution with the same computational effort. Assessing the quality of solutions in multi-objective optimization has certain complications. Thus, the methodology to make the comparison turns over determining which is a better approximation of a continuous manifold between two finite solution sets (A and B), each one corresponding to the output of each different algorithm.

Several performance indicators have been used during the last few years, in order to compare the quality of Pareto front approximations. Even with these performance indicators, making a good comparison is not an easy task. The main problem is how to assess that a certain finite approximation of a continuous $k-$dimensional manifold is 'better' than another one. In multi-objective evolutionary computation, three different aspects on solution quality are important. First, we want the solutions to be near to the true Pareto front of the MOP. Second, these solutions must cover most of the regions in the manifold. Finally, these solutions should be well spread over the space. In the rest of the chapters, we use certain performance indicators to assess the efficiency of our algorithms taking into account these different aspects; here, we will briefly describe them:

Given a finite approximation A of the Pareto front $\mathcal{PF}$ of a certain MOP, denote by δ_i the minimum Euclidean distance from a given point $y_i, i = 1, \ldots, |A|$, to the true Pareto front $\mathcal{PF}$. Then, the *Generational Distance* (GD) of a set (population) A is defined [143] as

$$GD = \frac{1}{|A|}\sqrt{\sum_{i=1}^{|A|} \delta_i^2}. \tag{5.1}$$

Notice that in practice, $\mathcal{PF}$ is also a finite approximation of the true Pareto front. This operator indicates proximity of A to the Pareto front.

Another indicator, most commonly used in practice implies a distance from $\mathcal{PF}$ to A; and it is known as the *Inverted Generational Distance* (IGD) [18]. This not only indicates the proximity of the set A to the front, but also gives a certain sense about its extension. The IGD is analogous to GD, but measured from $\mathcal{PF}$ to A; and is defined by

$$IGD(A) := \frac{\sqrt{\sum_{i=1}^{|\mathrm{PF}|} d_i^2}}{|\mathrm{PF}|}.$$

where PF is the finite representation of $\mathcal{PF}$, and d_i is the Euclidean distance from the i—th element of PF to the set A.

Given two finite subsets A and B of $\mathbb{R}^n$ the *Two Set Coverage Measure* [149] is defined as

$$SC(A,B) = A \prec B = \frac{|\{b \in B \text{ such that } \exists a \in A \text{ with } a \prec b\}|}{|B|} \tag{5.2}$$

If $A \prec B = 1$, it means that all the elements of B are dominated by at least one element of A. On the other hand, $A \prec B = 0$, means that no element of B is dominated by any element of A. Since the Two Set Coverage Metric is not symmetric, always both values $SC(A, B)$ and $SC(B, A)$ have to be taken into account.

To compute a relative measure of the distance between solutions, the *Spacing* indicator [108] is a commonly used option. It is defined as:

$$S = \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|}(d_i - \overline{d})^2}$$

where

$$d_i := \min_{\substack{j=1,\ldots,|A| \\ i \neq j}} \sum_{m=1}^{k} |y_i{}^{(m)} - y_j{}^{(m)}|; \qquad \text{and}$$

$$\overline{d} := \frac{1}{|A|}\sum_{i=1}^{|A|} d_i, \quad y_i \in A \subseteq \mathbb{R}^k.$$

the value of Spacing should be smaller as soon as the solutions y_i are close to uniformly spaced. *i.e.*, the smaller the spacing value is, the better is the distribution of the output solutions for a particular algorithm.

Finally, the Hausdorff distance d_H, of two sets $A, B \in \mathbb{R}^k$, is defined [103] as

$$d_H := \max\{dist(F_{true}, F_{known}), dist(F_{known}, F_{true})\}$$

where

$$dist(u, A) := \inf_{v \in A} ||u - v||, \quad \text{for } u \in B,$$

and

$$dist(B, A) := \sup_{u \in B} dist(u, A).$$

The Hausdorff distance measures how far two subsets are from each other, so a value near to zero is desired.

5.2 Two-stage Algorithm GBMES

In the following, we explain our proposal for a two-stage approach, called *Gradient-Based Multi-objective Evolutionary Strategy* (GBMES), which is designed to perform a gradient-based search on a reduced population. This approach is complemented with a technique to reconstruct the front at the end. Examples of some hybrid two-stage multi-objective algorithms can be found in [56] and [78].

Since MOEAs are known to perform well on problems with high multi-frontality, our method focuses instead on problems in which the gradient descent is a good option to speed up the first stages of the search. Although obtaining gradient information is an expensive process—because it requires several objective function evaluations—we have shown that it is possible to design a gradient-based hybrid which is very efficient. For this sake, it is important to devise a careful interleaving between the MOEA and the gradient-based search engine, so that we do not exceed a modest budget of function calls. Such a balance is achieved in particular by GBMES, which only performs a total of 3,000 objective function evaluations for the test problems—which is a very small value compared to the number of evaluations previously reported in the specialized literature for such problems. Another interesting aspect about this approach (which we believe that is shared by other hybrids between MOEAs and gradient-based methods[1]) is that it scales well as we increase the number of decision variables of a MOP. This is illustrated in this section by two examples in which we use up to 100 decision variables. When increasing the dimension of the MOP, our proposed approach is found to degrade significantly less than a state-of-the-art MOEA (in this case, the NSGA-II [31]), even while still performing a total of 3,000 objective function evaluations.

[1]This also depends on the sort of method adopted to approximate the derivatives of the functions.

5.2.1 Description of the Gradient-based Multi-objective Evolutionary Strategy

In the following, we describe the two stages that conform GBMES. The aim of the first stage is to obtain a small set of Pareto optimal points. In the case of facing moderate multi-frontality, the evolutionary part of the hybrid algorithm can deal with the critical points that are not optimal; then, we assume that by the end of this first stage, all the points in this set are part of the global front. The second stage is devoted to the reconstruction of the entire front, starting from a few solutions found in the first stage.

5.2.2 First stage (Approximation)

Several populations are involved in the first stage of the algorithm. There are a replaceable and a non-replaceable population, used in an analogous way to the Micro-GA for Multi-objective Optimization [24]. More precisely, we adopt an external population P with a replaceable part Pr and a non-replaceable part Pn. The former population will evolve over time and the latter will introduce diversity into the process. We use a small population Pt of parents, with $|Pt| = \mu$, which is randomly chosen from $Pr \cup Pn$. The individuals from Pt are recombined, at each iteration, to produce a set P_{off} of λ descendants. Unlike a traditional evolution strategy, we set $\lambda \approx \mu$ because we need to bound the number of function calls in this phase—in order to spend most of them in the gradient-based descent part. The individuals used for recombination are randomly chosen from Pt. We use two types of recombination, arithmetic and discrete ones, choosing one of them randomly, with a certain (predefined) probability. We set a higher probability for the discrete recombination at this stage (the proportion is 2:1). For the second stage of GBMES, arithmetic recombination plays a more important role.

Once we have selected the nondominated parents from $Pt \cup P_{off}$, to conform P_{nd}, we perform an insertion process in order to obtain the *secondary population* S, and the *elite population* E such that $E \subseteq S$. This insertion process has two cases illustrated by Algorithm 11, for the first time that it is applied, and by Algorithm 12, for further iterations. The secondary population is directly conformed by the points obtained after applying the steepest descent procedure, based on the descent direction proposed in [42]—also described in Section 3.3. To be more precise, the descent direction that we are looking for is v^* as in Expression (3.3); and, as Fliege suggests, the step length for the movement can be obtained by an Armijo's rule, *i.e.* by decreasing t until the condition

$$F(x + tv) \leq F(x) + \beta t JF(x) v \tag{5.3}$$

is fulfilled. The value $\beta \in (0, 1)$ is a control parameter to decide how fine grained, numerically speaking, the descent will be. After having $x^1 = x + tv$, the process is repeated by calculating a new descent direction for x^1 or, if this is not possible, we can assume that a critical point has been achieved.

The condition for a point $x \in S$ to be in E is precisely having finished the descent with an α^* (from (3.3)) value of zero—which means it is a candidate to be a local Pareto point. Note that the latter could not be the case for all the points in S. Thus, this separation is made because the descent could be stopped before obtaining a local Pareto point. This happens because, in practice, we restrict the number of steps used in the descent. Additionally, we could also stop the descent if a certain tolerance value between the initial and the final points in the movement is achieved. Also, in order to handle box constraints, when the computed steepest descent direction leads outside of the constraint bounds, the process must quit exploiting that solution, and then, take the point nearest to the constraint, as the final point of the movement. This is necessary, since it is possible that the movement of the descent points leads to a local optimum located in the infeasible region or to a local optimum located in the boundary between the feasible and the infeasible region. In both cases, it is unnecessary to perform a fine-grained steepest descent, and the cost of performing it could lead to an important increase in the number of function calls. Then, during the procedure these end points from the descent would enter the elite population, E, only if E is empty. The algorithm's implementation should detect the case when, after certain number of generations, no point has entered the elite population. In this case, we perform uniform mutation on the individuals in the secondary population and we only retain the nondominated solutions as candidates for becoming the outcome of the first-stage. In order to feed the replaceable population, we take P''_{nd}, which is conformed by individuals from P'_{nd} that entered the secondary population S at that generation, and we use them to replace those individuals from Pr that are dominated. To select the μ parents to conform Pt for the next generation, we take $min\{\mu/3, |P''_{nd}|\}$ individuals from P''_{nd} and the remainder are randomly selected from P. The general process of the first stage is described in Algorithm 13.

```
1: procedure INSERTION, CASE 1($p$, $P_{nd}$, $S = \emptyset$, $E = \emptyset$)
2:     for $p \in P_{nd}$ : do
3:         Perform the gradient-based descent for $p$ and get $p'$ as the final point of the descent.
4:         Insert the element: $S \leftarrow S \cup \{p'\}$. If the necessary condition for being in $E$ holds, then $E \leftarrow E \cup \{p'\}$.
5:     end for
6: end procedure
```

Algorithm 11: Case 1 of the insertion process of GBMES (applied for the initial generation)

For all the problems tested here, the values for μ and λ were 20, the initial population size was of 130 individuals, and the proportion between the non-replaceable and the replaceable parts of the initial population was 3:10. Regarding the value of μ, in general, it must be a trade-off because it has to be big enough in order to allow a good sampling of the population for the evolution strategy; but not too large, in order to avoid that too many function calls are spent.

```
 1: procedure INSERTION, CASE 2($p$, $P_{nd}$, $P_r$, $S$, $E$.)
 2:     Set $P'_{nd}$ as the elements from $P_{nd}$ that are not dominated by any $s \in E$.
 3:     for $p \in P'_{nd}$ : do
 4:         Perform the gradient-based descent for $p$ and get $p'$ as the final point of the descent.
 5:         Remove $s \in S$ such that $p' \prec s$.
 6:         Insert $p'$ in $S$ (in $E$ if applicable).
 7:         Remove $q \in P'_{nd}$ such that $p' \prec q$.
 8:     end for
 9:     Use the set of inserted points $p'$ to replace the dominated individuals from $P_r$.
10: end procedure
```

Algorithm 12: Case 2 of the insertion process of GBMES (applied in case that $generation \geq 2$.)

```
Ensure: $E$, $S$.
 1: procedure GBMES (FIRST STAGE)
 2:     Initialize the replaceable $P_r$ and non-replaceable $P_{nr}$ parts of the population.
 3:     Select $\mu$ parents to conform $P_t$.
 4:     Recombine the parents to produce $\lambda$ offspring $Off$.
 5:     Select $P_{nd}$ nondominated individuals from the union of parents and offspring.
 6:     Perform the insertion process with the gradient-based descent, and feed the secondary population $S$, the elite population $E$, and the replaceable population $P_r$.
 7:     Repeat steps 3 to 6 until having at least four individuals in the elite population or until running out of the function-calls budget allowed for this first stage.
 8: end procedure
```

Algorithm 13: Description of the first stage of GBMES.

5.2.3 Second stage (reconstruccion)

Once the Pareto front has been approached, the next step consists of a technique that performs a reconstruction of the entire front departing from a few points out of this set. One possibility at this point is to use continuation methods [112, 58]; if this is the case, only one point in the front is necessary to start, but we would have to be able to afford the computational cost of calculating the second derivatives of the functions, which is high, even if clever techniques are devised [50]—here we refer to the discussion in [86].

For the second stage of GBMES, we propose for example to use a reconstruction technique based on Rough Sets [98, 99]; and we reserve at least one third of our function-calls budget for this part. This approach (rough sets) consists of a stochastic technique which uses information about individuals that were dominated in the last iteration, in order to construct new solutions close to the nondominated individuals and far away from the dominated ones. This aims to generate new nondominated solutions and, as a consequence, fill up the missing parts of the Pareto front. For a thorough explanation of the technique we refer to [105, 106]. Here we give a brief description of how we used it for our purposes.

a) Preliminary Phase: Our approach requires an initial population ($P1$) which is close to the true Pareto front. This population is partitioned into two sets: DS, which contains the dominated solutions and NS, which contains the nondominated solutions. At the end of the first stage of GBMES, we assume that we have a few (at least four) individuals in the true Pareto front. Then, to start the rough sets procedure, we can generate a small population in the neighborhood of these solutions in the following way: For each point in the elite population, we generate a $n-$dimensional box with each point as one vertex and its nearest neighbor located in the opposite corner. Then, we generate two types of offspring: first, we apply total arithmetic recombination to get s descendants in the line that joins the reference point and its nearest neighbor and we produce another offspring outside the box, but in the same direction as before (see Figure 5.1). Next, the second kind of offspring consists of t descendants which are randomly built inside the box. For the examples presented here, we used $s = 2$ and $t = 3$.

b) Rough Sets: Once the population is close enough to the true Pareto front, and it is partitioned into the sets NS and DS, we perform a sequence of iterations until function-calls budget is spent. At each iteration, we build a grid with k dominated points from DS, which serve as vertices. We also take q individuals from the set of nondominated solutions NS and we apply a bounded mutation to them. The bounds for these mutations are set from half of the distance over each coordinate to the limits of the grid (see Figure 5.2). If there is no edge bounding the mutation operator, we set the limit for that specific coordinate as the natural limits of the box constraints from the problem. The complete procedure is described in Algorithm 14. For the examples we present here, we used $k = 20$, $q = 10$, $u = 4$, $v = 100$ for the two-objective problem, $v = 150$ for the three-objective problem, and $v' = 100$ in both cases.

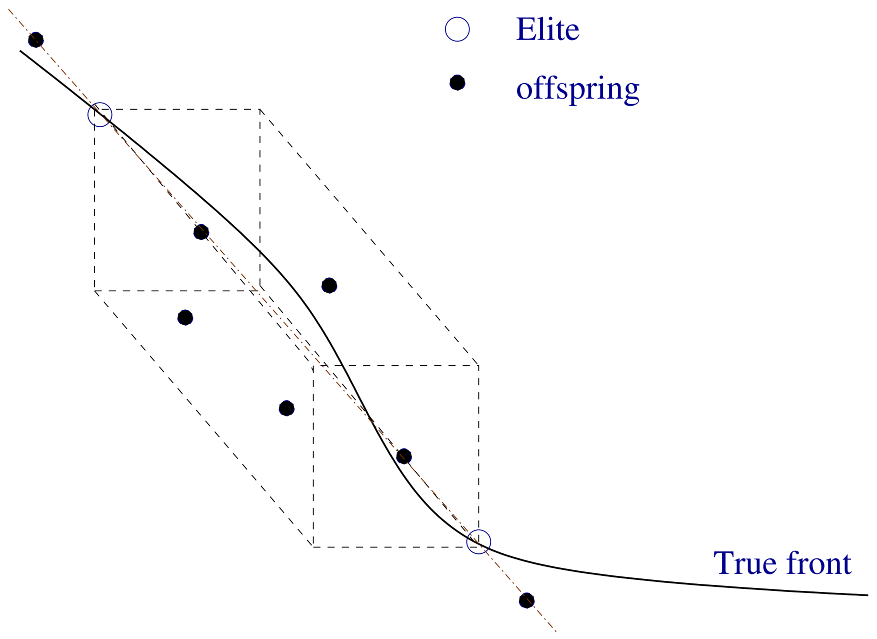


Figure 5.1: This figure shows the box formed by two elements in the elite set. Their offspring lie on the line that joins them, and in the neighborhood bounded by their own coordinates (n-dimensional box).

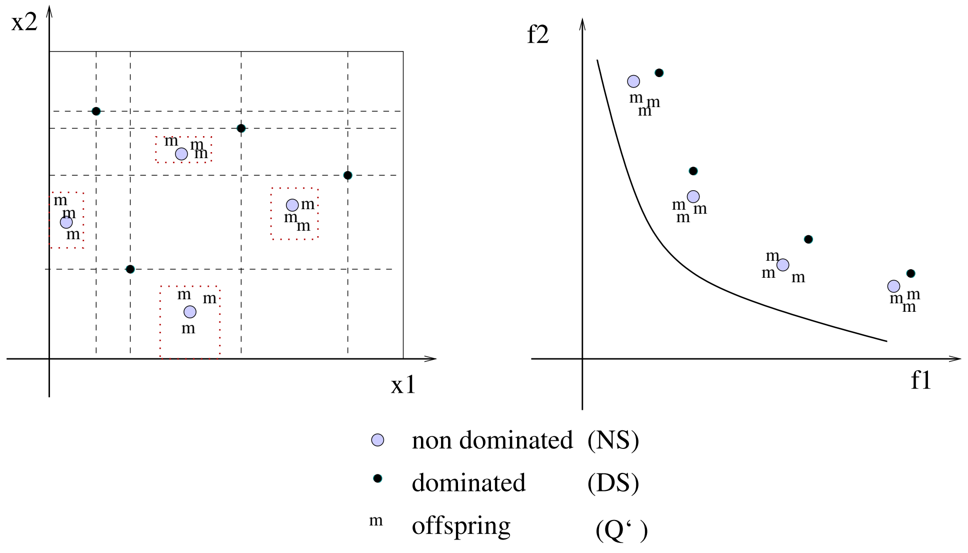


Figure 5.2: This figure shows the grid for the rough sets method formed with the elements of DS serving as vertices (marked as black dots in the figure). The nondominated individuals (taken from NS), which are used as the reference solutions, are marked with gray circles. Their descendants, produced by mutation, are marked with the letter ‘m’.

```
1:  procedure ROUGH SETS($NS, DS$)
2:      Divide the population $P1$ into the $DS$ and $NS$ sets.
3:      Randomly choose $k$ elements from $DS$ to set the limits of the grid.
4:      Randomly choose $q$ elements from $NS$ to form $Q$ (the set of parents).
5:      Apply $u$ mutations to each parent and conform $Q'$ with them.
6:      Divide the population $P \cup Q'$ into the $DS$ and $NS$ sets.
7:      If $v < |NS|$, we use a crowding or a clustering technique to reduce the size to $v$. $v$
    is a user-defined parameter.
8:      If $v' < |DS|$, we keep the solutions that were nondominated in the last iteration,
    and we choose the rest randomly from the dominated solutions in the population, until
    reaching the maximum size allowable for $v'$. $v'$ is a user-defined parameter.
9:      Repeat steps 3 to 8 until a certain (predefined) number of function-calls is performed.
10: end procedure
```

Algorithm 14: Adaptation of the rough sets technique to reconstruct the Pareto set.

5.2.4 Numerical results

In order to test the scalability of our proposed hybrid approach, we adopted the two problems defined in Table 5.1. These test problems have two and three objectives, respectively; Besides, they are both scalable in decision variable space. The final population for both algorithms was set to 100 individuals, for the problem with two objectives, and to 150 individuals for the problem with three objectives (a larger value was adopted in this case because of the higher number of objectives). Our results are compared with respect to those generated by the NSGA-II [31] using also 100 and 150 individuals, respectively; and performing the same number of evaluations as our proposed hybrid approach. Tables 5.2 and 5.3 show our comparison of results after performing 3,000 objective function evaluations for both problems. In this case, we use $n = 10$ on Problem1, and $n = 12$ on Problem2. From Tables 5.2 and 5.3, we can see that GBMES achieves much better convergence than the NSGA-II. This can be corroborated by looking at Figure 5.3, in which it is clear that the NSGA-II is unable to converge, even when Problem 1 only has ten decision variables.

Subsequently, we increased the number of decision variables of the problem and focused our analysis on the convergence of each approach (ours and the NSGA-II). Figures 5.3 to 5.6 show the plots of the final population, corresponding to the run in the mean obtained for IGD over 30 runs, for Problem 1, using $n \in \{10, 30, 60, 100\}$ decision variables. We can observe that, as we increase the number of decision variables, GBMES is still able to generate an important portion of the Pareto front (e.g., it is able to generate the "knee" [12] in all cases) with the same number of evaluations as before (3,000). The values of the performance measures (shown in Tables 5.4 to 5.6) indicate that both approaches suffer a performance degradation as we increase the number of decision variables. This behavior is consistent in the case of Problem 2, as well. Although the performance of our proposed GBMES degrades as we increase the number of decision variables, such degradation is less significant than the

Table 5.1: MOPs adopted for our experiments.

Problem 1 $f_1(x) = (x_1 - 1)^4 + \sum_{i=2}^{n}(x_i - 1)^2$ $f_2(x) = \sum_{i=1}^{n}(x_i + 1)^2$ with $n = 10$
Problem 2 (DTLZ2) $f_1(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2})\ldots\cos(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $f_2(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2})\ldots\sin(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $\vdots$ $f_{k-1}(x) = \cos(\frac{x_1\pi}{2})\sin(\frac{x_2\pi}{2})(1 + g(x))$ $f_k(x) = \sin(\frac{x_1\pi}{2})(1 + g(x))$ $g(x) = \sum_{i=k}^{n}(x_i - \frac{1}{2})^2$ $0 \leq x_i \leq 1,\ i = 1, \ldots, n$ with $k = 3,\ n = 12$

one suffered by the NSGA-II. Also, in all cases (for both problems), our approach outperforms the NSGA-II with respect to all the performance measures used to assess convergence (see Tables 5.4 to 5.6). This can be better appreciated in Figures 5.7 to 5.9 for Problem 1 and from 5.10 to 5.12 for Problem 2, in which we show a graphical comparison of the performance (regarding convergence) of the two approaches (ours and the NSGA-II) as we increase the number of decision variables.

5.3 Discussion on Memetic Aspects

Local search can improve a MOEA's performance if it is efficiently customized for a particular problem. Applying LS is expected to be less expensive, in terms of function evaluations, for combinatorial problems than for continuous problems. This is because evaluating fitness functions of neighbor solutions can be efficiently done in the combinatorial case. Nevertheless, we can take some ideas from the treatment of such problems, and find analogies that can be used in the continuous-space case.

Even when a computer represents only a finite set of points when discretizing continuous domains, the magnitude of this discretization makes it impossible to apply, in the continuous case, the currently available memetic algorithms designed for discrete spaces. In some combinatorial problems, a good LS mechanism explores more solutions than the evolutionary search (see for example [71, 67]). This is due to the fact that producing new solutions is faster with certain LS operators—especially in the combinatorial case—than with traditional evolutionary operators. Furthermore, quickly exploring new candidate solutions is not as

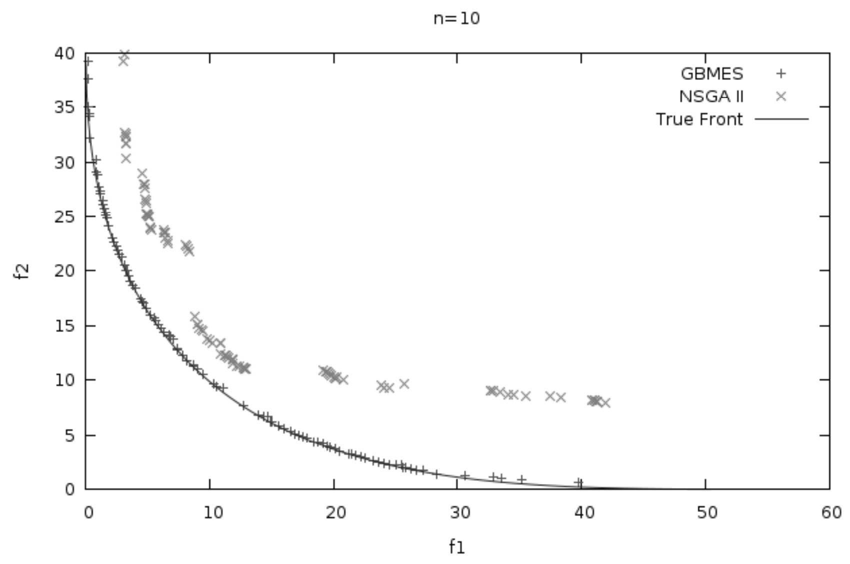


Figure 5.3: This graph shows the approximation of the Pareto front generated by GBMES and NSGA-II on Problem1 (with $n = 10$), after 3,000 function evaluations.

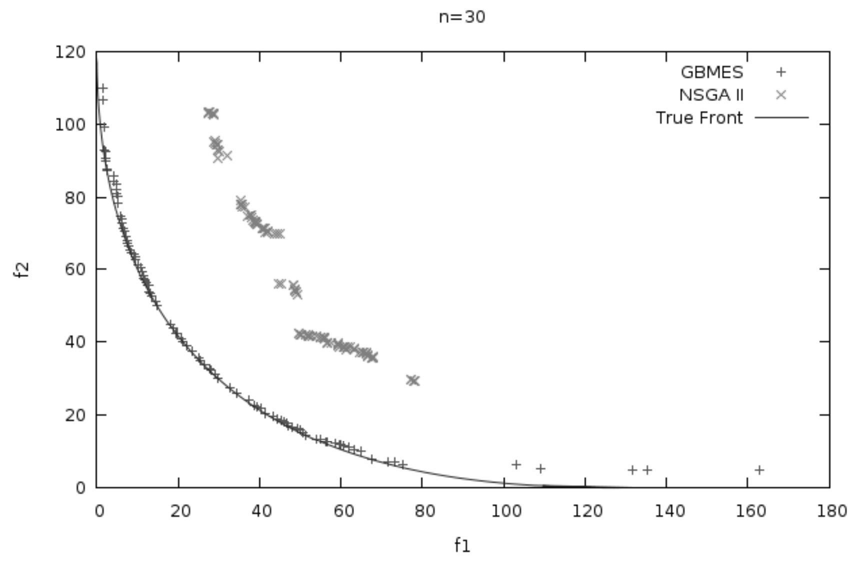


Figure 5.4: This graph shows the approximation of the Pareto front generated by GBMES and NSGA-II on Problem1 (with $n = 30$), after 3,000 function evaluations.

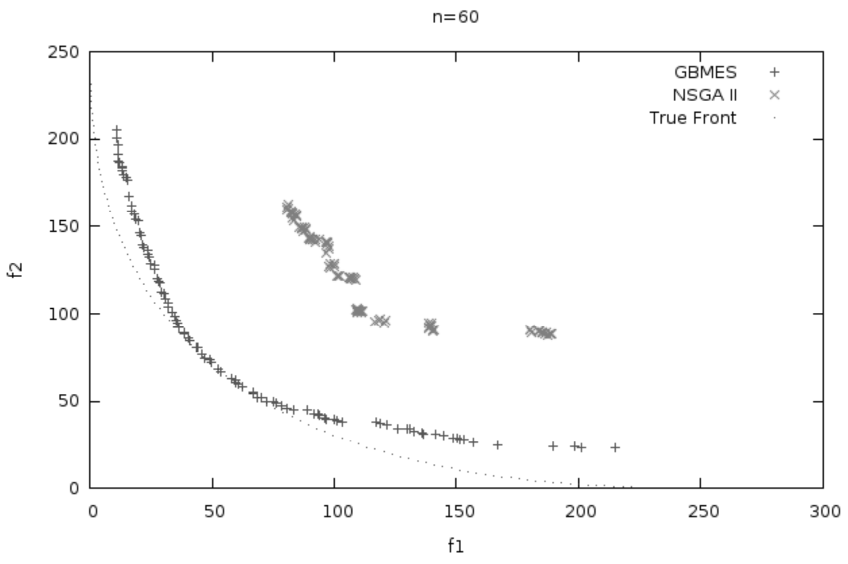


Figure 5.5: This graph shows the approximation of the Pareto front generated by GBMES and NSGA-II on Problem1 (with $n = 60$), after 3,000 function evaluations.

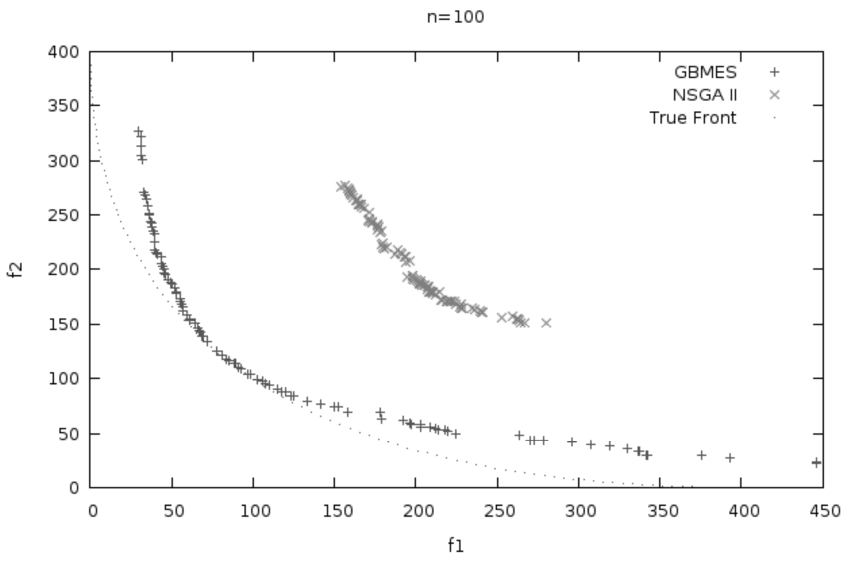


Figure 5.6: This graph shows the approximation of the Pareto front generated by GBMES and NSGA-II on Problem1 (with $n = 100$), after 3,000 function evaluations.

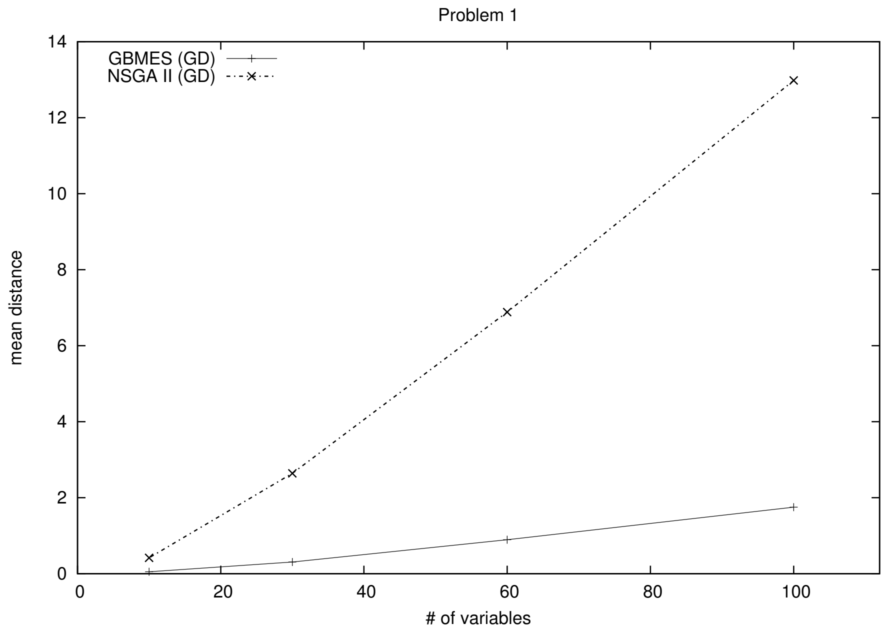


Figure 5.7: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 1, regarding the GD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

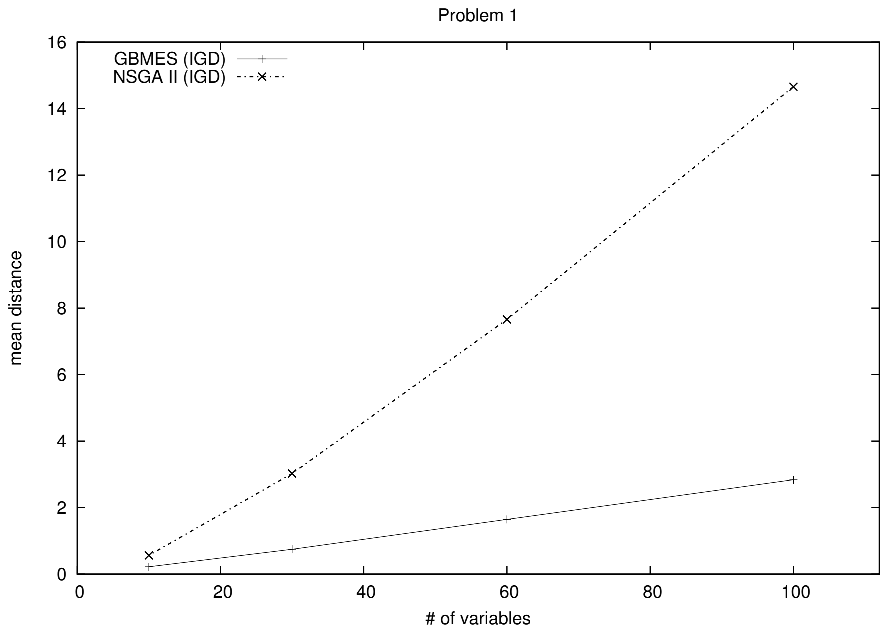


Figure 5.8: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 1, regarding the IGD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

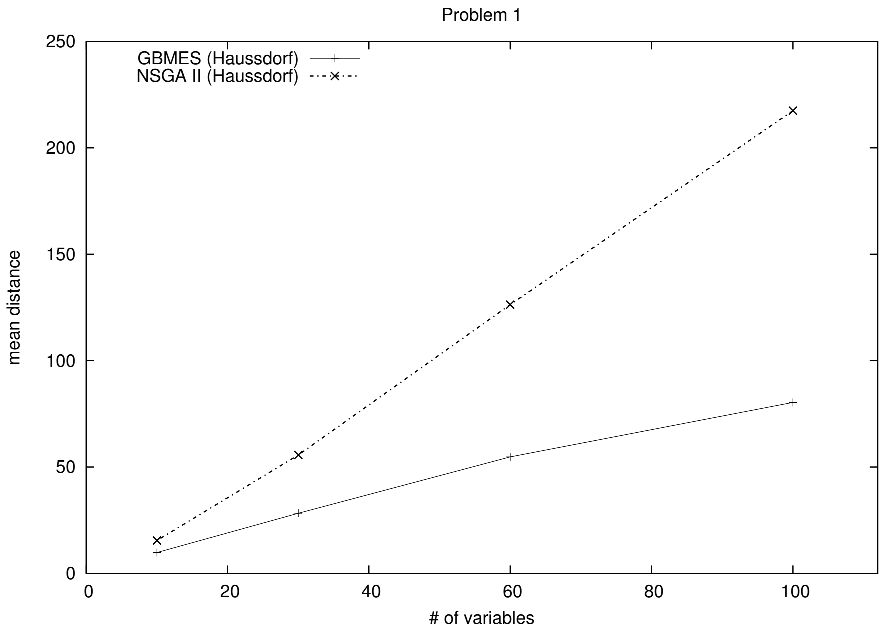


Figure 5.9: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 1, regarding Hausdorff distance, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

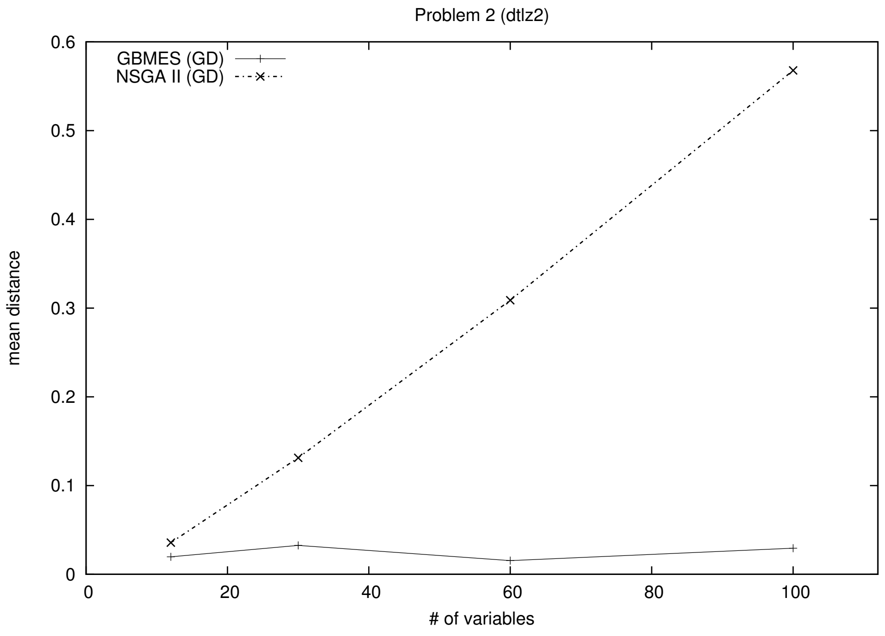


Figure 5.10: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 2, regarding the GD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

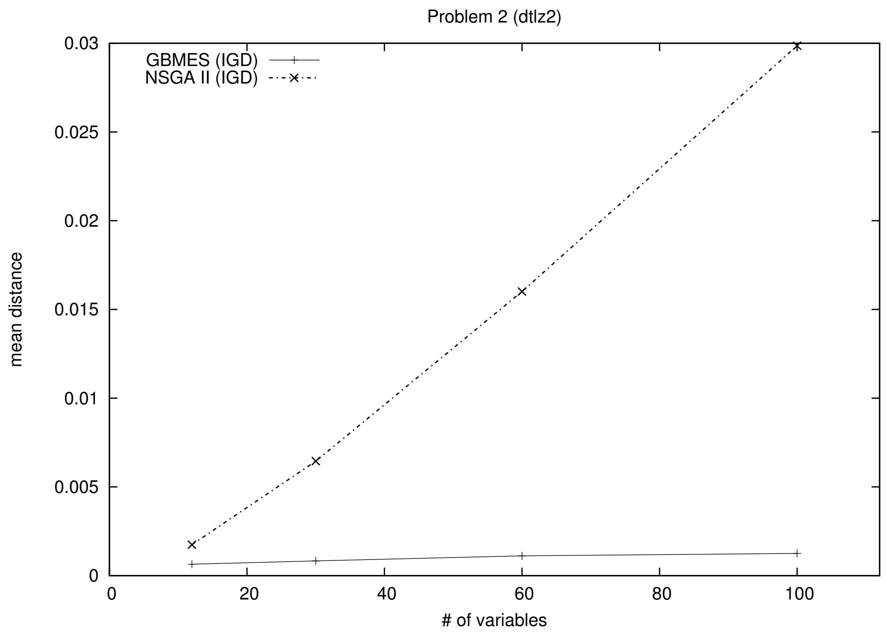


Figure 5.11: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 2, regarding the IGD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

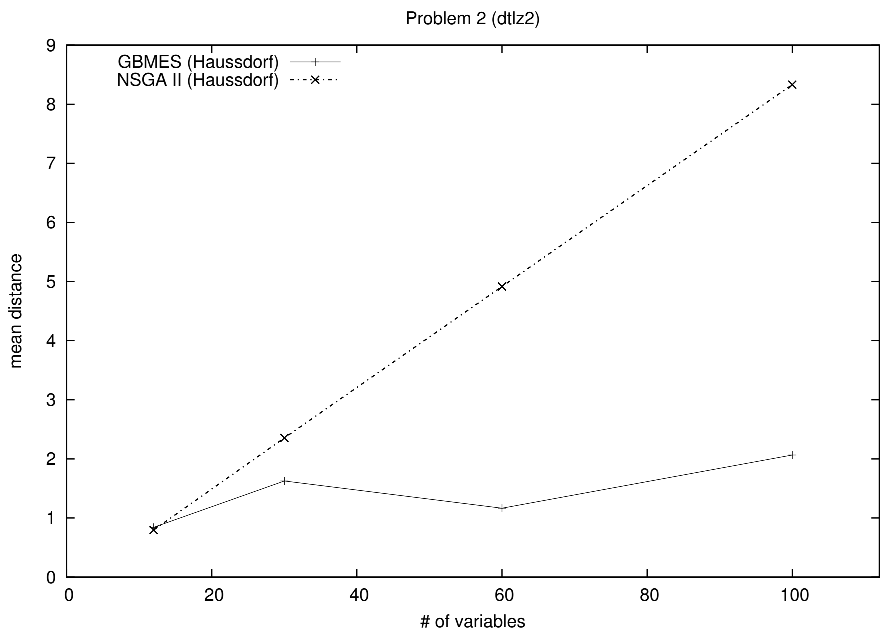


Figure 5.12: Graphical illustration of the performance of GBMES and the NSGA-II in Problem 2, regarding Hausdorff distance, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

Table 5.2: Comparison of results on Problem1, using $n = 10$, and performing 3,000 function evaluations. Statistics were gathered from 30 independent runs. The best results are shown in **boldface**.

GD			
Method	Best	Mean	Worst
GBMES	**0.023375**	**0.049261**	**0.103776**
NSGA II	0.242919	0.415592	0.539319

IGD			
Method	Best	Mean	Worst
GBMES	**0.092534**	**0.222116**	**0.549907**
NSGA II	0.367473	0.565376	0.863660

Spacing			
Method	Best	Mean	Worst
GBMES	0.004457	0.029559	0.081939
NSGA II	**0.002789**	**0.022826**	**0.070167**

Hausdorff distance			
Method	Best	Mean	Worst
GBMES	**3.493272**	**9.823990**	**21.668491**
NSGA II	7.022420	15.473644	25.665400

good as exploring only a, well chosen, few candidates. This is a direct motivation to use gradient-based search methods. *i.e.,* we aim to perform a more efficient exploration of potential solutions.

When hybridizing MOEAS with LS on continuous problems some difficulties arise; one is, for example, that the LS can consume all our budget of objective function evaluations (as was noticed in [66] for discrete problems). This situation supports the idea of performing a suitable selection process to determine which individuals will be modified by the LS mechanism. To the author's best knowledge, the work in which the ideas of elitism and local search hybridization were implemented, for the first time, is [64, 65]. Also, other important questions are: when to apply the LS into the MOEA, and how often to do it within the evolutionary search.

A suitable control of the local-search computational effort and the MOEA-search engine effort is subject of ongoing research. We focus here on studying this topic mostly for continuous and differentiable problems; studies in this topic are scarce since this is a difficult issue. For example, in [78] some attempts to control the probability of application of the LS are presented. That work focused on a particular MOEA—from the class of indicator-based MOEAs—and concluded that using these control mechanisms with a certain probability function (linear or quadratic) did not outperform the scheme when using a fixed *a priori* probability. Furthermore, this last scheme was even more stable than the others, in most of their test problems.

Table 5.3: Comparison of results on Problem2, using $n = 12$, and performing 3,000 function evaluations. Statistics were gathered from 30 independent runs. The best results are shown in **boldface**.

GD			
Method	Best	Mean	Worst
GBMES	**0.003234**	**0.019648**	**0.032110**
NSGA II	0.027392	0.035683	0.043271

IGD			
Method	Best	Mean	Worst
GBMES	**0.000421**	**0.000643**	**0.001017**
NSGA II	0.001422	0.001746	0.002200

Spacing			
Method	Best	Mean	Worst
GBMES	**0.000737**	**0.004314**	**0.009814**
NSGA II	0.001360	0.005380	0.032598

Hausdorff distance			
Method	Best	Mean	Worst
GBMES	**0.385372**	0.842908	1.335189
NSGA II	0.556158	**0.798111**	**1.170840**

In this point we can take a look on the studies made on discrete models, in order to analyze differences and common goals. To reduce the computational effort of the LS engine, Ishibuchi et al. proposed two strategies [66, 67]: First, reduce the number of individuals on which the local search is applied; and second, apply the LS just partially in a certain sense. The former idea leads to a parameter N_{ls}—explained below—and the latter leads to a parameter for early termination of the LS. They concluded experimentally that the values for these two parameters are related, since both contribute to help the search but they increase with the computational effort as well. Therefore, to control the resource spending, they choose to search over just some neighboring points instead of exploring all of them. This truncates the LS, in a certain way, since the LS engine is not exploiting the entire possible neighborhood. The analogy of this idea for the continuous case lies on the tolerance value for the line search, and on the number of times the descent movement is performed for each solution. Also, in gradient-based procedures these savings on function calls are related, for example, with a tolerance value in case of using a KKT-based stopping criterion—similar to the one we use throughout this work.

When limiting the number of individuals on which the LS has to be applied, we support the idea that the best individuals are the ones that should be pulled toward better regions. But, we expect a growth on the number of these best individuals as the evolutionary search

Table 5.4: Comparison of results regarding GD for both problems when using n =*original*,30, 60, 100, and after performing 3,000 function evaluations. The value corresponds to the mean over 30 independent runs. The *original* values adopted for n in each problem are those defined in Table 5.1. The best results are shown in **boldface**.

GD	Problem 1		Problem 2	
n	GBMES	NSGAII	GBMES	NSGAII
original	**0.0493**	0.4156	**0.019648**	0.035683
30	**0.3063**	2.6406	**0.032565**	0.131245
60	**0.8925**	6.8851	**0.015376**	0.308811
100	**1.7495**	12.9756	**0.029423**	0.567924

Table 5.5: Comparison of results regarding IGD for both problems when using n =*original*,30, 60, 100, and after performing 3,000 function evaluations. The value corresponds to the mean over 30 independent runs. The *original* values adopted for n in each problem are those defined in Table 5.1. The best results are shown in **boldface**.

IGD	Problem 1		Problem 2	
n	GBMES	NSGAII	GBMES	NSGAII
original	**0.2221**	0.5654	**0.000643**	0.001746
30	**0.7449**	3.0245	**0.000829**	0.006448
60	**1.6436**	7.6595	**0.001116**	0.016010
100	**2.8417**	14.6620	**0.001249**	0.029858

progresses. One idea to delimit this number is to fix a certain value, that we will call N_{ls}, and select this number out of the best individuals set. To perform this selection we want to take a spread sample of N_{ls} individuals; for this sake, using a crowding distance based selection would be helpful.

In order to state the control parameters for our gradient-based hybrids we can mention the ones presented in Table 5.7:

- N_{ls} determines the number of individuals that conform the LS pool.
- k is the frequency on generations to apply the LS.
- Tol is the tolerance value to finish the descent.
- $MaxIter$ determines the depth of the LS, which means the number of times the LS will be applied.

Table 5.6: Comparison of results regarding the Hausdorff distance for both problems when using n =*original*,30, 60, 100, and after performing 3,000 function evaluations. The value corresponds to the mean over 30 independent runs. The *original* values adopted for n in each problem are those defined in Table 5.1. The best results are shown in **boldface**.

Haus. dist	Problem 1		Problem 2	
n	GBMES	NSGA-II	GBMES	NSGA-II
original	**9.8240**	15.4736	0.842908	**0.798111**
30	**28.3035**	55.6760	**1.625643**	2.354311
60	**54.7381**	126.3136	**1.164723**	4.914662
100	**80.3792**	217.4507	**2.065171**	8.331932

Table 5.7: Control parameters for applying gradient-based LS.

N_{ls}	Number of individuals that conform the LS pool.
k	Frequency, in generations, of application of the LS.
Tol	Tolerance value to finish the descent.
$MaxIter$	Depth of the LS, which refers to the number of times the LS will be applied.

The parameter $MaxIter$ plays the role of an iterated LS. The parameter k has been assumed as 1 in most of the studies but has also been proposed with different values, for continuous problems, in some other algorithms [59, 67]. Here, we propose not to set this parameter as one, in order to let the evolutionary search keep its own "natural" process, but mostly to reduce the total cost for LS. In the works analyzed in Section 5.4 and Chapter 6 we use this idea and conducted experiments performing the search in this way, instead of doing it at each generation.

5.4 Integrating HCS into MOEAs

In this section we address the integration of the HCS into a given MOEA. One of the most prominent state-of-the-art MOEAs is the NSGA-II [31]. Since it has been found to be a very efficient and robust algorithm, it makes sense to take its operators when seeking for a hybridization with local searchers.

Several design aspects have to be decided in order to make this coupling. The first question that arises is: in which part of this well-known algorithm it is more appropriate to perform the LS? We discuss this issue further on, and also address the details about the interleaving of these two mechanisms—the global and the LS. We also support, with this proposal, our idea about investing local-search effort only on the best individuals of each generation. This is done in order to find specifically good individuals who will lead the entire population to better search regions during the search.

For this, we present first some modifications required on the standalone version of the HCS, in order to be efficiently coupled with an evolutionary algorithm, and discuss the cost of the procedure. Finally, we present two particular hybrids where NSGA-II and SPEA2 [150]—another state-of-the-art MOEA—are used as baseline MOEAs.

5.4.1 Adapting HCS for local search

The HCS is presented as standalone procedure in Algorithms 4 and 5—the goal in those cases was to generate an infinite sequence of candidate solutions, which is certainly not applicable when coupling it with a MOEA. To ensure enough resources for the global-search algorithm (the MOEA), it is rather advisable to stop the iteration of the local searcher after a few iterations—we denote this parameter by $maxiter$ inside the discussion. In case the HCS finds only a sequence of dominating solutions (*i.e.*, by the hill climber) merely the last dominating solution (denoted by x_d) has to be returned. This is done because the other intermediate solutions are all dominated by x_d; and they are thus not important for the current population of the MOEA. On the other case, when the sidestep is performed, that means that the iterates are 'near' to the (local) Pareto set; thus, the iteration can be stopped even before $maxiter$ is reached. The second modification of HCS that we suggest, compared to the standalone version presented above, is to perform the sidestep in each diversity direction which has been found during the LS. This is due to the fact that the sidestep is the most expensive part of the HCS (expensive in terms of function calls, see also the discussion below); and hence all accumulated information should be exploited. In that case, the modified HCS will return the dominating solution x_d (if not equal with the initial solution x_0) and also a maximum of $2^k - 2$ sidestep solutions in all diversity directions of x_d—depending on how many diversity directions of x_p have been found within the N_{nd} 'unsuccessful' trials.
Algorithm 15 shows such a modification of the HCS1 for $k = 3$. Hereby,

$$C_x(s_1, s_2, s_3),$$

where $s_i \in \{+, -\}, i = 1, 2, 3$, denotes the diversity cone at a point x. For instance, it is

$$y \in C_x(+, +, -)$$

if and only if

$$\{f_1(y) > f_1(x) \text{ and } f_2(y) > f_2(x) \text{ and } f_3(y) < f_3(x)\}$$

Finally, the algorithm requires the starting point x_0 and returns the set X_{new} which can consist of one candidate solution (*i.e.*, the result of the hill climber x_d) up to seven candidate

solutions (x_d plus candidates in all the six diversity directions (4.3) of x_d).
For HCS2, the modifications described above are much easier to handle: If the sidestep is performed (*i.e.*, if Equation (4.4) is false) the sidestep solutions can be chosen as

$$x_+^i := x_d + h_i q_i,$$

$$x_-^i := x_d - h_i q_i,$$

for all column vectors q_i of Q_K, which leads to $2k-2$ new candidate solutions.

```
Require: maximum number of iterations $maxiter$, rest as in Alg. 4
Ensure: set of candidate solutions $X_{new}$
 1: $L_1 := 0$, $L_2 := 0$, $L_3 := 0$
 2: $a_1 := 0 \in \mathbb{R}^n$, $a_2 := 0 \in \mathbb{R}^n$, $a_3 := 0 \in \mathbb{R}^n$
 3: $no\_a_1 := 0$, $no\_a_2 := 0$, $no\_a_3 := 0$
 4: $nondom := 0$
 5: $x_{1,0} := x_0$
 6: for $i = 1, 2, \dots, maxiter$ do
 7:     for $j = 1, 2, \dots, N_{nd}$ do
 8:         choose $x_2 \in B(x_{1,i-1}, r)$ at random
 9:         if $x_{1,i-1} \prec x_2$ then
10:             compute $t \in \mathbb{R}_+$ as in Alg. 4 (l. 6–10), set $x_{1,i} := x_2 + t(x_{1,i-1} - x_2)$.
11:             $nondom := 0, \quad a_1 = a_2 = a_3 = 0$
12:             continue
13:         else if $x_2 \prec x_{1,i-1}$ then
14:             comp. $t \in \mathbb{R}_+$ as in Alg. 4 (l. 11–13), set $x_{1,i} := x_{1,i-1} + t(x_2 - x_{1,i-1})$.
15:             $nondom := 0, \quad a_1 = a_2 = a_3 = 0$
16:             continue
17:         else
18:             $nondom := nondom + 1$
19:             if $x_2 \in C(x_{1,i-1}, -, -, +)$ then
20:                 $a_1 := a_1 + (x_2 - x_{1,i-1})/||x_2 - x_{1,i-1}||_\infty$
21:                 $no\_a_1 := no\_a_1 + 1$
22:                 $L_1 := \max(L_1, ||F(x_2) - F(x_{1,i-1})||_\infty/||x_2 - x_{1,i-1}||_\infty)$
23:             end if
24:             if $x_2 \in C(x_{1,i-1}, +, +, -)$ then
25:                 $a_1 := a_1 + (x_{1,i-1} - x_2)/||x_{1,i-1} - x_2||_\infty$
26:                 $no\_a_1 := no\_a_1 + 1$
27:                 $L_1 := \max(L_1, ||F(x_{1,i-1}) - F(x_2)||_\infty/||x_{1,i-1} - x_2||_\infty)$
28:             end if
```

Algorithm 15: HCS1, for use within MOEAs for $k = 3$.

```
29:             if $x_2 \in C(x_{1,i-1}, -, +, -)$ then
30:                 update $a_2, no\_a_1$, and $L_2$ analogue to lines 19 –22.
31:             end if
32:             if $x_2 \in C(x_{1,i-1}, +, -, +)$ then
33:                 update $a_2, no\_a_2$, and $L_2$ analogue to lines 24 –27.
34:             end if
35:             if $x_2 \in C(x_{1,i-1}, +, -, -)$ then
36:                 update $a_3, no\_a_3$, and $L_3$ analogue to lines 19 –22.
37:             end if
38:             if $x_2 \in C(x_{1,i-1}, -, +, +)$ then
39:                 update $a_3, no\_a_3$, and $L_3$ analogue to lines 24 –27.
40:             end if
41:         end if
42:     end for
43:     $X_{new} := \{x_{1,i}\}$                                    ▷ perform sidesteps and return
44:     if $no\_a_1 > 0$ then
45:         $v_1 := a_1/||a_1||_\infty, \quad h_1 := \epsilon_y/L_1$
46:         $x_+^{(1)} := x_{i,N_{nd}} + h_1 v_1, \quad x_-^{(1)} := x_{i,N_{nd}} - h_1 v_1$
47:         $X_{new} := X_{new} \cup \{x_+^{(1)}, x_-^{(1)}\}$
48:     end if
49:     if $no\_a_2 > 0$ then
50:         compute $x_+^{(2)}, x_-^{(2)}$ analogue to lines 44–47, set $X_{new} := X_{new} \cup \{x_+^{(2)}, x_-^{(2)}\}$
51:     end if
52:     if $no\_a_3 > 0$ then
53:         compute $x_+^{(3)}, x_-^{(3)}$ analogue to lines 44–47, set $X_{new} := X_{new} \cup \{x_+^{(3)}, x_-^{(3)}\}$
54:     end if
55:     return
56: end for
57: $X_{new} := \{x_{1,maxiter}\}$                                  ▷ return dominating solution
```

Algorithm 15: HCS1, for use within MOEAs for $k = 3$ (continued).

The HCS shows large potential when used within multi-objective memetic algorithms, mainly because it performs an efficient LS which starts with one point and ends not only with an improvement of this point, but also, with several candidates for spread. Figure 5.13 illustrates the application of the HCS as a local searcher over a population of three individuals. When the points are far, a hill climber movement (HC) is performed, and the hill climber with side step (HCS) is applied when the optima is 'almost' reached. The operator could (via *maxiter*) repeat the descent step—hill climber—several times until a sidestep is triggered. The cost and profit of this operator is discussed in the following sections of this chapter.

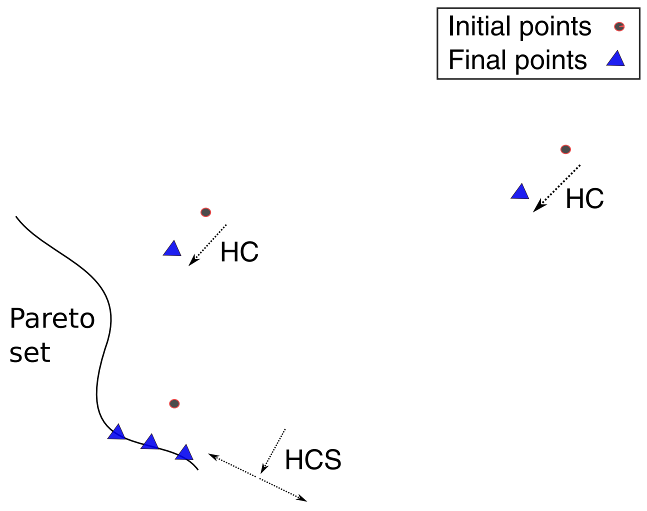


Figure 5.13: This figure shows the way to use the HCS as a LS operator, when coupled with a MOEA. The LS starts with a point and ends with an improved point, or with three points (in case of $k = 2$): the one obtained by the gradient-based descent and the other two obtained by sidesteps.

5.4.2 Cost of the local search

Crucial for the efficient usage of the HCS within a MOEA is the knowledge of its cost. Here we measure the cost of one step of the modified HCS as described above (*i.e.*, for $maxiter = 1$). Unfortunately, the different algorithms use different gradient information (quantitatively different) of the model. For sake of comparison, we measure the cost of the HCS in terms of the number of required function calls (to be more precise, we measure the running time for a function call and neglect the memory requirement). In other words, to measure HCS2 we have to find an equivalent in terms of function calls for the computation or approximation of the derivative $\nabla f(x)$, and of the second derivative $\nabla^2 f(x)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point x. If for instance automatic differentiation (AD) is used to compute the derivatives, we can estimate 5 function calls for the derivative call and $4 + 6n$ function call for the second derivative [47]. These values change when using finite differences (FD). If, for instance the forward difference quotient

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x_1, \dots, x_i + \delta_i, \dots, x_n) - f(x_1, \dots, x_n)}{\delta_i}, \quad i \in \{1, \dots, n\} \tag{5.4}$$

where $\delta_i \in \mathbb{R}_+$ is a small value, is used to estimate the gradient, then n function calls are required. The central difference quotient leads to more accurate approximations, but does in turn require $2n$ function calls (see [47]). A forward difference quotient approximation of the second derivative requires a total of n^2 function calls (and $2n^2$ or $4n^2$ function calls when using the central difference quotient, depending on how the rule is applied). Finally, we have to estimate the number of function calls required for a line search for the hill climber. Here, we take the value of 3 obtained by our observations.

Then, a call of HCS1 requires at least four function calls, from which one is for the LS around x_0. If the new candidate solution is either dominating or dominated by x_0—which is very likely at the early stages of the optimization process—the next point is found via line search resulting in 4 function calls due to our assumptions. The most expensive event of the HCS is the sidestep; when it is performed, this means that N_{nd} trial points have been generated around x_0. Then, it is still necessary to compute candidates in at most $2^k - 2$ directions, with a cost of one function call each (see (4.11)), leading to a total of

$$N_{nd} + 2^k - 2$$

function calls.
The HCS2 needs, for the realization of the hill climber, the gradients of all k objectives, the solution of a quadratic optimization problem (which we do not count here since k is typically low, and thus, the quadratic problem is easy to solve with standard techniques) and one line search. This makes $5k + 3$ function calls when using AD and $kn + 3$ function calls when using FD (here we assume the forward difference method). For the sidestep, k gradients and the second derivative of $\sum_{i=1}^{k} \alpha_i f_i(x_d)$ have to be computed, and further $2k - 2$ sidestep candidates are produced. This leads to $6n + 7k + 2$ function calls when using AD and to $n^2 + k(n + 2) - 2$ function calls when using FD.

Table 5.8 summarizes the cost of the different algorithms using the different conversion rules. Hereby, HC2 denotes the hill climber as presented in Algorithm 5 but without the sidestep operator (*i.e.*, for $\epsilon_{\mathcal{P}} = 0$). Table 5.9 gives the numerical values for $k = 3$, and $N_{nd} = 3$, and $n_1 = 10$ and $n_2 = 30$. It is obvious that FD should only be used for

Table 5.8: Cost of one step of the HCS measured in function calls. To convert the derivative calls in HCS2 into function calls we have used values based on automatic differentiation (AD) and finite differences (FD).

Method	Number of function calls required
HCS1	from 4 to $N_{nd} + 2^k - 2$
HC2 (AD)	$5k + 3$
HC2 (FD)	$kn + 3$
HCS2 (AD)	from $5k + 3$ to $6n + 7k + 2$
HCS2 (FD)	from $kn + 3$ to $n^2 + k(n + 2) - 2$

Table 5.9: Numerical values for the cost of the HCS algorithms for the settings (a) $n_1 = 10$, $k = 3$, $N_{nd} = 3$ and (b) $n_2 = 10$, $k = 3$, $N_{nd} = 3$. See Table 5.8 for details.

Method	No. of function calls required for $n_1 = 10$, $k = 3$, $N_{nd} = 3$	No. of function calls required for $n_2 = 30$, $k = 3$, $N_{nd} = 3$
HCS1	from 4 to 9	from 4 to 9
HC2 (AD)	18	18
HC2 (FD)	33	93
HCS2 (AD)	from 18 to 83	from 18 to 203
HCS2 (FD)	from 33 to 138	from 93 to 996

models with moderate dimensional parameter space, otherwise the cost of the HCS2 will get tremendous. On the other side, note that the cost of HCS1 is independent of n and thus relatively inexpensive, in particular in higher dimensions.

5.4.3 Combining both techniques

The question which remains open is how to integrate the HCS into a given MOEA, in order to obtain an efficient memetic strategy. Here we give the first steps to answer this problem, and propose hybrid versions for the state-of-the-art MOEAs NSGA-II and SPEA2. The numerical results obtained show that the combination is advantageous.

We start noticing that the modified HCS can be written in the shorthand form as

$$P_{HCS} = HCS\,(x_0), \tag{5.5}$$

where x_0 is a given point (*e.g.,* coming from the current population of the MOEA) and P_{HCS} is the output set. Given a probability p_{HCS} for the application of the procedure on an individual of a population, the operator can be defined set-wise as

$$P_{HCS} = HCS\,(P, p_{HCS}), \tag{5.6}$$

where P denotes a given population. By doing so, the HCS can be interpreted as a particular mutation operator, and thus it can, in principle, be integrated into any given MOEA with little effort. However, this should be handled with care since the efficiency of the resulting hybrid depends (among other issues) on (a) which elements of the population the HCS is applied to, and (b) the balance of the genetic search and the HCS. As an example for (a) we have observed that if the HCS is merely applied on elements of the external archive in a combination with SPEA2, that this 'elitist approach' has a negative effect on the diversity of the population, at least at early stages of the search. Even the application of the sidestep could not compensate this effect, since it is applied on a few, possibly closely located solutions. Problem (b) is another typical problem when designing memetic strategies (probably first reported in [67] in the context of multi-objective optimization), and in particular in our setting due to the relatively high cost of the HCS compared to classical mutation operators.
Most important for the effect and the cost of the HCS are the parameters $maxiter$ and N_{nd} (for HCS1). In general, it can be said that if both values of $maxiter$ and N_{nd} are high, the local improvement of a point x_0 will be nearly optimal (*i.e.*, the elements of P_{HCS} will be near to local solutions). This can be advantageous for uni-modal models but can in turn reduce the efficiency of the entire search algorithm for multi-modal models due to the high cost of the HCS and the relatively high chance that the search gets stuck in a local (and not global) solution. If the values of $maxiter$ and N_{nd} are low, the local improvements in one application of HCS will typically be sub-optimal. However, the choice of low values offers on the other hand two advantages: first, the HCS spends less function calls for unpromising starting points. That is apparently also the case for promising starting points but we have observed that it is advantageous to repeat the LS more often instead of spending the function calls in generating single solutions (future populations contain points which are at least as good as the point x_0 from the current population). The second advantage is that the population is not disturbed by drastic improvements of single solutions which may cause trouble in elitist strategies [75].
The next question is the choice of the probability p_{HCS} to apply the HCS. Due to the cost of the HCS, a low value seems to be advisable which also coincides with our observations. Further, we suggest not to apply the HCS in every generation in order not to disturb the efficient but highly sensitive interplay of the different operators of the MOEA (as done in [147]).

To summarize, we suggest low values for the parameters $maxiter$ and N_{nd} which influence efficiency and cost of one application of the HCS. Besides, it is convenient a low value for the probability p_{HCS} since the application of LS influences the overall cost of the entire search procedure. See the next section for particular choices of these values.
In the following, we propose two particular combinations where we use NSGA-II and SPEA2 as baseline MOEAs.

NSGA-II-HCS

As discussed above, crucial are the questions about when and to which elements the LS has to be applied on within a given MOEA. For NSGA-II, we suggest to perform the LS (*i.e.*,

HCS) only on the best individuals of a given generation. This is made in order to find leader solutions to pull the entire population to better solutions during the search. This exclusive search can be done since the diversity of the best (*i.e.*, non-dominated) solutions is typically quite high. Thus, it is likely to generate well-spread leader individuals from the beginning of the search which will help to pull the population towards the entire Pareto set.
Algorithm 16 shows the way to combine NSGA-II with HCS. Hereby, the procedures 'Fast Non-Dominated Sort', 'Crowding Distance Assignment' and 'Generate Child Population' are well known as parts of the NSGA-II—a thorough discussion can be found in [31].

```
procedure NSGA-II-HCS($N$,$G$, $p_{HCS}$, $s$)
    Generate Random Population P (size $N$).
    Evaluate Objective Values.
    Fast Non-Dominated Sort
    Crowding Distance Assignment
    Generate Child Population $P_{offs}$
    for $i := 1, \dots, G$ do
        Using $P := P \cup P_{offs}$:
        if  mod(i,s)==0  then
            LocalSearch($p_{HCS}$)
        end if
        Fast Non-Dominated Sort
        Crowding Distance Assignment
        Generate Child Population $P_{offs}$
    end for
end procedure

procedure LocalSearch($p_{HCS}$)
    for all $a \in P$ do
        if $\nexists b \in P$ such that $b \prec a$ then
            $A_a = HCS(\{a\}, p_{HCS})$
            $P := P \cup A_a$
        end if
    end for
end procedure
```

Algorithm 16: NSGA-II-HCS

Algorithm 16 applies the LS each s generations after the reproduction process. The LS is applied only to non-dominated individuals, and—due to the cost of the procedure—it is performed with a certain (low) probability. After having computed the improved solutions of LS, the regular operations of ranking and crowding are used as in NSGA-II.
Contrary to [126], where the LS has been applied after 75 percent of a given budget B of

function calls has been spent, we have observed that it is advantageous to apply the HCS in all stages of the search to pull the population permanently towards the Pareto set. In fact, we propose here that the LS should be evenly distributed over the run of the algorithm. This guideline and the choice to take only non-dominated solutions as starting points for the HCS have an implication on the rule to choose p_{HCS}: the number of non-dominated points (or rank 0 solutions) is typically very low at the beginning of the search, and increases later on. From a certain stage of the process, however, the number of non-dominated solutions is nearly constant (*i.e.*, equal to the population size). A constant value of p_{HCS} would hence lead to a permanent growth of the fraction of the LS within the memetic strategy, at least in the beginning of the search. To counteract to this effect it seems to be better to start with a relatively high probability $p_{\max}$ and to decrease this value during the search process until a prescribed (low) probability $p_{\min}$ is reached. This value is then chosen for the remainder of the algorithm's execution.
For the computations presented here, we have used the following strategy which is based on the above considerations: starting with the probability $p_{HCS}(0) := p_{\max}$, the LS probabilities for the subsequent generations are updated as follows

$$p_{HCS}(i) = \max\left\{ \frac{-2(p_{\max} - p_{\min})}{B} \sum_{j=1}^{i} fc(j) + p_{\max}\, ,\, p_{\min} \right\}, \tag{5.7}$$

where $fc(j)$ denotes the number of function calls spent in the j-th generation. Hereby, the first expression in (5.7) is a linear term in the number of function calls spent. Its value is $p_{\max}$ for zero function calls (*i.e.*, at the beginning of the search) and $p_{\min}$ for $B/2$. That is, after at least 50 percent of a given budget has been spent, the LS probability for future generations is constantly set to $p_{\min}$ (*i.e.*, $p_{\min}$ times the population size is the number of HCS calls one is willing to spend on average, per generation, at late stages of the search).

SPEA2-HCS

Unlike above, where NSGA-II is used as the baseline MOEA, we have observed that for a hybridization with SPEA2 it is not always beneficial to apply the HCS only to members of the archive which consists only of non-dominated solutions. This is because the archive can—in particular at early stages of the search—consist of few, and probably not well spread solutions (which changes with an increasing number of iterations). Thus, for a hybrid of HCS with SPEA2, we suggest to apply the LS operator to members of the mating pool, *i.e.*, also to dominated solutions. Consequently, we propose by the above discussion to set p_{HCS} constant since the size of the mating pool does not change. See Algorithm 17 for a pseudocode of SPEA2-HCS.

```
1: Generate initial population $P_0 \subset Q$ and set $A_0 := \emptyset$, $\bar{P}_0 := \emptyset$.
2: for $k = 0, 1, \ldots, N_{maxiter}$ do
3:     $\overline{P}_{k+1}$ := non-dominated solutions of $P_k \cup A_k$
4:     Set $A_{k+1}$ := non-dominated solutions of $\overline{P}_{k+1}$
5:     Calculate fitness values of individuals in $\overline{P}_{k+1}$
6:     Perform tournament selection in $\overline{P}_{k+1}$ to fill the mating pool
7:     Apply crossover, mutation and the LS operators (HCS) to the mating pool.
8:     Denote the resulting population by $P_{k+1}$.
9: end for
```

Algorithm 17: SPEA2–HCS

5.4.4 Numerical results

In order to demonstrate the benefit of using HCS within a given MOEA, we present and discuss some numerical results regarding the two specially designed memetic strategies. The particular MOPs we have used here, as test problems, are listed in Table 5.10 for (two and three objective) convex problems, and in Table 5.19 for multi-modal MOPs.

The comparisons presented here, show the two state-of-the-art MOEAs NSGA-II and SPEA2 against their corresponding hybrid variants NSGA-II-HCS and SPEA2-HCS. Since the Pareto sets of these test MOPs are located at the boundary of their corresponding domains, we have specially used a modification of HCS2 which acts just as a hill climber—in other words, the value $\epsilon_{\mathcal{P}}$ is set to 0; which means, that the search along the Pareto set is never performed. To agree with our notations, and to avoid confusions, we denote this special algorithm by $HC2$.

In order to evaluate the performance of the algorithms we have used the Generational Distance, the Inverted Generational Distance and the Two Set Coverage Measure—also described in Chapter 5—as indicators. All computations have been done using MATLAB[2].

Convex Models

First we consider the convex and thus uni-modal models CONV1 (with $k = 2$ objectives) and CONV2 (with $k = 3$ objectives) which are taken from [32]. The dimension of the parameter space is set as $n = 30$.

For both models we are interested in the unconstrained case (*i.e.,* where the Pareto set does not intersect with the boundary of the domain), and in the constrained case—also named here as the bounded case. We have chosen to take the domains

$$Q_u := [-5, 5]^{30}$$

and

$$Q_c := [-1, 1] \times [1, 2]^{29}$$

[2]`https://www.mathworks.com`

Table 5.10: Two and three objective convex problems adopted to test the memetic algorithms NSGA-II-HCS and SPEA2-HCS.

CONV1
$f_1(x) = (x_1 - 1)^4 + \sum_{j=2}^{n}(x_j - 1)^2$ $f_2(x) = \sum_{j=1}^{n}(x_j + 1)^2$
CONV2
$f_i(x) = \sum_{\substack{j=1 \\ j\neq i}}^{n}(x_j - a_j)^2 + (x_i - a_i)^4,\ i = 1, 2, 3$ $a_1 = (1, \ldots, 1) \in \mathbb{R}^n$ $a_2 = (-1, \ldots, -1) \in \mathbb{R}^n$ $a_3 = (1, -1, 1, -1 \ldots) \in \mathbb{R}^n$

for the unconstrained and the constrained model, respectively. Thus, the Pareto set of both problems[3] is located within the box $[-1, 1]^{30}$. The resulting models are denoted here by CONV1-U and CONV1-C for the two-objective problem, and CONV2-U and CONV2-C for the three-objective case.

Tables 5.12 to 5.17 show averaged numerical results obtained, on the convex models, using SPEA2 and NSGA-II and compare them against their respective memetic strategies. The parameter values used for their realization are displayed in Table 5.11. A budget of $B = 10,000$ function calls was spent in every case. It is worth noticing that B is chosen relatively low in order to obtain significant differences in the indicator values.

For the two cases where we hybridize the baseline MOEA with the HCS the following observations can be made: the values of the Set Coverage and the GD improve when the HCS1 is used as an additional local searcher. The values are even much better when using the gradient explicitly, *i.e.*, HC2 or HCS2 for the LS; in this case the improvement is roughly one order of magnitude. This big difference is certainly due to the fact that we are dealing with convex models. For this reason, the gradient based search—though much more costly than HCS1 for $n = 30$, see Table 5.8—leads to great improvements of given initial points since it does not get stuck at local solutions. Consequently, the population is pulled to the 'right' set at any stage of the optimization process. For IGD the results are not that conclusive; however, improvements can be observed.

One result of NSGA-II, and their memetic variants, is plotted in Figure 5.14 in consistency with the above discussion: Figure 5.14 (a) shows the effect of the HCS1 on a typical run, *i.e.*, better convergence and spread than the result of NSGA-II—due to the two search directions of HCS1. Convergence is on the other side much better in Figures 5.14 (b) and (c) where gradient information is used. When comparing the latter two figures, the effect of the sidestep gets visible: The spread in Figure 5.14 (c) is apparently better than in Figure 5.14 (b), where

[3]That is, considering the entire domain $Q = \mathbb{R}^{30}$

Table 5.11: Parameter values used for SPEA2 and NSGA-II and the memetic strategies SPEA2-HCS and NSGA-II-HCS on the convex problems.

	SPEA2-HCS		NSGA-II-HCS	
Parameters	*unbounded*	*bounded*	*unbounded*	*bounded*
N_{pop}	100	100	100	100
N_a	100	100	–	–
η_c	–	–	15	15
η_m	–	–	20	20
p_c	0.9	0.9	0.9	0.9
p_m	0.1	0.1	1/30	1/30
p_{HCS1}	0.2	0.2	–	–
s_{HCS1}	5	5	10	10
p_{HCS2}	0.1	0.1	–	–
s_{HCS2}	10	10	10	10
ε_y	5	5	5	5
$\varepsilon_{\mathcal{P}}$	0.0001	0.0001	0.0001	0.0001
r	0.1	0.1	0.1	0.1
maxiter	10	10	10	10
N_{nd}	5	5	3	3
tol	10^{-4}	10^{-4}	10^{-4}	10^{-4}

Table 5.12: Numerical results (GD and IGD) for SPEA2 and the memetic strategies SPEA2-HCS on the convex problems using dimension $n = 30$ and performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

	Indicators		
Problems		GD	IGD
CONV1-U	SPEA2(A)	3.5762(0.9190)	2.5469(0.3460)
	SPEA2-HCS1(B)	1.9399(0.5721)	2.2846(0.5483)
	SPEA2-HCS2(C)	0.1139(0.0505)	1.8377(0.9405)
	SPEA2-HC2(D)	0.1483(0.0463)	1.8792(0.9371)
CONV1-C	SPEA2(A)	6.8446(2.7245)	1.2491(0.1268)
	SPEA2-HCS1(B)	6.3508(1.9602)	1.1985(0.1225)
	SPEA2-HC2(C)	0.2800(1.1503)	0.4268(0.0069)
CONV2-U	SPEA2(A)	3.5762(0.9190)	2.5469(0.3460)
	SPEA2-HCS1(B)	1.9399(0.5721)	2.2846(0.5483)
	SPEA2-HCS2(C)	0.1139(0.0505)	1.8377(0.9405)
	SPEA2-HC2(D)	0.1483(0.0463)	1.8792(0.9371)
CONV2-C	SPEA2(A)	1.7887(0.4642)	0.4882(0.1086)
	SPEA2-HCS1(B)	1.6342(0.7454)	0.4341(0.1303)
	SPEA2-HC2(C)	0.4902(0.1789)	0.2774(0.1440)

Table 5.13: Numerical results, for the CONV1 problems, testing on Set Coverage the algorithms SPEA2 (A), and the memetic strategies SPEA2-HCS1 (B), SPEA2-HCS2 (C) and SPEA2-HC2 (D). The setting was for dimension $n = 30$, performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

Set Coverage CONV1-U	
$B \prec A$	$A \prec B$
0.7834(0.2726)	0.0984(0.1929)
$C \prec A$	$A \prec C$
0.9742(0.0755)	0(0)
$D \prec A$	$A \prec D$
0.9770(0.0753)	0(0)

Set Coverage CONV1-C			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.4881(0.4119)	0.2932(0.3604)	1(0)	0(0)

Table 5.14: Numerical results, for the CONV2 problems, testing on Set Coverage the algorithms SPEA2 (A), and the memetic strategies SPEA2-HCS1 (B), SPEA2-HCS2 (C) and SPEA2-HC2 (D). The setting was for dimension $n = 30$, performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

Set Coverage CONV2-U	
$B \prec A$	$A \prec B$
0.5602(0.2900)	0.1223(0.1405)
$C \prec A$	$A \prec C$
0.8963(0.2118)	0(0)
$D \prec A$	$A \prec D$
0.9893(0.0324)	0(0)

Set Coverage CONV2-C			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.6426(0.3622)	0.1796 (0.2752)	1(0)	0(0)

Table 5.15: Numerical results on GD and IGD for NSGA-II and the memetic strategies NSGA-II-HCS on the convex problems using dimension $n = 30$ and performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

	Indicators		
Problems		GD	IGD
CONV1-U	NSGA-II(A)	1.2847(0.2258)	1.6994(0.2843)
	NSGA-II-HCS1(B)	0.5661(0.0938)	1.1123(0.4191)
	NSGA-II-HCS2(C)	0.0606(0.0054)	1.5931(0.9827)
	NSGA-II-HC2(D)	0.0590(0.0048)	1.3167(0.8445)
CONV1-C	NSGA-II(A)	1.3747(0.1687)	1.0594(0.1027)
	NSGA-II-HCS1(B)	0.1143(0.0417)	0.3470(0.0661)
	NSGA-II-HC2(C)	0.0063(0.0041)	0.0386(0.0540)
CONV2-U	NSGA-II(A)	2.1814(0.4247)	0.4618(0.0652)
	NSGA-II-HCS1(B)	1.1465(0.1249)	0.4533(0.0784)
	NSGA-II-HCS2(C)	0.1041(0.0133)	0.3815(0.1582)
	NSGA-II-HC2(D)	0.1171(0.0136)	0.3374(0.1466)
CONV2-C	NSGA-II(A)	2.1165(0.3591)	1.4540(0.1725)
	NSGA-II-HCS1(B)	0.4333(0.1673)	0.5873(0.0794)
	NSGA-II-HC2(C)	0.0127(0.0114)	0.4232(0.1127)

Table 5.16: Numerical results, for the CONV1 problems, testing on Set Coverage the algorithms NSGA-II (A), and the memetic strategies NSGA-II-HCS1 (B), NSGA-II-HCS2 (C) and NSGA-II-HC2 (D). The setting was for dimension $n = 30$, performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

Set Coverage CONV1-U	
$B \prec A$	$A \prec B$
0.9516(0.0977)	0.0218(0.0597)
$C \prec A$	$A \prec C$
0.9412(0.0825)	0.0032(0.0155)
$D \prec A$	$A \prec D$
0.9418(0.1144)	0(0)

Set Coverage CONV1-C			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
1(0)	0(0)	1(0)	0(0)

Table 5.17: Numerical results, for the CONV2 problems, testing on Set Coverage the algorithms NSGA-II (A), and the memetic strategies NSGA-II-HCS1 (B), NSGA-II-HCS2 (C) and NSGA-II-HC2 (D). The setting was for dimension $n = 30$, performing 10,000 function calls. Statistics were gathered from 30 independent runs; average values are presented, showing the standard deviation in parentheses.

Set Coverage CONV2-U	
$B \prec A$	$A \prec B$
0.8433(0.1572)	0.0103(0.0332)
$C \prec A$	$A \prec C$
0.9397(0.1233)	0(0)
$D \prec A$	$A \prec D$
0.9633(0.1035)	0(0)

Set Coverage CONV2-C			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
1(0)	0 (0)	1(0)	0(0)

Table 5.18: Cost of the HCS variants within SPEA2-HCS in one run on CONV1-U (the run which produced the result displayed in Figure 5.14).

Method	Hill climber calls (no sidestep)	Sidestep calls (w or w/o sidestep)	Function calls (total)
HCS1	7	68	2229
HC2	69	0	1262
HCS2	51	20	6145

the solutions fall into clusters.

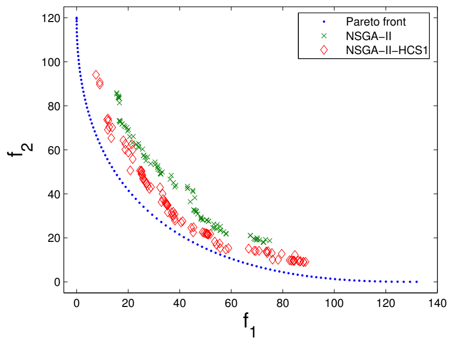
Table 5.18 gives an impression on the overall cost of the HCS within the search procedure. The table shows the amount of calls of the hill climber and the sidestep procedures, of HCS, when used within SPEA2-HCS to obtain the results shown in Figure 5.14. In this case, HCS1 used 22 percent of the total budget B of SPEA2-HCS1. The relatively large number of sidestep calls is due to the low value of N_{nd} ($= 3$). Larger values of N_{nd} would result in less sidestep calls and in turn more hill climber calls. The cheapest LS operator is HC2; only a portion of 13 percent of the function calls is used, since this operator merely computes the gradients to perform the hill climber. On the other hand, the sidestep operator is much more costly since the second derivatives are involved. Table 5.18 shows that, for HCS2, this LS operator spent 61 percent of B. The better result in Figure 5.14 (c) compared to Figure 5.14 (b) in terms of spread, was obtained by merely 20 sidestep calls which, however, used quite a lot of function calls for this.

To conclude, it can be said that on the convex models (two and three objectives, and also constrained and unconstrained) a combination of the two baseline MOEAs with the HCS variants improves the overall performance of the search. On the other hand, considerations of the cost of the operators show that its application should be handled with care, since the HCS can use a big portion of the budget, resulting therefore, in a risk to decrease the overall performance of the algorithm (note that we have assumed B to be constant).

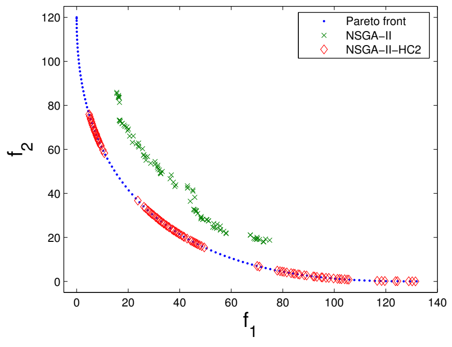
DTLZ for Three Objectives

To continue with the numerical analysis of the performance for the memetic algorithms, we consider three MOPs from the widely used benchmark DTLZ [29]. The problems DTLZ1 and DTLZ3 are highly multi-modal and thus very difficult for LS methods, in particular for gradient-based ones. Problem DTLZ2 is also a difficult problem but with a moderate multi-frontal geometry. The description of these MOPs is found in Table 5.19. We have set $n = 30$ for the dimension of the parameter space, $k = 3$ objectives, and the domain $Q = [0, 1]^{30}$ for all the DTLZ models adopted in our experiments.

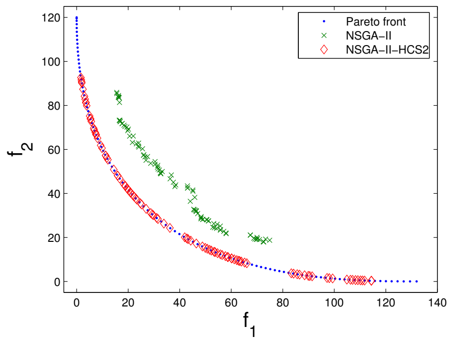
Tables 5.21 and 5.22 show averaged numerical results obtained by the MOEAs and its memetic variant. Hereby, we have used the parameter values shown in Table 5.20 and a



(a) NSGA-II vs. NSGA-II-HCS1



(b) NSGA-II vs. NSGA-II-HC2



(c) NSGA-II vs. NSGA-II-HCS2

Figure 5.14: Numerical result for CONV1-U using NSGA-II and the three memetic strategies NSGA-II-HCS1, NSGA-II-HC2, and NSGA-II-HCS2. The best result (spread *and* convergence) in this case was obtained by SPEA2-HCS2.

budget of $B_1 = 100,000$ function calls for the multi-modal models DTLZ1 and DTLZ3 and a budget of $B_2 = 10,000$ for the uni-modal model DTLZ2—this has again been done in order to prevent too small values of the indicators. The conclusions which can be drawn from the results are not as straightforward as for the convex case. While a similar trend as for the convex case can be observed for DTLZ2, this does not hold for the multi-modal models. Apparently, the two memetic versions which use HC2 can not compete with their baseline MOEAs. However, to be fair the construction of the models already suggests that gradient information is worthless. Thus, it is rather a question of the choice of the model than an indication of a general failure of the HC2. On the other hand, SPEA2-HCS1 outperforms its baseline MOEA—SPEA2—significantly on these highly multi-modal models. Such an improvement, however, can not be observed from NSGA-II-HCS1 and NSGA-II. The latter MOEA is already performing very well in these MOPs. As we noticed before, these problems make it hard for any LS strategy to improve the overall performance; besides, such an operator (the HCS in particular) causes an extra computational cost, which diminishes the resources invested in the MOEA execution.

Based on the numerical results presented in this section, it can be said that both variants of the standalone HCS accomplish their task; *i.e.*, they are capable of moving toward and along the Pareto set. This implies that, by using the HCS, entire connected components of the (local) Pareto set can be explored after starting with one single solution. Furthermore, it has been shown that the hybridization of the HCS, with a given MOEA, can improve the overall performance of the baseline MOEA adopted. More precisely, satisfying results have yet been obtained for uni-modal MOPs. The results on multi-modal models, however, do not allow us to draw general conclusions. Due to the relative high cost of the HCS and the natural handicap of LS methods for multi-modal problems, the balance of local and genetic search (such as for DTLZ1 and DTLZ3) is a challenging task. In order to handle this in an efficient manner, adaptive strategies are required, and further research should be done in that direction. An alternative for dealing with this problem, together with more insights about this, are presented in the next chapter.

Table 5.19: Three-objective problems adopted to test the memetic algorithms NSGA-II-HCS and SPEA2-HCS. Hereby, $\tilde{k} = n - k + 1$.

DTLZ1 $f_1(x) = \frac{1}{2}x_1x_2 \ldots x_{k-1}(1 + g(x))$ $f_2(x) = \frac{1}{2}x_1x_2 \ldots (1 - x_{k-1})(1 + g(x))$ $\vdots$ $f_{k-1}(x) = \frac{1}{2}x_1(1 - x_2)(1 + g(x))$ $f_k(x) = \frac{1}{2}(1 - x_1)(1 + g(x))$ $g(x) = 100\left[\tilde{k} + \sum_{i=k}^{n}(x_i - \frac{1}{2})^2 - \cos(20\pi(x_i - \frac{1}{2}))\right]$ $0 \leq x_i \leq 1,\ i = 1, \ldots, n$
DTLZ2 $f_1(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2}) \ldots \cos(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $f_2(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2}) \ldots \sin(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $\vdots$ $f_{k-1}(x) = \cos(\frac{x_1\pi}{2})\sin(\frac{x_2\pi}{2})(1 + g(x))$ $f_k(x) = \sin(\frac{x_1\pi}{2})(1 + g(x))$ $g(x) = \sum_{i=k}^{n}(x_i - \frac{1}{2})^2$ $0 \leq x_i \leq 1,\ i = 1, \ldots, n$
DTLZ3 $f_1(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2}) \ldots \cos(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $f_2(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2}) \ldots \sin(\frac{x_{k-1}\pi}{2})(1 + g(x))$ $f_{k-1}(x) = \cos(\frac{x_1\pi}{2})\sin(\frac{x_2\pi}{2})(1 + g(x))$ $f_k(x) = \sin(\frac{x_1\pi}{2})(1 + g(x))$ $g(x) = 100\left[\tilde{k} + \sum_{i=k}^{n}(x_i - \frac{1}{2})^2 - \cos(\alpha\pi(x_i - \frac{1}{2}))\right]$ $\alpha = 20$ $0 \leq x_i \leq 1,\ i = 1, \ldots, n$

Table 5.20: Parameter values used for SPEA2 and NSGA-II and the memetic strategies SPEA2-HCS and NSGA-II-HCS on the DTLZ functions.

Parameters	SPEA2-HCS	NSGA-II-HCS
N_{pop}	200	200
N_a	100	–
η_c	–	15
η_m	–	20
p_c	0.9	0.9
p_m	0.01	1/30
p_{HCS1}	0.3	–
p_{HCS2}	0.3	–
s	10	10
ε_y	1	5
$\varepsilon_{\mathcal{P}}$	0.0001	0.0001
r	0.05	0.1
maxiter	5	10
N_{nd}	5	3
tol	10^{-4}	10^{-4}

Table 5.21: Numerical results of SPEA2 and the memetic strategies SPEA2-HCS on the DTLZ benchmarks using dimensions $n = 30$, $k = 3$, and performing 100,000 function calls. Statistics were gathered from 30 independent runs.

	Indicators		
Problems		GD	IGD
$DTLZ1$	SPEA2(A)	22.7984(1.6658)	2.4303(0.4716)
	SPEA2-HCS1(B)	12.2165(3.9738)	1.5389(0.2132)
	SPEA2-HC2(C)	48.06789(7.5157)	1.3770(0.4404)
$DTLZ2$	SPEA2(A)	0.0529(0.0102)	0.0011(0.0001)
	SPEA2-HCS1(B)	0.0342(0.0404)	0.0007(0.0001)
	SPEA2-HC2(C)	0.0557(0.0164)	0.0012(0.0001)
$DTLZ3$	SPEA2(A)	218.3484(9.9639)	10.3885(2.2814)
	SPEA2-HCS1(B)	28.1974(3.8434)	3.2152(0.3890)
	SPEA2-HCS2(C)	198.8947(27.5283)	3.6925(0.7106)

Set Coverage DTLZ1			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.9789(0.0286)	0.0157(0.0227)	0.6925(0.2840)	0.2100(0.2710)

Set Coverage DTLZ2			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.7820(0.0779)	0.0040(0.0089)	0.2060(0.0888)	0.3660(0.1740)

Set Coverage DTLZ3			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
1(0)	0(0)	0.7905(0.0862)	0.0877(0.0586)

Table 5.22: Numerical results of NSGA-II and the memetic strategies NSGA-II-HCS on the DTLZ benchmarks using dimensions $n = 30$, $k = 3$, and performing 100,000 function calls. Statistics were gathered from 30 independent runs.

	Indicators		
Problems		GD	IGD
$DTLZ1$	NSGA-II(A)	8.5024(1.2081)	1.5998(0.1530)
	NSGA-II-HCS1(B)	10.4444(1.1505)	1.6856(0.2202)
	NSGA-II-HC2(C)	18.1555(4.6705)	1.0096(0.1899)
$DTLZ2$	NSGA-II(A)	0.0363(0.0041)	0.0022(0.0002)
	NSGA-II-HCS1(B)	0.0293(0.0033)	0.0018(0.0001)
	NSGA-II-HC2(C)	0.0097(0.0039)	0.0005(0)
$DTLZ3$	NSGA-II(A)	17.6126(3.5630)	3.2696(0.7721)
	NSGA-II-HCS1(B)	16.9381(2.9723)	2.9951(0.6861)
	NSGA-II-HC2(C)	24.6485(3.6724)	2.5127(0.4320)

Set Coverage DTLZ1			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.1610(0.1376)	0.6035(0.2565)	0.4885(0.1354)	0.3960(0.1627)

Set Coverage DTLZ2			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.7125(0.1869)	0.1235(0.0790)	0.9715(0.0284)	0.0105(0.0116)

Set Coverage DTLZ3			
$B \prec A$	$A \prec B$	$C \prec A$	$A \prec C$
0.5010(0.3365)	0.3965(0.3243)	0.4435(0.3225)	0.4550(0.2857)

"This principle is so perfectly general that no particular application of it is possible."

George Pólya

6 Seeking a Generic Procedure

In this chapter we suggest and investigate a generic way to introduce gradient-based information, as a means to improve the search process performed by a MOEA. We present ideas that can be easily incorporated into any MOEA, and provide some guidelines regarding the use of our proposed approach. This is an attempt to propose an adaptive mechanism that allows the local and global search strategies to dynamically interleave. We based the control mechanism on the observation of the evolutionary search efficiency; in this way, the algorithm can automatically assign more or less resources either to the local or to the global search procedure, as deemed necessary.

As a matter of example, in the sequel we present a particular algorithm (Algorithm 18) and then use it to discuss several issues. We also test this algorithm (in Section 6.3) over unconstrained two-objective optimization problems—since just two gradients can effort-less be combined into a common descent direction. However, the control part and the rest of the analysis could be used for problems with a higher number of objectives. To show our proposed coupling, we chose the widely used NSGA-II [31] as our global search engine. Nevertheless, the coupling with other MOEAs is also possible as will be later discussed. In the following, we present in Algorithm 18 our simple version of a MOEA hybridized with gradient information.

The parameters N and G in Algorithm 18, represent the population size and the maximum number of generations, respectively. The necessity for the module operation[1] in line 15, and the parameter k_g will be discussed later in this chapter. Again, the procedures "Fast Non-Dominated Sort", "Crowding Distance Assignment" and "Generate Offspring Population" correspond to the well-known components of the NSGA-II.

Algorithm 18 places the LS inside the NSGA-II just after the reproduction and the ranking-crowding process. The LS is applied only to non-dominated individuals, but not to all of them. Our control mechanism (global vs. local) uses information that has already emerged from the original ranking process.

[1] $a \equiv b \pmod{m}$ is that both numbers a and b leave the same residue when divided by m.

```
procedure GH-NSGAII($N$,$G$)
    Generate a Random Population $P$.
    Evaluate Objective Function Values.
    Fast Non-Dominated Sort
    Crowding Distance Assignment
    for $i \leftarrow 1, \ldots, G$ do
        Generate Offspring Population $P_{offs}$
        Set $P \leftarrow P \cup P_{offs}$:
        Fast Non-Dominated Sort
        Crowding Distance Assignment
        LocalSearch($i$, $P$)
    end for
end procedure

procedure LocalSearch($i$, $P$)
    if $i \equiv 0 \pmod{k_g}$ then
        Compute $e$ as in equation (6.1).
        Form the set $E$ taking $e$ individuals
        randomly selected out of $R_1(P)$.
        for all $a \in E$ do
            if local improvement is possible then
                Apply line search to obtain $a'$.
                Replace $a \leftarrow a'$.
                Set $a' \in R_1(P)$.
                Set the crowding distance of $a'$ as $\infty$.
            end if
        end for
    end if
end procedure
```

Algorithm 18: Hybrid GH-NSGAII

6.1 Multi-objective Line Search

6.1.1 Movement direction

We already discussed (in Chapter 3) the concept of multi-objective descent direction $\nu \in \mathbb{R}^n$. Several methods, that we previously discussed, to compute this directions are currently available. We have methods which calculate the greedy direction [42, 107], methods which compute the entire set of descent directions [53] [8], and our proposed HCS method [86] (Section 4.2) that performs two kinds of movements: towards and along the front. In addition, moving in a direction along the Pareto front is also possible using gradient-based continuation methods such as those in [58] [119] [52] or those without estimation of the gradient as in [86]. Furthermore, it is possible to perform directed movements not only towards and along the Pareto front, but also to any desired direction in the objective space; for this regard, the DS method [118] (discussed in Section 4.3) has the advantage of performing movements along determined paths in objective function space.

A descent direction particularly convenient is the one used to apply the line search in Algorithm 18 (line 21). For the two-objective case, we calculate the descent direction $\overline{\nabla}_x$ using equation (3.3), and we obtain the individual x' as

$$x' = x + t_x \overline{\nabla}_x.$$

The advantage of using the formula (3.3) for two objectives (over other currently available for multi-objective descent directions [107, 42, 8]) is that no additional effort is needed for its computation itself. While all the other alternatives need to solve a quadratic or linear optimization problem for each point, we emphasize that for two-objective problems just an arithmetic operation (a geometrical "average") over the gradients is necessary. This could be attractive for the algorithm designer because no additional code for solvers has to be added, and the procedure can be directly plugged into the MOEA adopted.

Once the movement direction is set, other issues come to our attention. Computing an optimal step length and applying a suitable stopping criterion are necessary in order to finish the design of the local searcher.

6.1.2 Step length control

Once the multi-objective descent direction ν is set for a specific point $x \in \mathbb{R}^n$, we can just define the new line search function as

$$\begin{aligned} f_\nu^i \ &: \ \mathbb{R} \longrightarrow \mathbb{R} \\ &\quad t \longmapsto f_i(x + t\nu). \end{aligned}$$

It is worth noticing that the computations of the step length of f_ν^i is again a multi-objective problem. In Figure 6.1 it is possible to observe that t_i is the local minimum corresponding to the f_ν^i function ($i \in \{1, 2, 3\}$); therefore, there are k functions (over one single variable) to

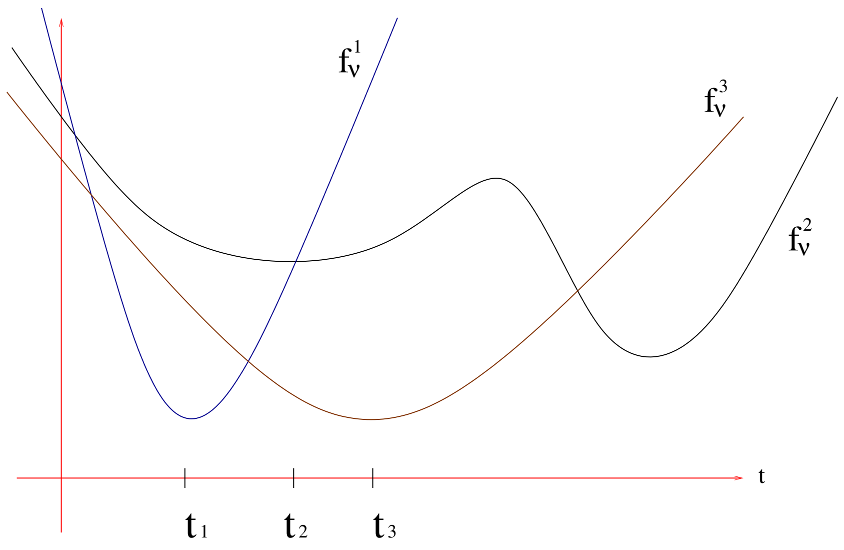


Figure 6.1: Simultaneous line search for three functions along the direction v.

be simultaneously minimized in the same neighborhood. Even when solving separately the k line searches (getting t_1, t_2 and t_3 as in the figure), it is not possible to say which is the suitable step length for all of them. Nevertheless, since not all the functions are in conflict with each other—at every point—it is possible to find regions of common improvement; for example, the region before the first local optimum appears, guarantees that all the functions have a certain improvement (decrease).

Since an exact step length calculation is not possible in this case, the use of inexact methods is a good option. One practical choice (suggested by [42]) is to adapt the widely known Armijo-Goldstein rule to the multi-objective case, and accept any step length t that holds

$$F(x + tv) \leq F(x) \; - \; c\, t\, J_{F(x)}\, v,$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ is the multi-objective function and $J_{F(x)} : \mathbb{R}^n \to \mathbb{R}^n$ is the Jacobian matrix of F at x. With a suitable initial step length, this method is easily applicable. However, this is not an efficient approach in general.

These inexact methods (such as Wolfe conditions, Armijo conditions, etc.) have been widely studied in mathematical programming for single-objective optimization, but their application in the multi-objective case is still an open problem. Ensuring convergence and the study of speed of convergence for these methods are important issues to address, when building an efficient interleaving between MOEAs and line search. The use of further heuristics and mathematical analysis to deal efficiently with this multi-objective problem (the step length calculation) are an open branch of study, and are also accessible for future experimental research.

As a practical choice, in this proposed algorithm, to estimate t_x, we use an Armijo-like rule starting from a size $t_{\max}$, and we reduce $t \leftarrow t/2$ at each iteration, until $f_i(x') \leq f_i(x)\ \forall i$ (or the Armijo condition) is fulfilled. The procedure is sensitive to this $t_{\max}$ parameter as any other line-search procedure is to the initial step length choice.

6.1.3 Stopping criterion

Since it could happen that a certain point x is already close to a Pareto optimal point, it is necessary to set a stopping criterion—in order not to apply the LS on certain points. In Algorithm 18, the stopping criterion is applied in line 20. As we stated above, most of the gradient-based descent directions can be automatically tested for a stopping criterion. These criteria are, in general, inspired on the KKT optimality conditions [81].

For the particular case of two objectives, it is very easy to set a stopping criterion to apply the steepest descent. The condition is related to certain small tolerance $0 < \epsilon_{\text{tol}} < 0.01$ and is given by the following rule: If

$$\langle \nabla_1, \nabla_2 \rangle < -1 + \epsilon_{\text{tol}}$$

holds, then, no LS movement is performed for the point x.

6.2 Balancing Resources

Since there is no *a priori* knowledge about the benefits of using LS (within the evolutionary search) for a specific problem, it is not convenient to previously determine—as it is done, for example, in two-stage algorithms—a specific amount of resources to be devoted for the global, and for the local part of the search. Nevertheless, it is a promising idea to incorporate LS, as a method to refine solutions, only at the end (as suggested in [51]); but this leads again to a two-stage algorithm. In this case, an adaptive switch mechanism between the two stages is desired, and its design is still an open research problem.

An adaptive mechanism to control the use of LS is advised in order to produce efficient algorithms, but this is, by itself, a non-trivial problem. Because of the high cost, in general, of the application of LS, such balance mechanisms must be capable of determining when the gradient method outperforms the pure evolutionary search during the solution of a specific problem. This control should be based on a mechanism to determine when the evolutionary search is not producing improvements and then allow more resources for the solutions refinement. In the following, we discuss our proposal to control the application of LS, based on the above observations.

In chapter 3 we discussed that the descent cone is wider for farther points than for those close to KKT points. In particular, the descent cone shrinks down as the search gets closer to the optimum. Thus, the probability to locally improve the point x with any kind of perturbation is high when x is far, being this the case—most of the time—during the first stages of the evolutionary search, when many points are likely to be replaced by non-dominated ones. On the other hand, when a point highly improves, it is unlikely that new points, stochastically generated, could replace it—since the descent cone shrinks down—and it will remain in this state for several iterations. This is the case when a trusty movement towards a descent direction is more necessary, and, therefore, the use of LS becomes cost-effective. However, since different MOPs have different fitness landscapes, we do not really know if this state is reached at the end of the search (or at any other stage, for that sake). Thus, it becomes desirable to have an adaptive mechanism that can recognize when the use of LS is useful and when it will produce no gains with respect to the use of the global search engine alone.

On the other hand, a well studied fact about the efficiency of MOEAs [62][69] is that when a large portion of the population becomes non-dominated, the selection mechanism of the evolutionary search fails[2],—possibly leading to a random behavior. This situation is easily detected during the execution of the algorithm, mostly within MOEAs that use Pareto ranking. Following this reasoning, we propose a control mechanism based on the number of non-dominated points at each iteration. The idea is that when the evolutionary search is promising, the LS is scarcely used. On the other hand, when the number of non-dominated solutions grows, our method allows the LS mechanism to be applied more often.

Then, the control parameter e that we propose should follow the rule:

$$e = \begin{cases} 0 & \text{if} \quad |R_1(P)| < (0.1)(N) \\ \left\lfloor \frac{|R_1(P)|}{(0.1)(N)} \right\rfloor & \text{otherwise.} \end{cases} \tag{6.1}$$

where

$$|R_1(P)|$$

is the cardinality of the subset $R_1(P)$ (lines 18 and 23 of Algorithm 18, in page 114) formed by the elements in P whose Pareto rank is one. N is the population size and $\lfloor \cdot \rfloor$ represents the *floor* function.

The idea behind the parameter k_g (line 15 of Algorithm 18) is that the LS is improving only a few individuals at a certain moment and, since we have a population-based approach (*i.e.,* the MOEA acting as our global search engine) we should let the evolution do its job. In other words, we are expecting that most of the effort will come from the global procedure, and that the gradient-based LS will only refine the solutions generated by the global engine. A deeper study over this parameter is left as future work. During the experiments done in this thesis, the best behavior was obtained with values between 5 and 10; but it seems that this parameter will be problem dependent, in a similar way that it was concluded for the discrete domain case [66]. Not applying the LS at each iteration will help us to make its cost more affordable. At the same time, this will allow us to push the population of the MOEA towards better solutions. Thus, parameter k_g determines how often do we apply the LS. We believe that this parameter must be varied adaptively during the search as a second control mechanism for balancing resources. A more in depth study of this topic is left as future work.

[2]This behavior is because the selection in EAs is normally based on the Pareto dominance relation.

6.3 Numerical Results

For assessing the performance of our proposed approach, and in order to deal with unconstrained problems, we used the modified Zitzler-Deb-Thiele (ZDT) problems presented in [130] and [129] (with the difference that to use our method we do not need the twice differentiability property). These test functions are defined in Table 6.1 (ZDT5 was not included because it is defined over a binary space). All these experiments were coded using the programming language C[3].
The parameters used in our experiments were the following:

- Population size $N = 100$,
- $\epsilon_{\text{tol}} = 0.0001$,
- $k = 2$, but similar results were obtained for values under 10, and
- $t_{\max} = 2$.

Regarding the computational costs involved in practice, we are assuming that the user estimates the gradients of the objective functions using an evaluations-saver method such as Automatic Differentiation [25]. This sort of approach introduces significant savings in the computational cost of the gradient values.

It is worth noting that we adopted the previously described test problems both, with their original dimensionality and, as a second instance, with a higher number of decision variables. The specific dimensionalities adopted are indicated in Table 6.2.

We compared the plain NSGA-II with respect to our proposed hybrid approach. All the experiments were run until reaching a certain number of function evaluations (5,000 for ZDT1, ZDT2 and ZDT3; 20,000 for ZDT4 and ZDT6) which corresponds to the moment when certain reasonable proximity to the front has been reached (see the right hand-side plots of Figures 6.7 to 6.11). Then, we allowed the algorithms to perform twice these numbers of evaluations. It is worth noting that the same number of evaluations were adopted for the two cases: for the original test problems and for their extended versions (see Table 6.2).

Tables 6.3 to 6.6 show the mean and the standard deviations over 30 independent runs, regarding the indicators *Set Coverage* and IGD. The calculation of IGD was done using the parameter $p = 1$. Table 6.3 shows that, for the first number of evaluations, the hybrid approach almost always set-covers the plain MOEA (its values are close to one), while the plain NSGA-II almost never set-covers the hybrid approach (its values are close to zero). The only two exceptions to notice are ZDT6, where the values of the coverage are exactly one and zero (which means total coverage of the hybrid over the original approach and never the opposite), and ZDT4 in which the coverage difference was reduced. When performing twice

[3]We took the implementation of NSGA-II available from its author at `http://www.iitk.ac.in/kangal/codes.shtml`

Table 6.1: Test problems adopted

Problem	Functions	Domain
ZDT 1	$f_1(x) = x_1$ $f_2(x) = g(x)(2 - \sqrt{f_1(x)/g(x)})$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i^2$	$[0,1] \times [-1,1]^n$
ZDT 2	$f_1(x) = x_1$ $f_2(x) = g(x)(2 - (f_1(x)/g(x))^2)$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i^2$	$[0,1] \times [-1,1]^n$
ZDT 3	$f_1(x) = x_1$ $f_2(x) = g(x)(2 - \sqrt{f_1(x)/g(x)} -$ $(f_1(x)/g(x))\sin(10\pi f_1))$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i^2$	$[0,1] \times [-1,1]^n$
ZDT 4	$f_1(x) = x_1$ $f_2(x) = g(x)(2 - \sqrt{f_1(x)/g(x)})$ $g(x) = 1 + 10(n-1) +$ $\sum_{i=2}^{n}(x_i^2 - 10\cos 4\pi f_1)$	$[0,1] \times [-5,5]^n$
ZDT 6	$f_1(x) = 1 - e^{-4x_1}$ $f_2(x) = g(x)(2 - (f_1(x)/g(x))^2)$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i^2$	$[0,1] \times [-1,1]^n$

the number of evaluations, the plain NSGA-II was able to converge closer to the true Pareto front, but was still never able to set-cover the hybrid approach.

In Table 6.4, we show the results obtained for the extended versions of the test problems. We can see here similar results to those obtained when dealing with the original versions of the test problems. However, in this case the outperformance of the hybrid over the plain NSGA-II improved as the number of iterations was increased. Tables 6.5 and 6.6 support these results, and show the IGD indicator values which measures both, proximity to the front and spread of solutions.

It is worth noticing that, since ZDT4 is a highly multi-frontal MOP, then, a gradient-based method is expected to get stuck when attempting to solve it, producing, as a consequence, a negative impact on the performance of the hybrid MOEA. However, our results do not indicate that this is always the case. However, it is important to notice that this problem, as well as ZDT2, were the least stable in terms of the standard deviation for the IGD indicator.

Another important aspect to mention is that, since the two compared algorithms use the same crowding selection procedure, the distance towards the Pareto front does not decrease monotonically at the last stages of the search, since some good solutions could be deleted at further iterations. This is also the reason why no useful information (for comparison purposes) can be obtained from letting the algorithms to perform a higher number of objective function

Table 6.2: Number of decision variables used for each test problem

Problem	Original Size	Extended Size
ZDT 1	30 variables	60 variables
ZDT 2	30 variables	60 variables
ZDT 3	30 variables	60 variables
ZDT 4	10 variables	15 variables
ZDT 6	10 variables	15 variables

Table 6.3: Comparison of results using the Set Coverage indicator for the original versions of the test problems.

Problem	NSGA-II	Set Coverage			
		5,000 f. evals.		10,000 f. evals.	
	version:	Mean	σ	Mean	σ
ZDT 1	Hybrid $<$ Plain	0.98	0.03	0.78	0.08
	Plain $<$ Hybrid	0.00	0.00	0.00	0.00
ZDT 2	Hybrid $<$ Plain	0.91	0.21	0.88	0.08
	Plain $<$ Hybrid	0.01	0.07	0.00	0.00
ZDT 3	Hybrid $<$ Plain	0.90	0.11	0.60	0.09
	Plain $<$ Hybrid	0.03	0.06	0.01	0.03
		20,000 f. evals.		40,000 f. evals	
		Mean	σ	Mean	σ
ZDT 4	Hybrid $<$ Plain	0.66	0.29	0.32	0.19
	Plain $<$ Hybrid	0.11	0.27	0.00	0.00
ZDT 6	Hybrid $<$ Plain	1.00	0.00	1.00	0.00
	Plain $<$ Hybrid	0.00	0.00	0.00	0.00

Table 6.4: Comparison of results using the Set Coverage indicator, for the extended versions of the test problems.

Problem	NSGA-II	Set Coverage			
		5,000 f. evals.		10,000 f. evals.	
	version:	Mean	σ	Mean	σ
ZDT 1	Hybrid < Plain	0.99	0.02	1.00	0.01
	Plain < Hybrid	0.00	0.00	0.00	0.00
ZDT 2	Hybrid < Plain	0.86	0.30	0.90	0.24
	Plain < Hybrid	0.05	0.17	0.00	0.00
ZDT 3	Hybrid < Plain	0.88	0.17	0.95	0.05
	Plain < Hybrid	0.10	0.18	0.01	0.03
		20,000 f. evals.		40,000 f. evals	
		Mean	σ	Mean	σ
ZDT 4	Hybrid < Plain	0.55	0.49	0.70	0.29
	Plain < Hybrid	0.52	0.48	0.13	0.32
ZDT 6	Hybrid < Plain	1.00	0.00	1.00	0.00
	Plain < Hybrid	0.00	0.00	0.00	0.00

evaluations.

Figures 6.2 to 6.6 show the Pareto fronts generated for each of the test problems, in their two instances (original and extended), and for the two numbers of iterations adopted. These Pareto fronts correspond to randomly selected runs.

Figures 6.7 to 6.11 show the average over 30 runs for the number of non-dominated points—which is the basis of our control mechanism—and the GD as the number of function evaluations increases. These comparisons are presented in the original and also in the extended size versions of the problems.

When hybridizing MOEAs with gradient-based procedures, an obvious question that arises is if this sort of hybrid scheme is more cost-effective than the use of a plain MOEA. This cannot be easily answered, and few studies that look into this are currently available. However, we conclude that the answer to this question depends on two things: the specific features of the problem to be solved, and the effectiveness of the mechanism that balances the LS with the global search. Regarding the scalability of Algorithm 18, we can comment that excluding the computation of the descent direction (Proposition 3.2.1), the ideas presented here can be used to hybridize MOEAs to deal with more than two objectives. For those cases, we suggest to use any of the the procedures proposed in Chapter 4, or the ones proposed by Fliege et al. [42] or by Schäffler et al. [107]. It is worth noting that, in these cases (except for HCS1), it is necessary to use a solver for convex quadratic optimization or, at least, a linear optimization solver.

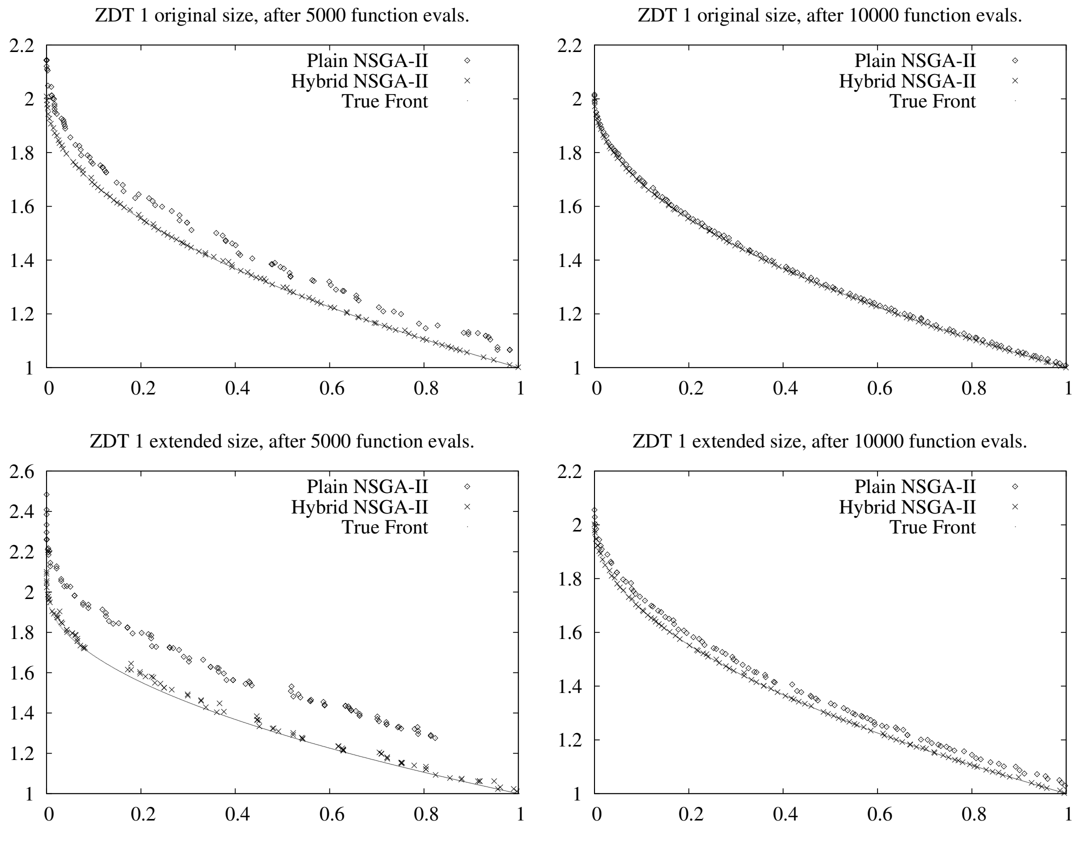


Figure 6.2: Pareto fronts corresponding to a random run (seed 0.39) for ZDT1. Original (top) and extended problem sizes (bottom), 5000 (left) and 10000 (right) evaluations.

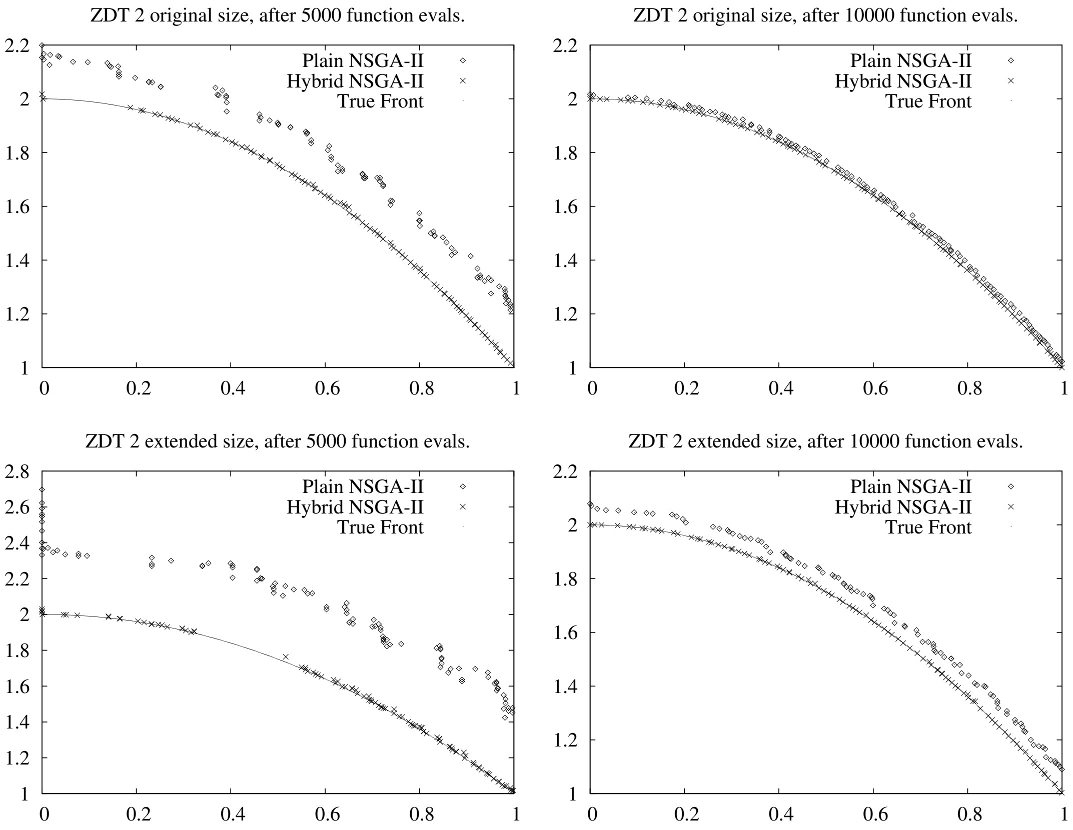


Figure 6.3: Pareto fronts corresponding to a random run (seed 0.39) for ZDT2. Original (top) and extended problem sizes (bottom), 5000 (left) and 10000 (right) evaluations.

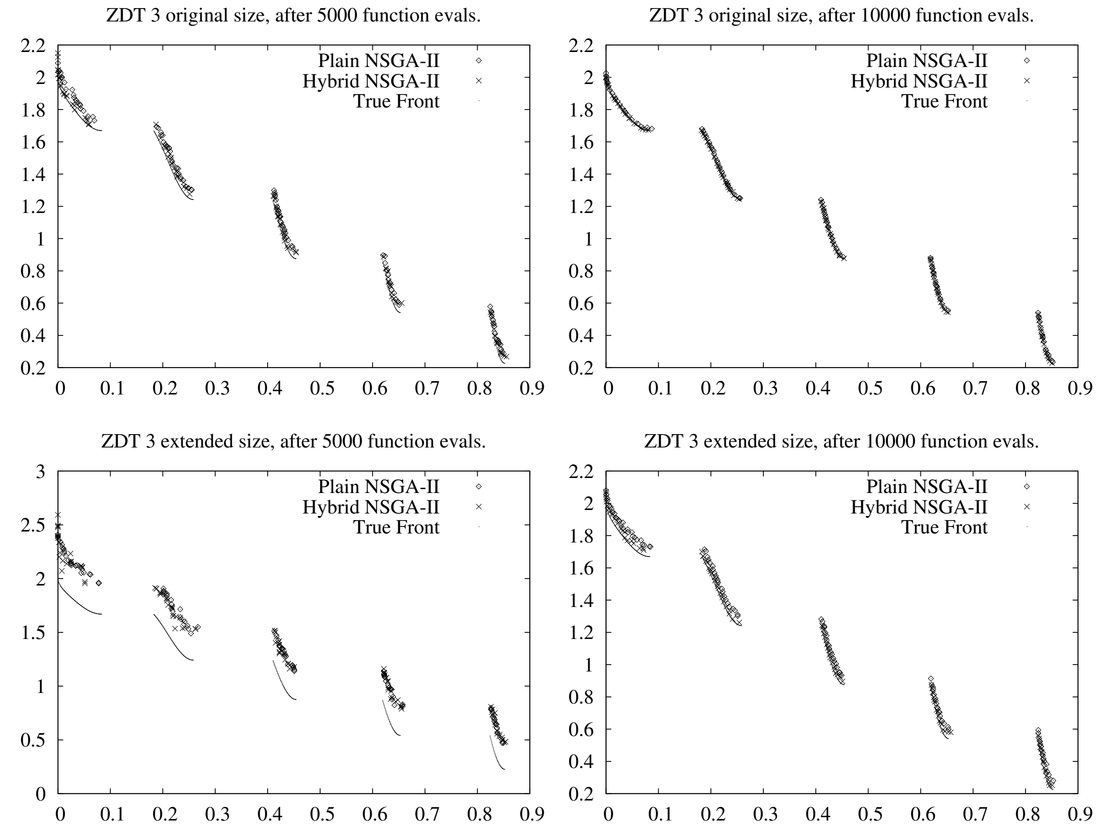


Figure 6.4: Pareto fronts corresponding to a random run (seed 0.39) for ZDT3. Original (top) and extended problem sizes (bottom), 5000 (left) and 10000 (right) evaluations.

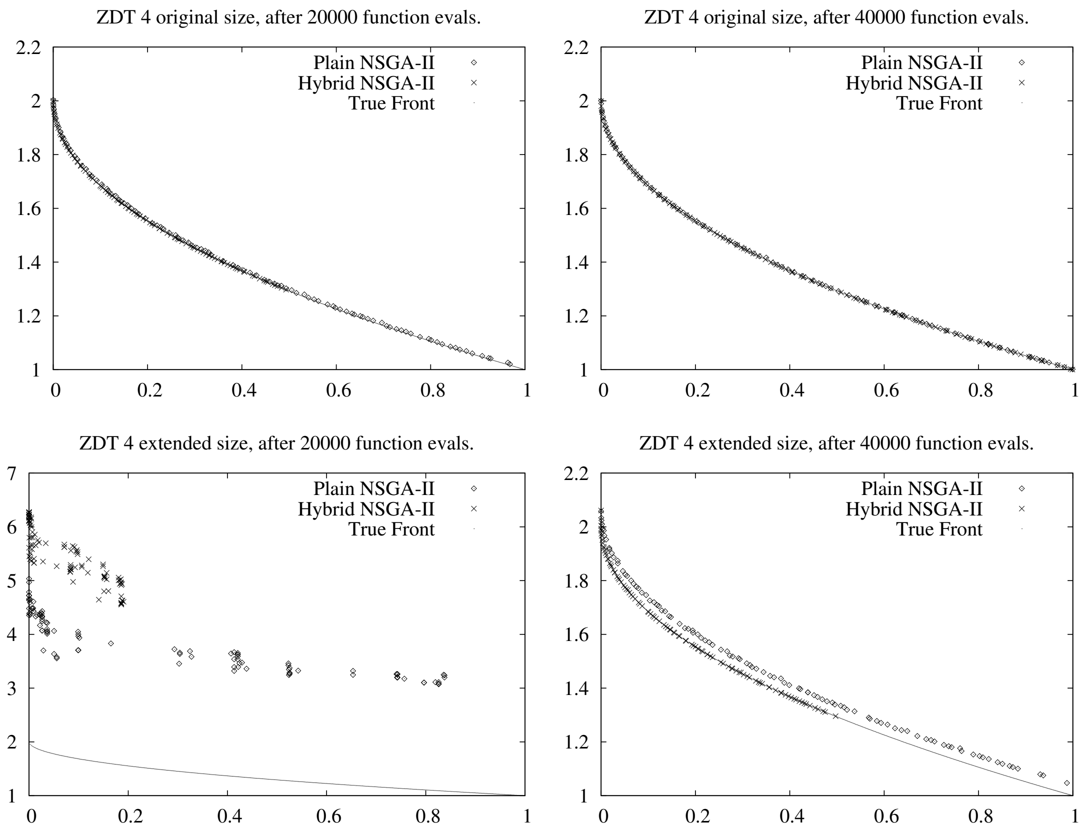


Figure 6.5: Pareto fronts corresponding to a random run (seed 0.39) for ZDT4. Original (top) and extended problem sizes (bottom), 20000 (left) and 40000 (right) evaluations.

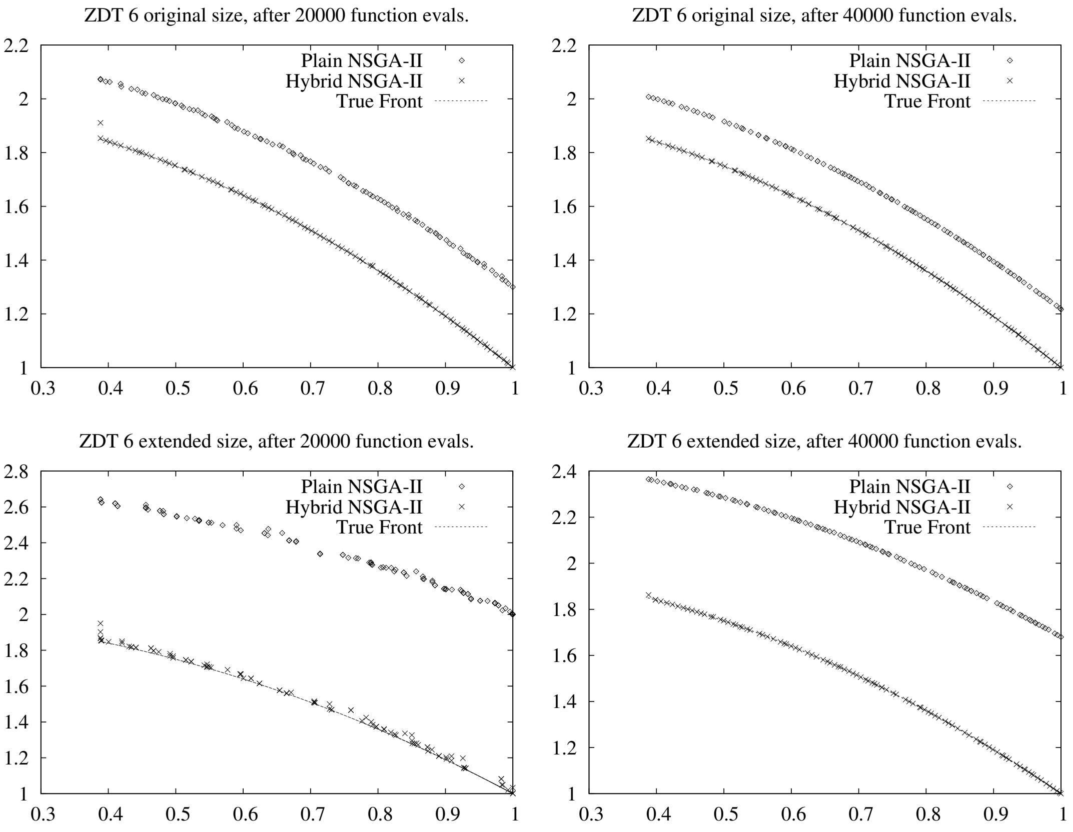


Figure 6.6: Pareto fronts corresponding to a random run (seed 0.39) for ZDT6. Original (top) and extended problem sizes (bottom), 20000 (left) and 40000 (right) evaluations.

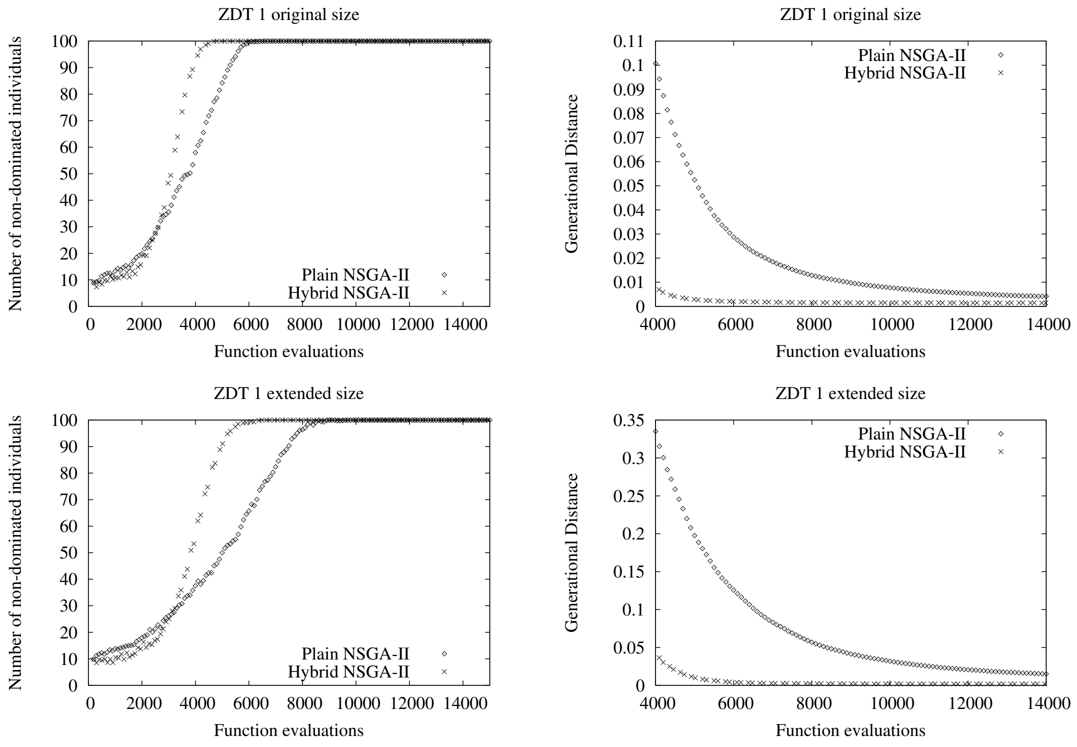


Figure 6.7: Average over 30 runs for ZDT1 with both its original size (top) and its extended size (bottom). We also show the number of non-dominated elements (left), and the Generational Distance (right).

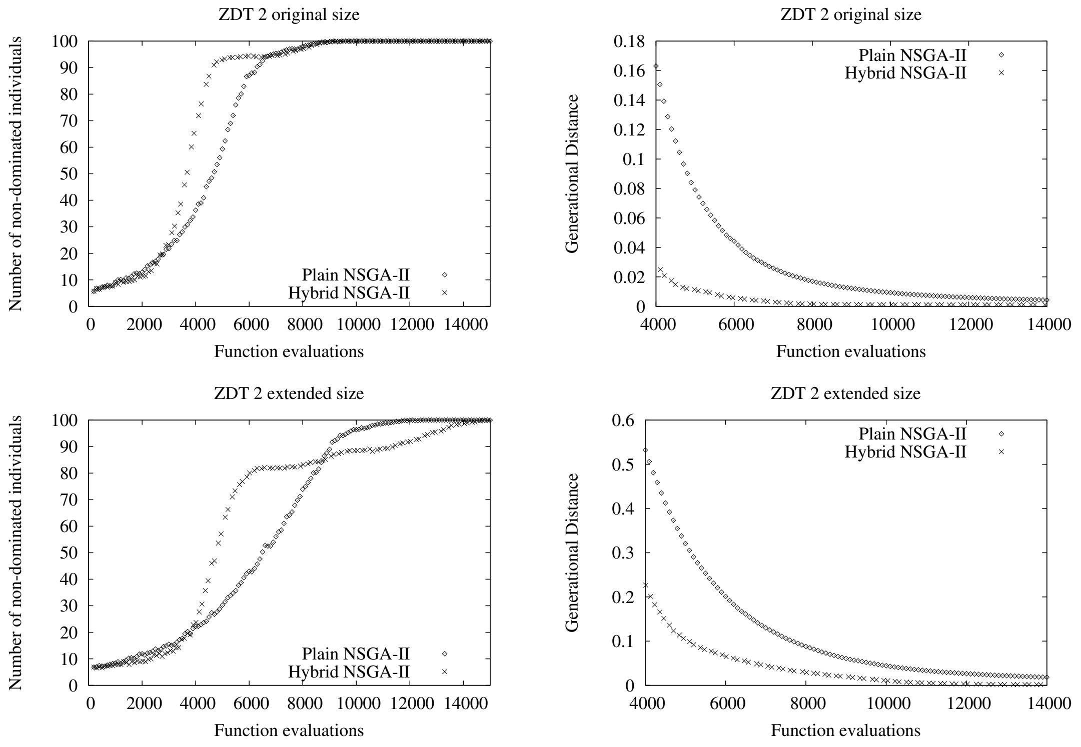


Figure 6.8: Average over 30 runs for ZDT2 with both its original size (top) and its extended size (bottom). We also show the number of non-dominated elements (left), and the Generational Distance (right).

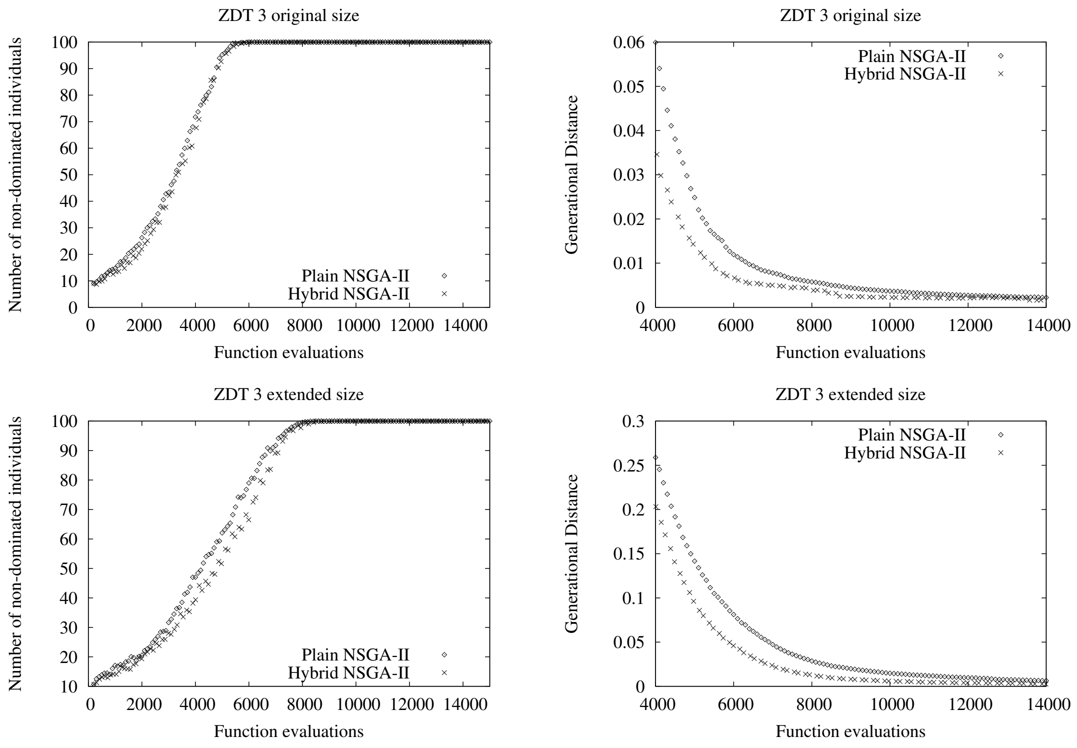


Figure 6.9: Average over 30 runs for ZDT3 with both its original size (top) and its extended size (bottom). We also show the number of non-dominated elements (left), and the Generational Distance (right).

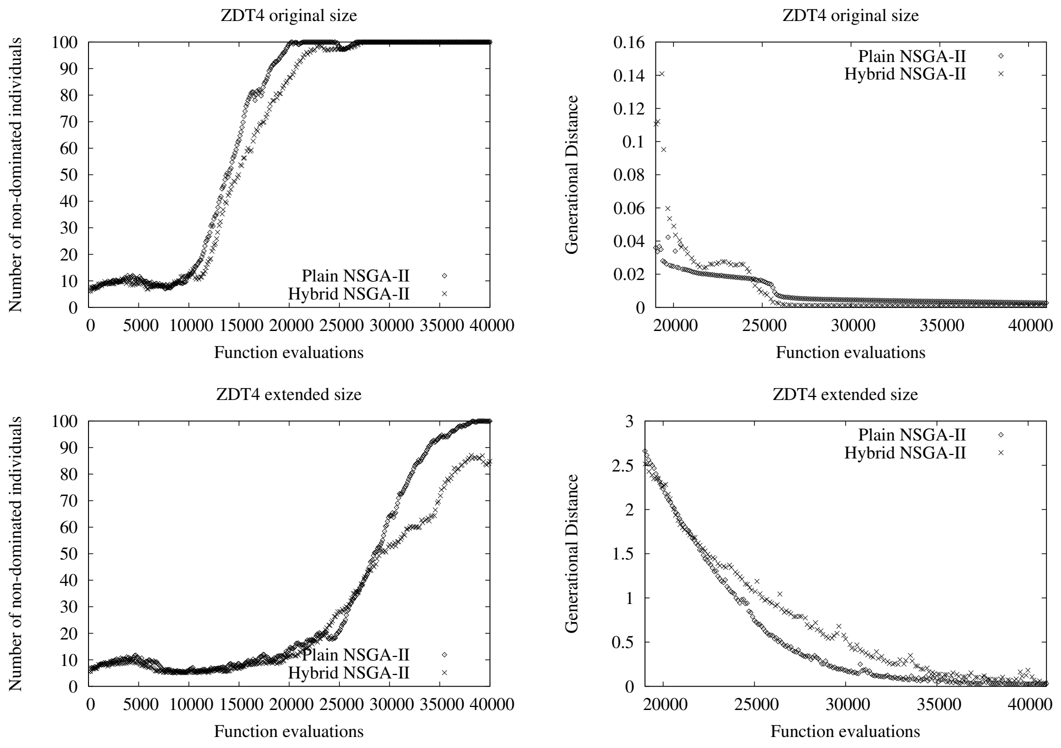


Figure 6.10: Average over 30 runs for ZDT4 with both its original size (top) and its extended size (bottom). We also show the number of non-dominated elements (left), and the Generational Distance (right).

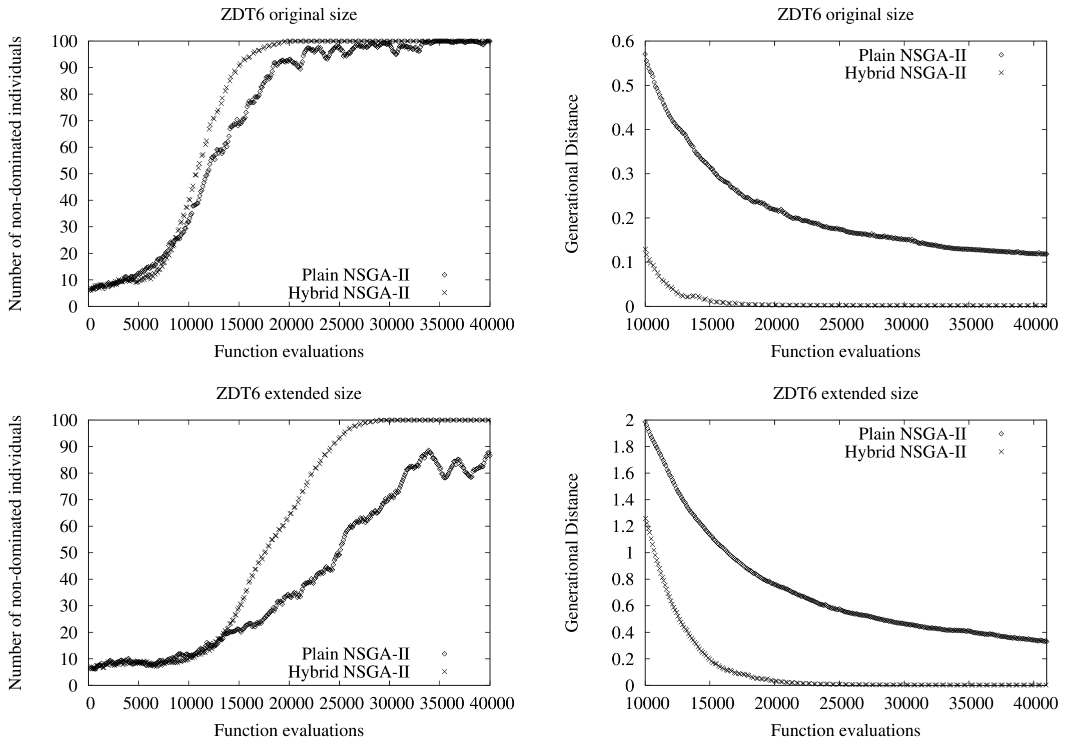


Figure 6.11: Average over 30 runs for ZDT6 with both its original size (top) and its extended size (bottom). We also show the number of non-dominated elements (left), and the Generational Distance (right).

Table 6.5: Comparison of results for the IGD indicator, adopting the test problems with their original sizes.

Problem	NSGA-II version:	IGD 5,000 function evals. Mean	Std. Dev.	10,000 function evals. Mean	Std. Dev.
ZDT 1	Hybrid	9.059E-03	9.080E-03	4.798E-03	5.318E-04
	Plain	5.134E-02	1.129E-02	9.349E-03	8.545E-04
ZDT 2	Hybrid	4.437E-02	1.109E-01	7.108E-03	1.264E-02
	Plain	1.114E-01	1.396E-01	1.446E-02	1.398E-02
ZDT 3	Hybrid	1.959E-02	1.252E-02	6.084E-03	4.021E-03
	Plain	3.722E-02	9.161E-03	7.443E-03	6.264E-04
		20,000 function evals. Mean	Std. Dev.	40,000 function evals Mean	Std. Dev.
ZDT 4	Hybrid	1.237E-01	1.233E-01	4.511E-03	1.441E-04
	Plain	6.191E-02	8.149E-02	5.483E-03	9.147E-04
ZDT 6	Hybrid	2.859E-03	3.342E-04	2.422E-03	1.828E-04
	Plain	3.262E-01	7.280E-02	1.806E-01	3.616E-02

Table 6.6: Comparison of results for the IGD indicator, adopting the test problems in their extended versions.

Problem	NSGA-II version:	IGD 5,000 function evals. Mean	Std. Dev.	10,000 function evals. Mean	Std. Dev.
ZDT 1	Hybrid	1.269E-02	5.371E-03	4.816E-03	1.682E-04
	Plain	1.899E-01	3.366E-02	3.273E-02	5.349E-03
ZDT 2	Hybrid	1.785E-01	3.038E-01	4.189E-02	1.016E-01
	Plain	3.167E-01	1.102E-01	5.197E-02	4.667E-02
ZDT 3	Hybrid	1.086E-01	2.525E-02	1.206E-02	7.814E-03
	Plain	1.487E-01	2.186E-02	2.632E-02	9.028E-03
		20,000 function evals. Mean	Std. Dev.	40,000 function evals Mean	Std. Dev.
ZDT 4	Hybrid	2.018E+00	9.879E-01	1.137E-01	9.409E-02
	Plain	2.086E+00	8.390E-01	3.600E-02	2.234E-02
ZDT 6	Hybrid	1.080E-02	1.044E-02	2.439E-03	1.434E-04
	Plain	1.023E+00	1.293E-01	4.985E-01	9.087E-02

6.4 Archive Bounding Process

Why should we spend resources refining a solution that will be later deleted by a crowding-like strategy? When using gradient information at the end of the search, solutions are as accurate as the amount of resources we want to spend in the line search. If we incorporate this refinement as part of the LS engine during the execution of the hybrid algorithm, the same archive truncation mechanism—present in most of the MOEAs—will withdraw these improvements.

Another option is to look at special archiving strategies, such as those described in [124], which could deal with this issue. Important open problems in this sense are the interplay of the archiving and the selection mechanism of the MOEA, how to set the mutation radius (step size) in the MOEA, its relation with the archive setting, and what resolution of the archive should be used. Besides, the detection of promising elements in the archive is very important during the evolutionary search.

From the experiments presented here, it can be noticed in Figures 6.7 to 6.11 (right hand-side) that the crowding procedure truncation does not allow the method to converge. In other words, the zero value is never reached when assessing GD for a large number of function evaluations. The worst part, of this situation, is that there is a positive probability to lose, during the selection (when using the crowding procedure), points which have already been improved (by the application of LS). The alternative that we adopted for these experiments was to skip the calculation of the crowding value for the elements resulting from the application of the LS (by assigning a special value to them). However, this mechanism only saves these elements for the next generation and it does not prevent that, at future generations, the improved individuals are deleted. Otherwise, we could interfere with the original diversity control process of the MOEA and this could have a negative effect.

As a final suggestion, if guaranteed convergence is searched for the procedure, the observation presented above motivates the study and use of different types of archivers for MOEAs.

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing

7 Conclusions and Extensions

In this work we have presented our research on the design of hybrid MOEAs that incorporate gradient information. We analyzed the main issues when making this coupling from both, the LS and the global search point of view. This is an important problem since most of the current multi-objective memetic algorithms have focused on discrete problems, in which the LS over neighborhoods has been well studied. However, for the continuous case, there is no direct comparison with such (discrete) proposals. In this sense, for the exploration of continuous neighborhoods we used gradient-based line search. This is a powerful tool that certainly implies a high cost, but that it is also a reliable way to produce improvements. The material presented in this thesis contributes to increase the knowledge about these mathematical techniques and to motivate their use to improve the efficiency of MOEAs. We have shown the huge potential of combining gradient information with multi-objective evolutionary search. Also, theoretical aspects and open research branches have been discussed.

The contributions of this thesis are stated in the following:

1) Specialized local search mechanisms: This is about the construction and computacion of gradient-based search directions; also about some different aspects regarding the design of LS operators based on them.

First, we incorporated the gradient of two functions (Expression 3.3, in page 26), in the most simple possible way compared with other works which use gradient-based multi-objective line search—since these other procedures require at least the solution of linear equation systems. In this sense, we developed a "plug and play" method that can be easily coupled with many MOEAs with little effort. The most important advantage, of implementing LS with our proposed approach, is that no additional quadratic or linear optimization solvers are required to calculate the descent direction for two-objective problems. This makes our approach cost-free, in a certain extent, for such types of problems.

Second, in Chapter 4 we have proposed a novel point-wise iterative search procedure, the Hill Climber with Sidestep (HCS), which is designed for the LS of a given MOP. The HCS is intended to be capable of moving both toward and along the set of (local) Pareto points, depending on the location of the current iterate. We proposed two variants of the HCS, a gradient-free version (HCS1) and one version which explicitly exploits gradient information (HCS2). Both can be used as standalone algorithms, to explore parts of the Pareto set starting with one single solution; and are able to handle constraints of the model, to some extent. We also presented, in Section 4.2.5 (page 47), some numerical results indicating the efficiency of the HCS as a standalone algorithm, and its benefit when being integrated into a MOEA in Section 5.4.4 (page 99).

Third, we presented the Directed Search method (DS) for multi-objective optimization, which allows to steer the search into any direction α, given in objective space. Based on this idea we also presented a novel continuation procedure, which depends of the initial point x_0 and of the choice of the desired direction α. We have emphasized the similarities and differences among our approach and other possibilities like the Weighted Sum approach and Goal Programming. Though these directions are locally optimal, the resulting approaches do not guarantee, however, that at least a locally optimal point is reached. The essential novelty in the alternative continuation method we have presented is the choice of the predictor direction: instead of linearizing the Pareto *set* we have used a linearization of the Pareto *front* to obtain a new predictor solution; By this, no second order information is required, as in 'classical' continuation methods. This aspect makes the new strategy a competitive alternative, in particular for the treatment of higher dimensional problems. To conclude, we have illustrated the behavior of both methods in some benchmark problems.

Finally, our three proposals for local searchers also include a tolerance controlled stopping criterion—in order to avoid applying LS in a particular (almost optimal) point.

2) Memetic algorithm construction: We have addressed in Chapter 5 the problem of integrating some of our local searchers into a given MOEA, in order to obtain novel memetic strategies. As examples, we have proposed, in Section 5.4 (page 89), the two algorithms (or family of algorithms) SPEA2-HCS and NSGA-II-HCS which are derived from SPEA2 and NSGA-II, respectively. More precisely, the results of SPEA2-HCS and NSGA-II-HCS show that the combination as proposed here is advantageous in many cases. However, it has to be mentioned that for this, the design parameters of the HCS and the balance between local and genetic search has to be chosen properly. In that chapter we also presented GBMES, a two stage algorithm which reaches such a balance while requiring a total of 3,000 objective function evaluations—a very competitive performance for the particular benchmark problems used. With this work we showed that although obtaining gradient information is an expensive process, it is possible to design an efficient gradient-based hybrid. Another studied feature of GBMES is that it scales well as we increase the number of decision variables of the MOP. This was illustrated by two examples in which we use up to 100 decision variables. This approach was found to degrade significantly less than a state-of-the-art MOEA (the NSGA-II), while still performing the small budget of 3,000 objective function evaluations.

We also tackled, in Chapters 5 and 6, the problem of the balance between LS and global search. This is not a trivial issue, and it has been indeed recognized as one of the main difficulties when designing memetic MOEAs. The work presented in Section 6.2 (page 117) is a first step towards developing a fully adaptive method, that can automatically balance the role of each of the two search engines (*i.e.*, the global search and the LS engines) when dealing with continuous problems. We presented a criterion to switch between the local and the global search procedures based on the cardinality of the non dominated set. The aim of such mechanism is precisely to automatically release resources (*i.e.*, function evaluations) for each of the two engines. Algorithm 18 presents a generic gradient-helped MOEA which uses this kind of balance method. The hybrid algorithm showed advantages over the plain MOEA in the test problems adopted in different instances (*i.e.*, with their original dimensionality and with a higher one). Since the application of LS is an expensive procedure for MOPs, it is advisable to perform an efficient interleaving with the MOEA.

Future Work

As part of our future work, we are interested in coupling alternative archiving methods (see for example those in [124]) with our approaches. Such type of mechanism should be able to preserve good solutions during a longer time, which would be beneficial for the performance of the final algorithm. Archiving strategies have different edges to investigate, since convergence of the procedures lies most of the time on them. On the other hand, the management of a good archiving method is not an easy task since it has several computational issues—even when dealing with a moderate number of dimensions.

Since gradient-based descent was proved to increase the accuracy of solutions, an interesting way to exploit it is by reconstruction techniques—like the one presented in Section 5.2.3 (page 77). In this regard, the rough sets mechanism can be improved—as a future work—by introducing gradient-based information into it; however, this should be done very carefully, because of the high computational cost associated with obtaining this information. For this sake, it is possible to take advantage of the construction explained in the preliminary phase of Section 5.2.3 (page 77). Such construction could be selectively repeated and mixed with a gradient-based descent applicable only to a few selected individuals.

When performing a gradient-based line search in multi-objective problems, two important factors determine the success of the procedure: First, we must choose a suitable search direction, and second, a good step length control must be set. In this thesis, we have performed studies for the former issue, which make promising the use of this technique. On the other hand, methods to control the step length in a multi-objective problem represent a hard open question—but a treatable one since these methods have been widely studied in the mathematical programming literature for the single-objective case.

In addition, we have the hypothesis that gradient-based movements could deal with *dominance resistant solutions* (DRS) [61]. DRS are points in the domain which are difficult to improve using the evolutionary operators. This is an unexplored and natural application of our proposed methods. Finally, even when the computation of search directions has been well studied, calculating them on the presence of constraint MOPs has scarcely been tackled [53, 42]; this is certainly a topic that deserves further research.

Bibliography

[1] S. F. Adra, I. Griffin, and P. J. Fleming. **Hybrid Multiobjective Genetic Algorithm with a New Adaptive Local Search Process**. In Hans-Georg Beyer et al., editor, *2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, **1**, pages 1009–1010, New York, USA, June 2005. ACM Press.

[2] E. L. Allgower and K. Georg. *Numerical Continuation Methods*. Springer, 1990.

[3] Eugene L. Allgower and Kurt Georg. *Numerical Continuation Methods*. Springer Verlag, 1990.

[4] L. Armijo. **Minimization of functions having Lipschitz continuous first partial derivatives**. *Pacific Journal of mathematics*, **16**(1):1–3, 1966.

[5] L. Armijo. **Minimization of functions having Lipschitz-continuous first partial derivatives**. *Pacific Journal of Mathematics*, **16**:1–3, 1966.

[6] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996.

[7] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. **Systematic Integration of Parameterized Local Search Into Evolutionary Algorithms**. *IEEE Transactions on Evolutionary Computation*, **8**(2):137–155, 2004.

[8] Peter A.N. Bosman and Edwin D. de Jong. **Exploiting Gradient Information in Numerical Multi-Objective Evolutionary Optimization**. In Hans-Georg Beyer et al., editor, *2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, **1**, pages 755–762, New York, USA, June 2005. ACM Press.

[9] Peter A.N. Bosman and Edwin D. de Jong. **Combining Gradient Techniques for Numerical Multi-Objective Evolutionary Optimization**. In Maarten Keijzer et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, **1**, pages 627–634, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[10] Peter A.N. Bosman and Dirk Thierens. **The Naive MIDEA: A Baseline Multi-objective EA**. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 428–442, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[11] R. Bramley and B. Winnicka. **Solving linear inequalities in a least squares sense**. *SIAM Journal on Scientific Computing*, **17**(1):275–286, 1996.

[12] J. Branke, K. Deb, H. Dierolf, and M. Osswald. **Finding knees in multi-objective optimization**. *Lecture notes in computer science*, **3242**:722–731, 2004.

[13] Jürgen Branke and Sanaz Mostaghim. **About selecting the personal best in multi-objective particle swarm optimization**. In *Parallel Problem Solving from Nature - PPSN IX*, pages 523–532, Springer Berlin / Heidelberg, 2006.

[14] Martin Brown and R.E. Smith. **Effective Use of Directional Information in Multi-objective Evolutionary Computation**. In *Genetic and Evolutionary Computation âĂŞ GECCO 2003*, **Volume 2723/2003** of *Lecture Notes in Computer Science*, pages 778–789. Springer Berlin / Heidelberg, 2003.

[15] Martin Brown and R.E. Smith. **Directed Multi-objective Optimization**. *International Journal of Computers, Systems and Signals*, **6**(1):3–17, 2005.

[16] A. Caponio and F. Neri. **Integrating Cross-Dominance Adaption in Multi-Objective Memetic Algorithms**. In C.-K. Goh, Y.-S. Ong, and K. C. Tan, editors, *Multi-Objective Memetic Algorithms*, pages 325–351. Springer, Studies in Computational Intelligence , Vol. 171, 2009.

[17] A. Charnes and W. W. Cooper. *Management models and industrial applications of linear programming*. Wiley, New York, 1961.

[18] C. A. Coello Coello and N. Cruz Cortés. **Solving Multiobjective Optimization Problems using an Artificial Immune System**. *Genetic Programming and Evolvable Machines*, **6**(2):163–190, June 2005.

[19] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[20] Carlos A. Coello Coello. **A Short Tutorial on Evolutionary Multiobjective Optimization**. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[21] Carlos A. Coello Coello and A. D. Christiansen. **Multiobjective optimization of trusses using genetic algorithms**. *Computers and Structures*, **75**(6):647–660, May 2000.

[22] Carlos A. Coello Coello and Gary B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004. ISBN 981-256-106-4.

[23] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.

[24] Carlos A. Coello Coello and Gregorio Toscano Pulido. **Multiobjective Optimization using a Micro-Genetic Algorithm**. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[25] George Corliss, Christèle Faure, Andreas Griewank, Lauren Hascoët, and Uwe Naumann, editors. *Automatic Differentiation of Algorithms: From Simulation to Optimization*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[26] I. Das and J. Dennis. ***Normal-boundary intersection*: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems.** *SIAM Journal of Optimization*, **8**:631–657, 1998.

[27] 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 and Multidisciplinary Optimization*, **14**(1):63–69, August 1997.

[28] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. **Scalable test problems for evolutionary multi-objective optimization**. *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145, 2005.

[29] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. **Scalable Test Problems for Evolutionary Multiobjective Optimization**. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.

[30] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[31] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. **A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II**. *IEEE Transactions on Evolutionary Computation*, **6**(2):182–197, April 2002.

[32] M. Dellnitz, O. Schütze, and T. Hestermeyer. **Covering Pareto Sets by Multilevel Subdivision Techniques**. *Journal of Optimization Theory and Applications*, **124**:113–155, 2005.

[33] Michael Dellnitz, Oliver Schütze, and T. Hestermeyer. **Covering Pareto Sets by Multilevel Subdivision Techniques**. *Journal of optimization theory and applications*, **124**(1):113–136, 2005.

[34] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[35] P. Deuflhard and F. Borneman. *Scientific Computing with Ordinary Differential Equations*. Texts in Applied Mathematics 42. Springer New York, 2002.

[36] F. Y. Edgeworth. *Mathematical Physics*. P. Keagan, London, England, 1881.

[37] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer, Berlin Heidelberg, 2008. ISBN 978-3-540-79157-7.

[38] M. Farina and P. Amato. **On the Optimal Solution Definition for Many-criteria Optimization Problems**. In *Proceedings of the NAFIPS-FLINT International Conference'2002*, pages 233–238, Piscataway, New Jersey, June 2002. IEEE Service Center.

[39] J. Figueira, S. Greco, and M. Ehrgott. *Multiple criteria decision analysis: state of the art surveys*, **78**. Springer Verlag, 2005.

[40] J. Fliege. **Gap-free computation of Pareto-points by quadratic scalarizations**. *Mathematical Methods of Operations Research*, **59**:69–89, 2004.

[41] J. Fliege and B. Fux Svaiter. **Steepest descent methods for multicriteria optimization**. *Mathematical Methods of Operations Research*, **51**(3):479–494, 2000.

[42] Jörg Fliege and Benar Fux Svaiter. **Steepest descent methods for multicriteria optimization**. *Mathematical Methods of Operations Research*, **51**(3):479–494, February 2000.

[43] E. A. Portilla Flores, E. Mezura Montes, J. Alvarez Gallegos, Carlos A. Coello Coello, and C. A. Cruz Villar. **Integration of structure and control using an evolutionary approach: an application to the optimal concurrent design of a CVT**. *International Journal for Numerical Methods in Engineering*, **71**(8):883–901, 2007.

[44] P. García, M.L. Fátima, C.F. Julián, C.C. Rafael, A. Carlos, and A.G. Hernández-Díaz. **Hibridación de métodos exactos y heurísticos para el problema multiobjetivo**. *Rect@*, **Actas15**(1), 2007.

[45] T. Goel and K. Deb. **Hybrid Methods for Multi-Objective Evolutionary Algorithms**. In Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim, and Xin Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, **1**, pages 188–192, Orchid Country Club, Singapore, November 2002. Nanyang Technical University.

[46] D.E. Goldberg. **Genetic algorithms in search, optimization, and machine learning**. 1989.

[47] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000.

[48] Y.Y. HAIMES, L.S. LASDON, AND D.A. WISMER. **On a bicriterion formulation of the problems of integrated system identification and system optimization**. *IEEE Transactions on Systems, Man, and Cybernetics*, **1**(3):296–297, 1971.

[49] N. HANSEN AND A. OSTERMEIER. **Completely Derandomized Self-Adaptation in Evolution Strategies**. *Evolutionary Computation*, **9**(2):159–195, Summer 2001.

[50] K. HARADA, J. SAKUMA, AND S. KOBAYASHI. **Uniform Sampling of Local Pareto-Optimal Solution Curves by Pareto Path Following and its Applications in Multi-objective GA**. In DIRK THIERENS, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, **1**, pages 813–820, London, UK, July 2007. ACM Press.

[51] KEN HARADA, KOKOLO IKEDA, AND SHIGENOBU KOBAYASHI. **Hybridizing of Genetic Algorithm and Local Search in Multiobjective Function Optimization: Recommendation of GA then LS**. In MAARTEN KEIJZER ET AL., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO2006)*, **1**, pages 667–674, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[52] KEN HARADA, J. SAKUMA, S. KOBAYASHI, AND I. ONO. **Uniform sampling of local pareto-optimal solution curves by pareto path following and its applications in multi-objective GA**. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, page 820. ACM, 2007.

[53] KEN HARADA, JUN SAKUMA, AND SHIGENOBU KOBAYASHI. **Local Search for Multiobjective Function Optimization: Pareto Descent Method**. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 659–666, New York, NY, USA, 2006. ACM Press.

[54] W. E. HART. *Adaptive Global Optimization with Local Search*. PhD thesis, Univeristy of California, San Diego. USA, 1994.

[55] M. E. HENDERSON. **Multiple Parameter Continuation: Computing Implicitly Defined k-manifolds**. *International Journal of Bifurcation and Chaos*, **12**:451–476, 2002.

[56] ALFREDO G. HERNÁNDEZ-DÍAZ, CARLOS A. COELLO COELLO, FÁTIMA PÉREZ, RAFAEL CABALLERO, JULIÁN MOLINA, AND LUIS V. SANTANA-QUINTERO. **Seeding the Initial Population of a Multi-Objective Evolutionary Algorithm using Gradient-Based Information**. In *Congress on Evolutionary Computation (CEC 2008)*, pages 1617–1624, Hong Kong, June 2008. IEEE Service Center.

[57] C. HILLERMEIER. *Nonlinear Multiobjective Optimization - A Generalized Homotopy Approach*. Birkhäuser, 2001.

[58] CLAUS HILLERMEIER. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*, **135** of *International Series of Numerical Mathematics*. Birkhäuser, 2001.

[59] Xiaolin Hu, Zhangcan Huang, and Zhongfan Wang. **Hybridization of the Multi-Objective Evolutionary Algorithms and the Gradient-based Algorithms**. In *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*, **2**, pages 870–877, Canberra, Australia, December 2003. IEEE Press.

[60] C. Igel, N. Hansen, and S. Roth. **Covariance Matrix Adaptation for Multi-objective Optimization**. *Evolutionary Computation*, **15**(1):1–28, Spring 2007.

[61] K. Ikeda, H. Kita, and S. Kobayashi. **Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal?** In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, **2**, 2001.

[62] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. **Evolutionary many-objective optimization: a short review**. In *IEEE Congress on Evolutionary Computation (2008)*, pages 2419–2426, 2008.

[63] H. Ishibuchi, T. Yoshida, and T. Murata. **Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling**. *IEEE Transactions on Evolutionary Computation*, **7**(2):204–223, 2003.

[64] Hisao Ishibuchi and Tadahiko Murata. **Multi-Objective Genetic Local Search Algorithm**. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[65] Hisao Ishibuchi and Tadahiko Murata. **Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling**. *IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews*, **28**(3):392–403, August 1998.

[66] Hisao Ishibuchi, Tadashi Yoshida, and Tadahiko Murata. **Balance between Genetic Search and Local Search in Hybrid Evolutionary Multi-Criterion Optimization Algorithms**. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 1301–1308, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[67] Hisao Ishibuchi, Tadashi Yoshida, and Tadahiko Murata. **Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling**. *IEEE Transactions on Evolutionary Computation*, **7**(2):204–223, April 2003.

[68] J. Jahn. **Multiobjective search algorithm with subdivision technique**. *Computational Optimization and Applications*, **35**(2):161–175, 2006.

[69] A. López Jaimes, C. A. Coello Coello, and J. E. Uriás Barrientos. **Online objective reduction to deal with many-objective problems**. In M. Ehrgott et al., editor, *Evolutionary Multi-Criterion Optimization. 5th International Conference (EMO 2009)*, pages 423–437. Springer. Lecture Notes in Computer Science Vol. 5467, 2009.

[70] A. Jaszkiewicz. **Do Multiple-Objective Metaheuristics Deliver on Their Promises? A Computational Experiment on the Set-Covering Problem**. *IEEE Transactions on Evolutionary Computation*, **7**(2):133–143, April 2003.

[71] Andrzej Jaszkiewicz. **Genetic local search for multiple objective combinatorial optimization**. *European Journal of Operational Research*, **137**(1):50–71, 2002.

[72] Jahn Johannes. *Mathematical vector optimization in partially ordered linear spaces*. Frankfurt am Main ; New York : Lang, 1986.

[73] W. E. Karush. *Minima of functions of several variables with inequalities as side conditions*. PhD thesis, University of Chicago, 1939.

[74] K. Klamroth, J. Tind, and M. Wiecek. **Unbiased Approximation in Multicriteria Optimization**. *Mathematical Methods of Operations Research*, **56**:413–437, 2002.

[75] J. Knowles and D. Corne. **M-PAES: A Memetic Algorithm for Multiobjective Optimization**. In *2000 Congress on Evolutionary Computation*, **1**, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[76] J. Knowles and D. Corne. **Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects**. In William E. Hart, N. Krasnogor, and J.E. Smith, editors, *Recent Advances in Memetic Algorithms*, pages 313–352. Springer. Studies in Fuzziness and Soft Computing, Vol. 166, 2005.

[77] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

[78] P. Koch, Oliver Kramer, Günter Rudolph, and Nicola Beume. **On the hybridization of SMS-EMOA and local search for continuous multiobjective optimization**. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 603–610. ACM, 2009.

[79] N. Krasnogor. *Studies on the Theory and Design Space of Memetic Algorithms*. PhD thesis, University of the West of England, Bristol, United Kingdom, 2002.

[80] H. Kuhn and A. Tucker. **Nonlinear programming**. In J. Neumann, editor, *Proceeding of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, 1951.

[81] H. W. KUHN AND A. W. TUCKER. **Nonlinear Programming**. In *Proceedings of the Second Berkeley Symposium on Mathematics Statistics and Probability*. University of California Press, 1951.

[82] ADRIANA LARA, SERGIO ALVARADO, SHAUL SALOMON, GIDEON AVIGAD, CARLOS A. COELLO COELLO, AND OLIVER SCHÜTZE. **The Gradient Free Directed Search Method as Local Search within Multi-Objective Evolutionary Algorithms**. In *EVOLVE 2012, A bridge between Probability, Set Oriented Numerics and Evolutionary Computation (to appear)*, CINVESTAV, Mexico City, Mexico, August 2012.

[83] ADRIANA LARA, CARLOS A. COELLO COELLO, AND OLIVER SCHÜTZE. **Using Gradient-Based Information to Deal with Scalability in Multi-objective Evolutionary Algorithms**. In *IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 16–23, Trondheim, Norway, May 2009. IEEE Press.

[84] ADRIANA LARA, CARLOS A. COELLO COELLO, AND OLIVER SCHÜTZE. **A painless gradient-assisted multi-objective memetic mechanism for solving continuous bi-objective optimization problems**. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE Press, 2010.

[85] ADRIANA LARA, CARLOS A. COELLO COELLO, AND OLIVER SCHÜTZE. **Using gradient information for multi-objective problems in the evolutionary context**. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO 2010)*, pages 2011–2014. ACM, 2010.

[86] ADRIANA LARA, GUSTAVO SANCHEZ, CARLOS A. COELLO COELLO, AND OLIVER SCHÜTZE. **HCS: A New Local Search Strategy for Memetic Multi-Objective Evolutionary Algorithms**. *IEEE Transactions on Evolutionary Computation*, **14**(1):112–132, February 2010.

[87] ADRIANA LARA, OLIVER SCHÜTZE, AND CARLOS A. COELLO COELLO. *EVOLVE: A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, chapter On Gradient-based Local Search to Hybridize Multi-objective Evolutionary Algorithms. Studies in Computational Intelligence. Springer (in press).

[88] ADRIANA LARA, OLIVER SCHÜTZE, AND CARLOS A. COELLO COELLO. **New Challenges for Memetic Algorithms on Continuous Multi-objective Problems**. In *GECCO 2010 Workshop on Theoretical Aspects of Evolutionary Multiobjective Optimization*, pages 1967–1970, Portland, Oregon USA, July 2010. ACM.

[89] M. LOZANO, F. HERRERA, N. KRASNOGOR, AND D. MOLINA. **Real-coded memetic algorithms with crossover hill-climbing**. *Evolutionary Computation*, **12**(3):273–302, 2004.

[90] K. MIETTINEN. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.

[91] Kaisa M Miettinen. *Nonlinear Multiobjective Optimization*. Springer, 1999.

[92] Kaisa M Miettinen. **Some Methods for Nonlinear Multi-objective Optimization**. In E. et al Zitzler, editor, *Evolutionary Multi-Criterion Optimization*, **1993/2001** of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin, Heidelberg, 2001.

[93] P. Moscato and M.G. Norman. **A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems**. *Parallel Computing and Transputer Applications*, pages 177–186, 1992.

[94] H. Mukai. **Algorithms for multicriterion optimization**. *Automatic Control, IEEE Transactions on*, **25**(2):177–186, 1980.

[95] T. Murata, S. Kaige, and H. Ishibuchi. **Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms**. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 1234–1245. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[96] J. Nocedal and S. Wright. *Numerical Optimization, Series in Operations Research and Financial Engineering*. Springer, New York, 2006.

[97] Vilfredo Pareto. *Cours Déconomie Politique*. Lausanne, F. Rouge; Paris, Pichon, 1896.

[98] Zdzislaw Pawlak. **Rough sets**. *International Journal of Computer and Information Sciences*, **11**(1):341–356, Summer 1982.

[99] Zdzislaw Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.

[100] S. Poles, E. Rigoni, and T. Robič. **MOGA-II Performance on Noisy Optimization Problems**. In Bogdan Filipič and Jurij Šilc, editors, *Bioinspired Optimization Methods and Their Applications. Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2004*, pages 51–62. Jožef Stefan Institute, Ljubljana, Slovenia, October 2004.

[101] S.S. Rao and SS Rao. *Engineering optimization: theory and practice*. Wiley, 2009.

[102] E. Rigoni and S. Poles. **NBI and MOGA-II, two complementary algorithms for Multi-Objective optimization**. In *Dagstuhl Seminar Proceedings 04461. Practical Approaches to Multi-Objective Optimization*, pages 1–22, 2005.

[103] R. Tyrrell Rockafellar and J.B Wets Roger. *Variational analysis*, **317**. Springer Verlag, 1998.

[104] H.H. Rosenbrock. **An automatic method for finding the greatest or least value of a function**. *The Computer Journal*, **3**(3):175–184, 1960.

[105] Luis Vicente Santana-Quintero. *Development of techniques to improve computational efficiency in multi-objective evolutionary algorithms*. PhD thesis, Department of Computer Science, CINVESTAV-IPN, November 2008.

[106] L.V. Santana-Quintero, A.G. Hernández-Díaz, J. Molina, C.A. Coello Coello, and R. Caballero. **DEMORS: A hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems**. *Computers & Operations Research*, **37**(3):470–480, 2010.

[107] S. Schäffler, R. Schultz, and K. Weinzierl. **Stochastic method for the solution of unconstrained vector optimization problems**. *Journal of Optimization Theory and Applications*, **114**(1):209–222, 2002.

[108] J.R. Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Master's thesis, Department of Aeronautic and Astronautics, Massachusets Institute of Technology, 1995.

[109] Oliver Schuetze, L. Jourdan, T. Legrand, E.-G. Talbi, and J.-L. Wojkiewicz. **New analysis of the optimization of electromagnetic shielding properties using conducting polymers and a multi-objective approach**. *Polymers for Advanced Technologies*, **19**(7):762–769, July 2008.

[110] O. Schütze. *Set Oriented Methods for Global Optimization*. PhD thesis, University of Paderborn, 2004. <http://ubdata.uni-paderborn.de/ediss/17/2004/schuetze/>.

[111] O. Schütze, C. A. Coello Coello, S. Mostaghim, E.-G. Talbi, and M. Dellnitz. **Hybridizing Evolutionary Strategies with Continuation Methods for Solving Multi-Objective Problems**. *Engineering Optimization*, **40**(5):383–402, May 2008.

[112] O. Schütze, A. Dell'Aere, and M. Dellnitz. **On Continuation Methods for the Numerical Treatment of Multi-Objective Optimization Problems**. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Ralph E. Steuer, editors, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/349>.

[113] O. Schütze, M. Laumanns, E. Tantar, C. A. Coello Coello, and E.-G. Talbi. **Convergence of stochastic search algorithms to gap-free Pareto front approximations**. In *Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 892–901, 2007.

[114] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. **Covering Pareto Sets by Multilevel Evolutionary Subdivision Techniques**. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 118–132, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[115] Oliver Schütze, Carlos A. Coello Coello, Sanaz Mostaghim, El-Ghazali Talbi, and Michael Dellnitz. **Hybridizing evolutionary strategies with continuation methods for solving multi-objective problems**. *Engineering Optimization*, **40**(5):383–402, 2008.

[116] Oliver Schütze, Xavier Equivel, Adriana Lara, and C.A. Coello Coello. **Some Comments on GD and IGD and Relations to the Hausdorff Distance**. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1971–1974. ACM, 2010.

[117] Oliver Schütze, Xavier Esquivel, Adriana Lara, and Carlos A. Coello Coello. **Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multi-Objective Optimization**. *IEEE Transactions on Evolutionary Computation. (in press)*, (99):DOI: 10.1109/TEVC.2011.2161872, 2012.

[118] Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **The Directed Search Method for Multi-Objective Optimization Problems**. Technical report, http://delta. cs. cinvestav. mx/~schuetze/technical_reports/index. html, 2009.

[119] Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **Evolutionary Continuation Methods for Optimization Problems**. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 651–658, Montreal, Canada, July 8–12 2009. ACM Press. ISBN 978-1-60558-325-9.

[120] Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **The Directed Search Method for Unconstrained Multi-objective Optimization Problems**. In *EVOLVE 2011, A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, Bourglinster Castle, Luxembourg, May 2011.

[121] Oliver Schütze, Adriana Lara, and Carlos A. Coello Coello. **On the influence of the number of objectives on the hardness of a multiobjective optimization problem**. *IEEE Transactions on Evolutionary Computation*, **15**(4):444–455, August 2011.

[122] Oliver Schütze, Adriana Lara, Carlos A. Coello coello, and Massimiliano Vasile. **Computing approximate solutions of scalar optimization problems and applications in space mission design**. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE.

[123] Oliver Schütze, Adriana Lara, Carlos A. Coello Coello, and Massimiliano Vasile. **On the detection of nearly optimal solutions in the context of single-objective space mission design problems**. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **225**(11):1229–1242, 2011.

[124] Oliver Schütze, Marco Laumanns, Emilia Tantar, Carlos A. Coello Coello, and El-Ghazali Talbi. **Computing Gap Free Pareto Front Approximations with Stochastic Search Algorithms**. *Evolutionary Computation*, **18**(1):65–96, Spring 2010.

[125] Oliver Schütze, Michaell Laumanns, Emilia Tantar, Carlos A. Coello Coello, and El-Ghazali Talbi. **Convergence of Stochastic Search Algorithms to Gap-Free Pareto Front Approximations**. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007), 2007.

[126] Oliver Schütze, Gustavo Sanchez, and Carlos A. Coello Coello. **A New Memetic Strategy for the Numerical Treatment of Multi-Objective Optimization Problems**. In *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)*, pages 705–712. ACM Press, Atlanta, USA, July 2008. ISBN 978-1-60558-131-6.

[127] Oliver Schütze, El-Ghazali Talbi, Carlos Coello Coello, Luis Vicente Santana-Quintero, and Gregorio Toscano Pulido. **A Memetic PSO Algorithm for Scalar Optimization Problems**. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 128–134, Honolulu, Hawaii, USA, April 2007. IEEE Press.

[128] Oliver Schütze, Massimilano Vasile, Oliver Junge, Michael Dellnitz, and D. Izzo. **Designing optimal low thrust gravity assist trajectories using space pruning and a multi-objective approach**. *Engineering Optimization*, **41**(2):155–181, 2009.

[129] Pradyumn Shukla. **Gradient Based Stochastic Mutation Operators in Evolutionary Multi-objective Optimization**. In *Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms*, pages 58–66. Springer. Lecture Notes in Computer Science Vol. 4431, Warsaw, Poland, 2007.

[130] Pradyumn Shukla. **On Gradient Based Local Search Methods in Unconstrained Evolutionary Multi-objective Optimization**. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 96–110, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[131] K. Sindhya, K. Deb, and K. Miettinen. **A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence**. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 815–824. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany, September 2008.

[132] Karthik Sindhya, Kalyanmoy Deb, and Kaisa Miettinen. **A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence**. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 815–824. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germ., September 2008.

[133] J. E. Smith and T. C. Fogarty. **Operator and parameter adaption in genetic algorithms**. *Soft Computing*, **1**(2):81–87, 1997.

[134] O. Soliman, L. T. Bui, and H. Abbass. **A Memetic Coevolutionary Multi-Objective Diffierential Evolution Algorithm**. In C.-K. Goh, Y.-S. Ong, and K. C. Tan, editors, *Multi-Objective Memetic Algorithms*, pages 325–351. Springer, Studies in Computational Intelligence , Vol. 171, 2009.

[135] N. Srinivas and K. Deb. **Muiltiobjective optimization using nondominated sorting in genetic algorithms**. *Evolutionary computation*, **2**(3):221–248, 1994.

[136] B. Suman. **Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem**. *Computers & Chemical Engineering*, **28**:1849–1871, 2004.

[137] G. Timmel. **Ein stochastisches Suchverfahren zur Bestimmung der optimalen KompromisslÃűsung bei statistischen polykriteriellen Optimierungsaufgaben**. Technical report, TH Illmenau, 1980.

[138] G.N. Vanderplaats. *Numerical optimization techniques for engineering design: with applications*. McGraw-Hill New York, 1984.

[139] M. Vasile. **A Behavior-based Meta-Heuristic for Robust Global Trajectory Optimization**. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, pages 494–497, Singapore, 2007. IEEE Press.

[140] M. Vasile. **Hybrid behavioral-based multiobjective space trajectory optimization**. In C.-K. Goh, Y.-S. Ong, and K. C. Tan, editors, *Multi-Objective Memetic Algorithms*, pages 231–254. Springer, Studies in Computational Intelligence , Vol. 171, 2009.

[141] M. Vasile and M. Locatelli. **A hybrid multiagent approach for global trajectory optimization**. *Journal of Global Optimization*, **44**(4):461–479, 2009.

[142] M. Vasile and C. Maddock. **Design of Optimal Spacecraft-Asteroid Formations Through a Hybrid Global Optimization Approach**. *International Journal of Intelligent Computing and Cybernetics*, **1**(2):239–268, 2008.

[143] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[144] Y. Wang, Z. Cai, G. Guo, and Y. Zhou. **Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems**. *IEEE Transactions on Systems, Man and Cybernetics Part B–Cybernetics*, **37**(3):560–575, June 2007.

[145] E. F. Wanner, F. G. Guimaraes, R. H.C. Takahashi, and P. J. Fleming. **A Quadratic Approximation-Based Local Search Procedure for Multiobjective Genetic Algorithms**. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 3361–3368, Vancouver, BC, Canada, July 2006. IEEE.

[146] K. Witting and M. Hessel von Molo. Private communication, 2006.

[147] Y. Yin, T. Okabe, and B. Sendhoff. **Adapting Weighted Aggregation for Multi-Objective Evolution Strategies**. In *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, pages 96–110. Springer. Lecture Notes in Computer Science. Volume 1993, 2001.

[148] E. Zitzler and L. Thiele. **Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach**. *Evolutionary Computation, IEEE Transactions on*, **3**(4):257–271, 1999.

[149] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. **Comparison of Multiobjective Evolutionary Algorithms: Empirical Results**. *Evolutionary Computation*, **8**(2):173–195, Summer 2000.

[150] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. **SPEA2: Improving the Strength Pareto Evolutionary Algorithm**. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001.

[151] Eckart Zitzler and Lothar Thiele. **Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach**. *IEEE Transactions on Evolutionary Computation*, **3**(4):257–271, November 1999.