List of Tables

Chapter 1

Hybridizing Multi-objective Evolutionary Algorithms with Mathematical Programming Techniques

Saúl Zapotecas
CINVESTAV-IPN

Adriana Lara
Escuela Superior de Física y Matemáticas-IPN, Mexico City, Mexico

Carlos A. Coello Coello
CINVESTAV-IPN

1.1 Introduction

This chapter presents methods, and the general background, to the coupling of Multi-Objective Evolutionary Algorithms (MOEAs) with mathematical programming methods. The goal of this mixture of techniques is to improve the efficiency of MOEAs when solving continuous multi-objective optimization problems. Introductory descriptions for both techniques (MOEAs and classical approaches) are also included. The chapter makes a distinction between gradient-based and non-gradient-based methods. Different hybridization options are presented, in order to provide the reader with a richer companion of algorithms as well as discussions about them.

1.1.1 Multi-objective Background Concepts

The traditional optimization literature [52, 53], focuses on solving a Single-objective Optimization Problem (SOP), in which there is only one objective function to deal with; *i.e.,* they focus on optimizing

$$f : \mathbb{R}^n \to \mathbb{R}$$

subject to certain domain constraints. In this case, the solution is a single point—or set of points with the same image value—such that its image is a unique maximum or minimum value in $\mathbb{R}$. In contrast, when dealing with a Multi-objective Optimization Problem (MOP), we are interested in finding the best values for a function

$$\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m.$$

In this case, there is not a single point solution. This can be noticed, for example, in Figure 1.1, in which the functions are in conflict with each other—that is, while one increases, the other decreases. In the following, we are going to deepen into this idea, after explaining some basic concepts.

If an objective function is to be maximized (for example, if it corresponds to a certain profit) it is possible to restate it in terms of minimization, using the *duality principle* when multiplying the function by -1. Therefore, a MOP can be stated in general as:

$$\text{Minimizing } \mathbf{F}(\mathbf{x}) := [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \tag{1.1}$$

subject to:

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \ldots, k \tag{1.2}$$

$$h_j(\mathbf{x}) = 0 \quad j = 1, 2, \ldots, p \tag{1.3}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$ is the vector of *decision variables* (also known as *decision parameters* vector or *solution* vector), $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ are the *objective functions* (or *objectives,* in short) and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., k$, $j = 1, ..., p$ 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. For most of the following sections we will assume unconstrained MOPs.

Even though MOPs can be defined over other domains—such as discrete sets, for example—in the methods presented on this chapter we are only interested in continuous domains, which are contained in $\mathbb{R}^n$. When all the objective functions and the constraint functions are linear, the problem (1.1) 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 chapter we are dealing with nonlinear problems, either convex or not. For the remainder of this chapter, if conditions such as differentiability or continuous differentiability are assumed for the f_i functions, this will be pointed out for each particular method.

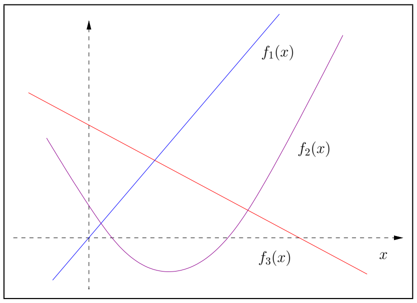

FIGURE 1.1: This figure shows function $f_3(\mathbf{x})$ in conflict with functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$.

In general, a solution of a MOP consists of an entire set of points which fulfills certain properties. We state the next definitions in order to establish a way to decide when a point can be considered part of this particular solution set.

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

Definition 2 *We say that a vector of decision variables* $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ *is* **non-dominated** *with respect to* $\mathcal{X}$ *(where* $\mathcal{X}$ *is the feasible region), if there does not exist another* $\mathbf{x}' \in \mathcal{X}$ *such that* $\mathbf{x}' \prec \mathbf{x}$*.*

Definition 3 *We say that a vector of decision variables* $\mathbf{x}^* \in \mathcal{X} \subset \mathbf{R}^n$ *is* **Pareto optimal** *if it is non-dominated with respect to* $\mathcal{X}$*.*

Definition 4 *The Pareto Optimal Set or* **Pareto Set** $\mathcal{P}^*$ *is defined by:*

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

Definition 5 *The* **Pareto Front** $\mathcal{PF}^*$ *is defined by:*

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

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 $\mathbf{x}^*$ as a Pareto point of (1.1), there exist [37] a vector $\boldsymbol{\alpha} \in \mathbb{R}^k$, with $0 \leq \alpha_i, i = 1, \ldots, k$ and $\sum_{i=1}^k = 1$ such that

$$\sum_i^k \alpha_i \nabla f_i(\mathbf{x}^*) = 0. \tag{1.4}$$

A point $\mathbf{x}^*$ that satisfies (1.4) is called a Karush-Kuhn-Tucker (KKT) point.

Example 1 *Consider the following unconstrained MOP: Minimize*

$$\begin{aligned} f_1 &= x^2 + y^2 \\ f_2 &= (x+4)^2 + y^2, \end{aligned} \tag{1.5}$$

with $x, y \in \mathbb{R}$*. The Pareto set of this problem is the line segment* $[(0,0),(-4,0)] \in \mathbb{R}^2$*. Figure 1.2 shows the image of uniformly generated points in the domain of the problem. The axes correspond to the values regarding each function, what is known like the objective space.*

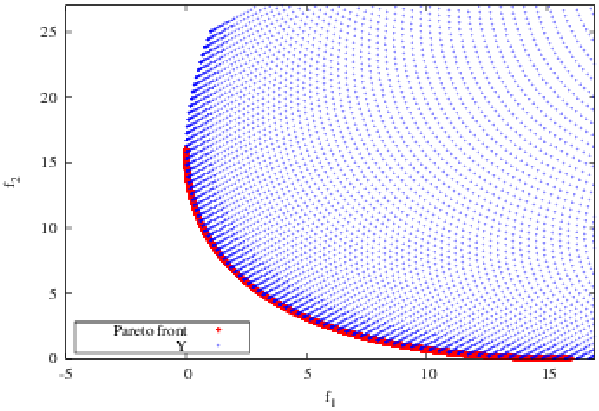

FIGURE 1.2: This figure emphasizes the Pareto front corresponding to Example 1. The points of Y are the images of uniformly generated vectors from the domain.

1.2 Solving a MOP

The most common procedures to solve (1.1) can be classified [45, 46] in four classes: *no-preference methods, a posteriori methods, a priori methods and interactive methods,* according to the stage, and level, at which the decision maker intervenes. In this chapter, we focus on the *a posteriori* approach, in which the goal is to obtain the best approximation of the entire set of optima. This solution will then be presented *a posteriori* to the decision maker—who will then select the most suitable solution out of it. We present in the following, a brief review of some classical techniques in order to mention important aspects that will then be compared with respect to MOEAs.

1.2.1 Mathematical Programming Techniques

Traditional approaches developed for solving MOPs are part of the mathematical programming literature, and are known as *classical methods.* Up to the early 1980s 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 [48]. 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:

$$\text{Minimize} \quad g(\mathbf{x}|\mathbf{w}) = \sum_{i=0}^{m} w_i f_i(\mathbf{x}) \tag{1.6}$$

where $\mathbf{x} \in \mathcal{X}$ and $\mathbf{w}$ is a weighting vector, i.e. $w_i \geq 0$ for all $i \in \{1, \ldots, m\}$ and $\sum_{i=1}^{m} w_i = 1$.

After this reformulation, the solution set found consists only of one point, corresponding to each weight combination. The main drawback of this approach is that the weights distribution does not necessary corresponds to the distribution of the points in the parameter space. Besides, there are points that cannot be generated as a combination of weights in non-convex cases—see [8].

Tchebycheff approach: This approach also transforms the vector of function values into a scalar optimization problem which is of the form:

$$\text{Minimize} \quad g(\mathbf{x}|\mathbf{w}, \mathbf{z}^\star) = \min_{1 \leq i \leq k} \{w_i |f_i(\mathbf{x}) - z_i|\} \tag{1.7}$$

where $\mathbf{x} \in \mathcal{X}$, $\mathbf{z}^\star = [z_1, \ldots, z_k]^T$, such that: $z_i = \min\{f_i(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$ and $\mathbf{w}$ is a weighting vector.

For each Pareto optimal point $\mathbf{x}^\star$ there exists a weighting vector $\mathbf{w}$ such that $\mathbf{x}^\star$ is the optimal solution of (1.7) and each optimal solution of (1.7) is a Pareto optimal solution of (1.1). Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector $\mathbf{w}$. One weakness of this approach is that its aggregation function is not smooth for a continuous MOP. There exist, in general, many scalarization methods which transform the MOP into a 'classical' SOP. It is worth noticing that certain selection of suitable SOPs can lead to a reasonable good approximation of the entire Pareto set—see for example [9].

ϵ-constraint method: In the *ϵ-constraint method* [18], one of the objectives is chosen for minimization while the rest of the objectives conform a set of constraints limited by user-specified bounds ϵ_i. *i.e.:*

$$\begin{aligned} \text{Minimize} \quad & f_j \\ \text{subject to} \quad & f_i \leq \epsilon_i \ \text{ for all } i \in \{1, \ldots, m\}, i \neq j. \end{aligned}$$

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 [45].

When a method explores iteratively solutions from a neighborhood, it is classified as a Local Search (LS) procedure. When solving a MOP according to the *a posteriori* approach, the use of population-based heuristics, such as the Evolutionary Algorithms (EAs) presented in the following, 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.

1.3 Multi-objective Evolutionary Algorithms

The limitations of traditional mathematical programming methods to solve MOPs, have motivated the development of new strategies to deal with these type of problems. The EAs are stochastic search and optimization methods that simulate the natural evolution process. At the end of the 1960s, Rosenberg [55] proposed the use of a Genetic Algorithms (GAs) to solve MOPs. However, it was until 1984, when David Schaffer [57] introduced the first implementation of what it is now called a MOEA. Since then, many researchers [33, 77, 11, 74] have developed a wide variety of MOEAs. MOEAs are particularly well-suited to solve MOPs because they operate over a set of potential solutions (they are based on a population). This feature allows them to generate several elements of the Pareto optimal set (or a good approximation of them) in a single run. Furthermore, MOEAs are less susceptible to the shape or continuity of the Pareto front than traditional mathematical programming techniques, require little domain information and are relatively easy to implement and use. As pointed out by different authors [79, 7], finding an approximation to the Pareto front is by itself a bi-objective problem whose objectives are:

1. minimize the distance of the generated vectors to the true Pareto front, and
2. maximize the diversity of the achieved Pareto front approximation.

Therefore, the fitness assignment scheme must consider these two objectives.

1.3.1 Nondominated Sorting Genetic Algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb [65] and is a variation of Goldbergs approach [17]. The NSGA is based on several layers of classifications of the individuals. Before selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diveristy of the population, these classified individuals are shared with their dummy fitness values. Then, this group of classified individuals is ignored and another layer of nondominated individuals is considered. The process continues until all individuals in the population are classified. Stochastic remainder proportionate selection is adopted for this technique. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. This allows for a better search of the different nondominated regions and results in convergence of the population toward such regions. Sharing, by its part, helps to distribute the population over this region. As a result, one might think that this MOEA converges rather quickly; however, a computational bottleneck occurs with the fitness sharing mechanism. An improved version of the NSGA algorithm, called Non-dominated Sorting Genetic Algorithm II (NSGA-II) was proposed by Deb et al. [11]. The NSGA-II builds a population of competing individuals, ranks and sorts each individual according to its nondomination level, it applies evolutionary operators to create a new offspring pool, and then combines the parents and offspring before partitioning the new combined pool into fronts. For each ranking level, a crowding distance is estimated by calculating the sum of the Euclidean distances between the two neighboring solutions from either side of the solution along each of the objectives. Once the nodomination rank and the crowding distance is calculated, the next population is stated by using the crowded-comparison operator ($\prec_n$). The crowded-comparison operator guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front. Assuming that every individual in the population has two attributes: 1) nondomination rank (i_{rank}) and 2) crowding distance ($i_{distance}$), the partial order $\prec_n$ is defined as:

$$\begin{aligned} i \prec_n j : \quad & \text{if } (i_{rank} < j_{rank})\text{or} \\ & ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})) \end{aligned} \tag{1.8}$$

That is, between two solutions with differing nondomination ranks, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then the solution that is located in a lesser crowded region is preferred. Algorithm 1 presents the outline of the NSGA-II, which (in the last decade) has been the most popular MOEA, and it is frequently adopted to compare the performance of newly introduced MOEAs.

```
Input:
$N$: the population size;
$T_{max}$: the maximum number of generations;
Output:
$A$: the final approximation to the Pareto optimal front;
begin
    $t = 0$;
    Generate a random population $P_t$ of size $N$;
    Evaluate the population $P_t$;
    while $t < T_{max}$ do
        Generate the offspring population $Q_t$ by using binary tournament and genetic
        operators (crossover and mutation);
        Evaluate the offspring population $Q_t$;
        $R_t = P_t \cup Q_t$;
        Rank $R_t$ by using nondomited sorting to define $\mathcal{F}$; // $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all
        nondominated fronts of $R_t$
        $P_{t+1} = \emptyset$ and $i = 1$;
        while $(|P_{t+1}| + |\mathcal{F}_i| \leq N)$ do
            Assign crowding distance to each front $\mathcal{F}_i$;
            $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$;
            $i = i + 1$;
        end
        Sort $\mathcal{F}_i$ by using the crowded-comparison operator;
        $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$;
        $t = t + 1$;
    end
    $A = P_t$;
end
```

Algorithm 1: General Framework of NSGA-II

1.3.2 Strength Pareto Evolutionary Algorithm

The Strength Pareto Evolutionary Algorithm (SPEA) was introduced by Zitzler and Thiele [79]. This evolutionary approach integrates some successful mechanisms from other MOEAs, namely, a secondary population (external archive) and the use of Pareto ranking. SPEA uses an external archive containing nondominated solutions previously found. At

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

Algorithm 2: General Framework of SPEA2

each generation, nondominated individuals are copied to the external nondominated set. In SPEA, the fitness of each individual in the primary population is computed using the individuals of the external archive. First, for each individual in this external set, a strength value is computed. The strength, $S(i)$, of individual i is determined by $S(i) = \frac{n}{\overline{N}+1}$, where n is the number of solutions dominated by i, and $\overline{N}$ is the size of the archive. Finally, the fitness of each individual in the primary population is equal to the sum of the strengths of all the external members that dominate it. This fitness assignment considers both closeness to the true Pareto front and even distribution of solutions at the same time. Thus, instead of using niches based on distance, Pareto dominance is used to ensure that the solutions are properly distributed along the Pareto front. Since the size of the archive may grow too large, the authors employed a technique that prunes the contents of the external nondominated set so that its size remains below a certain threshold. There is also a revised version of SPEA, called Strength Pareto Evolutionary Algorithm 2 (SPEA2) [77]. SPEA2 has three main differences with respect to its predecessor: 1) it incorporates a fine-grained fitness assignment strategy which takes into account, for each individual, the number of individuals that dominate it and the number of individuals to which it dominates; 2) it uses a nearest neighbor density estimation technique which guides the search more efficiently, and it has an enhanced archive truncation method that guarantees the preservation of boundary solutions. The outline of the SPEA2 is shown in Algorithm 2.

In detail, let $\overline{P}_t$ and P_t be the external archive and the population at the t generation. Each individual i in the external archive $\overline{P}_t$ and the population P_t is assigned a strength value $S(i)$, representing the number of solutions it dominates: $S(i) = |j|j \in P_t + \overline{P}_t \wedge i \prec j|$, where $| \cdot |$ denotes the cardinality of a set, $+$ stands for multiset union and the symbol $\prec$ corresponds to the Pareto dominance relation. On the basis of the S values, the raw fitness $R(i)$ of an individual i is calculated:

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

Although the raw fitness assignment provides a sort of niching mechanism based on the concept of Pareto dominance, it may fail when most individuals do not dominate each other. Therefore, additional density information is incorporated to discriminate between individuals having identical raw fitness values. The density estimation technique used in SPEA2 is an adaptation of the k-th nearest neighbor method, and it is calculated by:

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

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

$$F(i) = R(i) + D(i) \tag{1.9}$$

During environmental selection, the first step is to copy all nondominated individuals. If the nondominated front fits exactly into the archive ($|\overline{P}_{t+1}| = N$) the environmental selection step is completed. Otherwise, there can be two situations: Either the archive is too small ($|\overline{P}_{t+1}| < N$) or too large ($|\overline{P}_{t+1}| > N$). In the first case, the best $\overline{N} - |\overline{P}_{t+1}|$ dominated individuals in the previous archive and population are copied to the new archive. This can be implemented by sorting the multiset $P_t + \overline{P}_t$ according to the fitness values and copy the first $\overline{N} - |\overline{P}_{t+1}|$ individuals i with $F(i) \geq 1$ from the resulting ordered list to $\overline{P}_{t+1}$. In the second case, when the size of the current nondominated (multi)set exceeds $\overline{N}$, an archive truncation procedure is invoked which iteratively removes individuals from $\overline{P}_{t+1}$ until $|\overline{P}_{t+1}| = \overline{N}$. Here, at each iteration that individual i is chosen for removal for which $i \leq_d j$ for all $j \in \overline{P}_{t+1}$ with

$$\begin{aligned} i \leq_d j \quad \iff \quad & \forall 0 < k < |\overline{P}_{t+1}| : \sigma_i^k = \sigma_j^k \quad \vee \\ & \exists 0 < k < |\overline{P}_{t+1}| : [(\forall 0 < l < k : \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^k] \end{aligned}$$

where σ_i^k denotes the distance from i to its k-th nearest neighbor in $\overline{P}_{t+1}$. In other words, the individual which has the minimum distance to another individual is chosen at each stage; if there are several individuals with minimum distance, the tie is broken by considering the second smallest distances and so forth.

1.3.3 Multi-Objective Evolutionary Algorithm Based on Decomposition

The Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) was introduced by Zhang and Li [74]. MOEA/D explicitly decomposes the MOP into several scalar optimization subproblems. It is well-known that a Pareto optimal solution to a MOP, under certain conditions, could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the functions f_i's. Therefore, an approximation of the Pareto optimal front can be decomposed into a number of scalar objective optimization subproblems. This is a basic idea behind many traditional mathematical programming methods for approximating the Pareto optimal front. Several methods for constructing aggregation functions can be found in [13, 45]. This basic idea of decomposition is used by MOEA/D, and it solves these subproblems simultaneously by evolving a population of solutions. At each generation, the population is composed of the best solution found so far (i.e., since the start of the run of the algorithm) for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions to two neighboring subproblems should be very similar. Each subproblem (i.e., each scalar aggregation function) is optimized in MOEA/D by using information only from its neighboring subproblems. To obtain a good representation of the Pareto optimal front, a set of evenly spread weighting vectors needs to be previously generated.

Considering N as the number of scalar optimization subproblems and $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ as the set of weighting vectors which defines such subproblems, MOEA/D finds the best solution to each subproblem along the evolutionary process. Assuming the Tchebycheff approach (1.7), the fitness function of the i^{th} subproblem is stated by $g(\mathbf{x}|\mathbf{w}_i, \mathbf{z})$. MOEA/D defines a neighborhood of each weighting vector $\mathbf{w}_i$ as a set of its closest weighting vectors in W. Therefore, the neighborhood of the i^{th} subproblem consists of all the subproblems with the weighting vectors from the neighborhood of $\mathbf{w}_i$ and it is denoted by $B(\mathbf{w}_i)$. At each generation, MOEA/D maintains: 1) a population of N points $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathcal{X}$. is the current solution to the i^{th} subproblem; 2) $FV^1, \ldots, FV^N$, where FV^i is the F-value of $\mathbf{x}_i$, i.e., $FV^i = \mathbf{F}(\mathbf{x}_i)$ for each $i = 1, \ldots, N$; 3) an external archive EP, which is used to store the nondominated solutions found during the search. In contrast to NSGA-II and SPEA2 which use density estimators (crowding distance and neighboring solutions, respectively), MOEA/D uses a well distributed set of weighting vectors for guiding the search, and therefore, multiple solutions along the Pareto optimal set are maintained. With that, the diversity in the population of MOEA/D is implicitly maintained. For an easy interpretation of MOEA/D, it is outlined in Algorithm 3.

1.3.4 Memetic Algorithms

The term Memetic Algorithm (MA) was first introduced in 1989 by Pablo Moscato [47]. The term "memetic" has its roots in the word "meme" introduced by Richard Dawkins in 1976 [10] to denote the unit of imitation in cultural transmission. The essential idea behind MAs is the combination of LS refinement techniques with a strategy based on a population, such as EAs. The main difference between genetic and memetic algorithms is the approach and view of the information's transmission techniques. In the GAs, the genetic information carried by genes is usually transmitted intact to the offspring, meanwhile in the MA, the base units are the so-called "memes" and they are typically adapted by the individual transmitting information. While GAs are good at exploring the solution space from a set of candidate solutions, MAs explore from a single point, allowing to exploit solutions that are close to the optimal solutions. The main design goal of such an approach will be the efficiency of the final algorithm (i.e., the MA approach should perform a reduced number of objective function evaluations as compared to state-of-the-art EA) on standard test functions. During the last years, MAs have been successfully applied to find solutions of SOPs. In the multi-objective case, first efforts have been done over discrete domain problems; followed by an increasing interest for the continuous case. Some important decisions should be taken when mixing these techniques, they are:

a) In which moment, of the evolutionary iteration should the LS be performed?

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

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

Algorithm 3: General Framework of MOEA/D

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

c) From which population solutions should the LS be started?.

While answering the above questions it is necessary to keep in mind the evident trade-off between the computational cost and the benefits of the LS procedure. This makes important to focus on the development of suitable methods to perform LS. Because of this, mathematical methods based on particular geometrical properties of the MOP would produce good results. In particular, methods that use gradient information will produce excellent approximations; but in practice, this is not always the ideal strategy, since they imply a high computational cost. From this point of view, other choices have shown advantages in practice, producing results of comparative quality. There is no specific method to design a MA. However, Algorithm 4 shows a general framework of what a MA should contain.

Hybridization of MOEAs with LS algorithms has been investigated for more than one decade [32]. Some of the first memetic MOEA for models on discrete domains were presented, in [26, 27]. These include Multi-Objective Genetic Local Search (MOGLS), and Pareto Memetic Algorithm (PMA) [28]; these two approaches use scalarization functions. In [31], a proposal employing a Pareto ranking selection (called Memetic Pareto Archived Evolution

```
Input:
$N$: the population size;
$T_{max}$: the maximum number of generations;
Output: $A$: the final population
begin
    $t = 0$;
    Generate a random population $P_t$ of size $N$;
    Evaluate the population $P_t$;
    while $t < T_{max}$ do
        Generate the offspring population $Q_t$ by using stochastic operators;
        Evaluate the offspring population $Q_t$;
        Select a set of solutions $R_t$ from $Q_t$;
        forall the $r^t \in R_t$ do
            IMPROVE: $r^t = i(r^t)$ // using an improvement mechanism;
        end
        Set $P_{t+1}$ as the set of $N$ solutions selected from $P_t \cup Q_t \cup R_t$ according to any
        rule of selection (commonly using the fitness of each individual);
        $t = t + 1$;
    end
    $A = P_t$;
end
```

Algorithm 4: General scheme of a MA

Strategy (M-PAES)) was introduced. Also, in [49], the authors proposed a LS process with a generalized replacement rule based on the dominance relation. In [6], the Cross Dominant Multi-Objective Memetic Algorithm (CDMOMA) was proposed as an adaptation of the NSGA-II, and two local search engines: a multi-objective implementation of Rosenbrock's algorithm [56], which performs very small movements, and the Pareto Domination Multi-Objective Simulated Annealing (PDMOSA) [67], which performs a more global exploration. A memetic version of Coevolutionary Multi-Objective Differential Evolution (CMODE) was proposed in [64] and was called CMODE-MEM. Most of this work has been proposed for combinatorial problems. For the continuous case—i.e., continuous objectives defined on a continuous domain— the first attempts started, to the authors' best knowledge, with [16], where a neighborhood search was applied to NSGA-II. This is a very simple scheme and the authors found that the added computational work had a severe impact on the efficiency of the algorithm. In the following sections, we provide an in-depth analysis of the continuous domain case.

1.4 Methods Based on Descent Directions

Many mathematical programming techniques are traditionally based on differentiable properties of the objective functions. These tools are popular since they have strong foundations, and have been intensively taught to engineers through many decades. Exploiting particular knowledge about the problem is a very common feature of almost all traditional optimization techniques. For example, some of the methods developed in [29], use linear properties of the functions and have a well-founded theoretical basis. Other methods, such as those explained in [45], succeed on solving nonlinear MOPs, when using just differentiability properties of the functions. One drawback of all of these (traditional) methods is precisely that they cannot be applied to a wide variety of MOPs, because of their special assumptions. On the other hand, it is worth noticing that when certain conditions are fulfilled, they are indeed very efficient. Because of the nature of differentiability theory, these techniques can be seen as LS procedures; this is, they can only guarantee the approximation of local optima points. Talking specifically about differentiable problems, in this section we focus mainly on those methods that use gradient-based descent directions to perform line search. Remarkable features of these classic methods are the following:

- They generate new candidate solutions in an iterative way. For each iteration, at least an approximation of the function gradient is required; second order information (Hessian matrix) is also necessary, in some cases.
- They take advantage of a starting point for triggering the search, and they normally produce a single solution per run (in fact, in some cases, different starting points may lead to the same final objective value).
- They can only guarantee that the final point is optimum in a local neighborhood.

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 population-based heuristic is an obvious choice. Several proposals, like those developed in [22, 5, 62, 2, 25], are based on this hypothesis, i.e., they attempt to show that the combination of gradient-based methods and MOEAs can boost performance.

Directing the search, from a particular solution, towards a special improvement direction is a widely used idea in optimization. This procedure is known as *line search.* It starts with a particular (non-optimal) vector solution $\boldsymbol{x_0}$, and is composed by two goals:

1. First, it is necessary to find a promising search direction $\boldsymbol{\nu} \in \mathbb{R}^n$ in which the function value improves. This is commonly based on information related to the gradient of the objective function.
2. Once the direction $\boldsymbol{\nu}$ is obtained, it is necessary to state a suitable step size t^* to move from $\boldsymbol{x_0}$ in this direction, such that this movement best improves the objective value of the new iterate solution $\boldsymbol{x_1}$.

For example, when minimizing

$$f : \mathbb{R}^n \to \mathbb{R}$$

to solve a SOP, the well-known *steepest descent method* is based on an iteration given by:

$$\boldsymbol{x^{(1)}} = \boldsymbol{x^{(0)}} - {t^*}^{(0)} \boldsymbol{\nabla f(x)}.$$

When trying to use a similar procedure for the multi-objective optimization case, the first goal is to find a descent direction to be equivalent to the role played by the gradient of a function in the SOP case. Let's remember that, for MOPs, m conflicting objective functions are involved. So, in order to do this, the gradient of each of the objective functions should be combined. The idea is then to find a direction

$$\boldsymbol{\nu} \in \mathbb{R}^n$$

such that, moving along $\boldsymbol{\nu}$ decrements the value of each objective function $f_i(\boldsymbol{x})$ simultaneously. This is stated on the next definition:

Definition 6 *Let* $f_1, \ldots, f_m$ *be the objective functions of a MOP. A Multi-objective Descent Direction (MDD) is a direction* $\boldsymbol{\nu} \in \mathbb{R}^n$ *such that the directional derivatives* $D_{\boldsymbol{\nu}} f_i(\boldsymbol{x})$ *with respect to* $\boldsymbol{\nu}$*, at the point* $\boldsymbol{x} \in \mathbb{R}^n$*, are non-positive, i.e,* $D_{\boldsymbol{\nu}} f_i(\boldsymbol{x}) = \langle \nabla f_i(\boldsymbol{x}), \boldsymbol{\nu} \rangle \leq 0$ *for all* $i \in 1, \ldots, m$ *without allowing them to be all equal to zero.*

Then, starting from a solution $\boldsymbol{x} \in \mathbb{R}^n$, when we perform a small movement over a MDD $\boldsymbol{\nu}$, we will get a local improvement simultaneously for all the objective functions. The computation of a MDD is a procedure that regularly implies the solution of another optimization problem. Different proposals are available to set this problem; we explain some of these methods in the following.

We will start by first noticing that, given $i \in \{1 \ldots m\}$, any direction $\boldsymbol{\nu}$ such that $\langle \boldsymbol{\nu}, \nabla f_i(\boldsymbol{x}) \rangle < 0$ is a descent direction for f_i at $\boldsymbol{x}$. Figure 1.3 shows how, for any point $\boldsymbol{x}$, the search space is split into two semi-spaces, one of them corresponding to descent directions. From definition 6, it follows that, considering the intersection of these semi-spaces provides a number of MDDs.

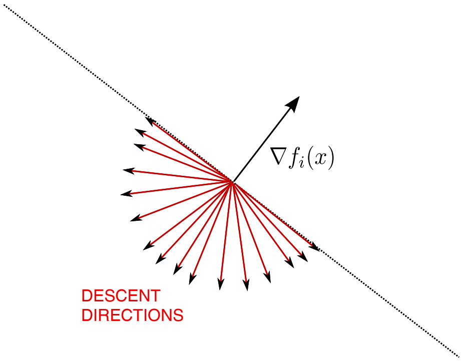

FIGURE 1.3: This figure illustrates the descent directions for a particular function f_i.

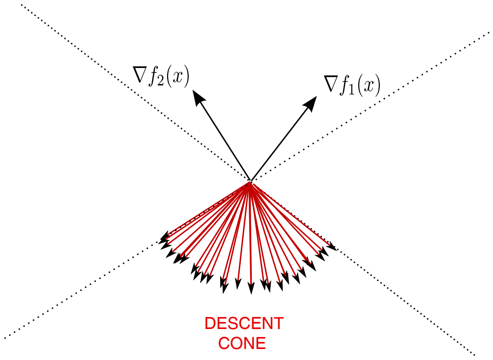

FIGURE 1.4: This figures illustrates the descent cone which encloses multi-objective descent directions.

It is worth noticing that the problem of finding an appropriate descent direction is again a MOP. To explain this, let's assume that two different vectors $\boldsymbol{\nu_1}$ and $\boldsymbol{\nu_2}$ are descent directions. And also that,

$$D_{\boldsymbol{\nu_1}} f_i(\boldsymbol{x}) \leq D_{\boldsymbol{\nu_2}} f_i(\boldsymbol{x})$$

and

$$D_{\boldsymbol{\nu_2}} f_i(\boldsymbol{x}) \leq D_{\boldsymbol{\nu_1}} f_i(\boldsymbol{x}),$$

hold. Then, even when both directions get simultaneous improvements, it is not possible to decide which one is best—at least not without setting a preference between the objective functions. This is illustrated in Figure 1.5. Since finding a MDD led us to face a MOP again, it should be clear that the "best" direction is not defined in the MOP context.

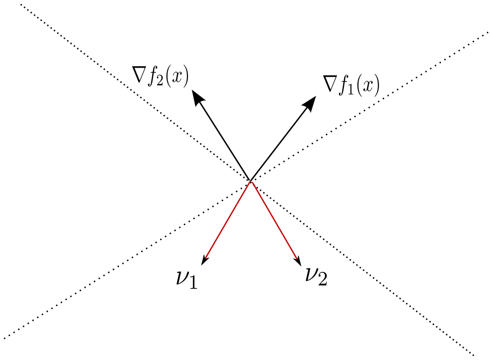

FIGURE 1.5: This figure illustrates two multi-objective descent directions $\boldsymbol{\nu_1}$ and $\boldsymbol{\nu_2}$ for which the simultaneous objective function improvement is incomparable.

Given a point $\boldsymbol{x} \in \mathbb{R}^n$ which is not a KKT point, solving a quadratic programming problem leads to a descent direction, as the next theorem says.

Theorem 1 *[58] Given an unconstrained MOP, as in (1.1), and* $q : \mathbb{R}^n \to \mathbb{R}^n$ *be defined by*

$$q(\boldsymbol{x}) = \sum_{i=1}^{m} \hat{\alpha}_i \boldsymbol{\nabla f_i}(\boldsymbol{x}), \tag{1.10}$$

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

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \left\{ || \sum_{i=1}^{m} \alpha_i \boldsymbol{\nabla f_i(x)} ||_2^2 ; \alpha_i \geq 0, i = 1, \ldots, m, \sum_{i=1}^{m} \alpha_i = 1 \right\}, \tag{1.11}$$

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

Proof 1 *See [58]*

An alternative way to get a MDD can be found in [15]; the direction computed in that way is called by its authors as the *Steepest Descent Direction.* The method works for convex Pareto fronts as well as for concave Pareto fronts. In this case, a MDD $\boldsymbol{\nu}$ can be computed using the information provided by the Jacobian $J_{F(\mathbf{x})}$ of the problem F evaluated on $\mathbf{x}$. It is, then, necessary to solve the following quadratic programming problem:

$$\text{Minimize} \quad \alpha + \frac{1}{2} ||\boldsymbol{\nu}||^2 \tag{1.12}$$

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

The above problem produces a solution

$$(\boldsymbol{\nu^*}, \alpha^*),$$

where the descent direction that we are looking for is $\boldsymbol{\nu^*}$. This method triggers automatically, similar to the Theorem 1, by using a condition that verifies if $\mathbf{x}$ fulfills condition (1.4), which is the case when

$$\alpha^* = 0.$$

In practice, it is necessary to set a tolerance parameter τ, such that $\tau < 0$ to stop the descent when

$$\tau \leq \alpha^*$$

holds; note that, by construction, $\alpha^* \leq 0$.

For two-objective problems, there is a simple way to compute a MDD with no cost, other than the gradient approximation for each function. In this case, the descent direction $\overline{\nu}_x$ can be obtained by

$$\overline{\nu}_{\boldsymbol{x}} = -\left(\frac{\nabla f_1(\boldsymbol{x})}{||\nabla f_1(\boldsymbol{x})||} + \frac{\nabla f_2(\boldsymbol{x})}{||\nabla f_2(\boldsymbol{x})||} \right), \tag{1.13}$$

where $||\cdot|| = ||\cdot||_2$ is the Euclidean norm, and $\nabla f_1(\boldsymbol{x}), \nabla f_2(\boldsymbol{x})$ are the gradients of the objective functions, at solution x. The reader can refer to [38] for details and a proof that 1.13 leads to a MDD. Also for a possible hybridization of this procedure with NSGA-II, it is important to notice that this simple formula can not be generalized for problems with more than two objectives.

Once the MDD is set, a regular line search can be performed. Computation of step length in the multi-objective context is an open problem, since each objective function will have its own optimal step size (see [41]). In practice, an Armijo like rule, or a sequential quadratic approximations approach have been commonly used. Then, after computing a reasonable step length t, and getting $\mathbf{x^{(1)}} = \mathbf{x} + t\boldsymbol{\nu}$, we are in condition to repeat the movement by calculating a new descent direction from $\mathbf{x^{(1)}}$ or, if this is not possible, we can assume that a critical point has been achieved. Other ways to calculate MDD have been proposed [2, 3, 5, 22]. 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 (see [40]).

To show the coupling of this LS procedure with a MOEA, we chose the widely adopted NSGA-II as our global search engine. Nevertheless, the coupling with other MOEA is also possible. We present in Algorithm 5 a simple version of a MOEA hybridized with a gradient-based LS procedure. The parameters N and G in Algorithm 5, represent the population size and the maximum number of generations. The procedures "Fast Non-Dominated Sort", "Crowding Distance Assignment" and "Generate Offspring Population" correspond to the well-known components of the NSGA-II. Algorithm 5 places the LS inside the NSGA-II just after the reproduction and the ranking-crowding process. LS is applied only to non-dominated individuals, but not to all of them. In Algorithm 6, the LS procedure is described.

1.5 Gradient-based Numerical Continuation

A method to deal with both, unconstrained and constrained MOPs, is shown by Hillermeier in [23]. He introduces an homotopy approach using differential geometry and pa-

```
Input:
$G$: maximum number of generations;
Output:
$P$: final approximation to the Pareto front;
begin
    Generate a Random Population $P$ of size $N$;
    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;
        Apply Local Search using $(i, P)$;
    end
end
```

Algorithm 5: Memetic NSGA-II using gradient-based line search.

```
Input:
$i$ : generation number;
$\xi$ : rule to establish the frequency of application of LS;
$\eta$ : rule to select the individuals to which the LS is applied;
$P$ : population before applying the LS;
Output:
$P$ : population after applying the LS;
begin
    if $i$ fullfils $\xi$ then
        Conform the set $E \subset P$ according to $\eta$, with randomly selected individuals
        from the non-dominated set $R_1 \subset P$;
        for $a \in E$ do
            if local improvement is possible then
                Apply line search to obtain $a'$;
                Replace $a \leftarrow a'$;
                Set $a' \in R_1 \subset P$;
                Set the crowding distance of $a'$ as $\infty$;
            end
        end
    end
end
```

Algorithm 6: Gradient-based line search procedure.

rametric-optimization concepts to extend the MOP with k objective functions on an n-dimensional space to an auxiliary problem. This is, when having a Pareto point $\boldsymbol{x}$, a necessary condition (see [45]) is the existence of

$$\boldsymbol{\alpha} \in \mathbb{R}^m \text{ with } \sum_{i=1}^{m} \alpha_i = 1$$

such that

$$\sum_{i=1}^{m} \alpha_i \nabla f_i(\boldsymbol{x}) = \mathbf{0}.$$

Based on this, it is possible to construct a suitable function

$$\widetilde{F} : \mathbb{R}^{n+m} \longrightarrow \mathbb{R}^{m+1}$$

defined by

$$\widetilde{F}(\boldsymbol{x}, \boldsymbol{\alpha}) = \begin{pmatrix} \sum_{i=1}^{m} \alpha_i \nabla f_i(\boldsymbol{x}) \\ \sum_{i=1}^{m} \alpha_i - 1 \end{pmatrix}. \tag{1.14}$$

Then, for every Pareto point

$$\boldsymbol{x}$$

there exists a vector

$$\boldsymbol{\alpha^*} \in \mathbb{R}^m$$

such that

$$\widetilde{F}(\boldsymbol{x}, \boldsymbol{\alpha^*}) = \mathbf{0}.$$

With this auxiliary function, it is possible to show [23] that the Pareto set of the original problem forms 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 procedure also calculates a set of Pareto optimal points in a single run, it is of local nature. Nevertheless, it is worth noticing that this drawback can be avoided if the method is complemented with a stochastic technique—such as a MOEA.

An interesting way to use local search with stochastic methods is to refine the solutions obtained by a population-based algorithm. As an example, multi-objective continuation methods have been used [61] as a recovering technique to complement the application of a Multi-objective Particle Swarm Algorithm [7]. It is worth noticing that the Particle Swarm heuristic could be replaced, in this algorithm, by any other MOEA.

When using the continuation part of [61], it is assumed that a Pareto optimal point

$$\boldsymbol{x_0}$$

is given. Then, a linearization of the Pareto front, at the solution $\boldsymbol{x_0}$, is computed using second order information. In this way, a predictor-corrector method is used to follow the frontier of the Pareto Front, and a new candidate point

$$\boldsymbol{x_0'}$$

is obtained. The possible error in the prediction is corrected by the solution of equation (1.11).

It is worth noticing that, as with Hillermeier's method, this predictor-corrector method can only detect connected components of the Pareto set; then, the entire procedure can be seen as a local-search technique. This observation means an important motivation to combine these techniques with an stochastic heuristic, when looking for global solutions of MOP. A key element of this mix is the archiving method. This is used to manage the solutions generated by both techniques, in different moments. It is because of this feature that the convergence of the entire method can be ensured, in a theoretical sense. Also, the data structure to storage the information is based on data trees—which reduce the

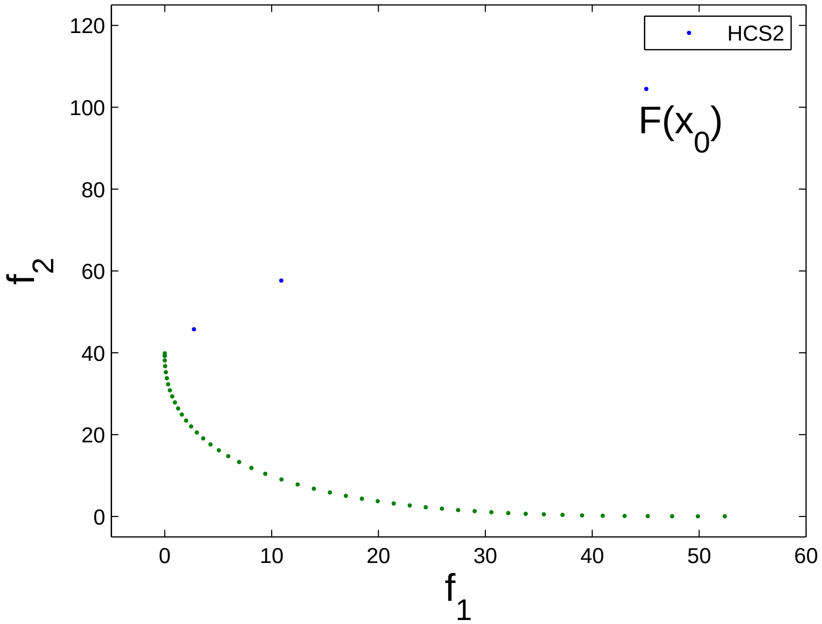

FIGURE 1.6: This figure plots the set of points obtained by the local operator HCS2 as a stand-alone procedure.

computational complexity of the information access. This information is related to specially designed boxes which contain, and manage, the approximation points.

As another example, in [39] a local search engine called HCS2—Hill climber with side step 2—is conformed by a hill climber part and a one-step continuation. The hill climber part is performed by a method based on Fliege's method [15]. Alternatively, Shaffler's method (equation (1.11)) can also be used. In both cases, a stopping criterion is automatically given, when the MDD cannot be computed since a Pareto optimal point has been reached. This is done by a certain tolerance value, for computational purposes, to assess when a quantity is small enough—and can then be considered as zero. When this condition is fulfilled, a side-step is performed—assuming that the Pareto front has been reached, and a movement over the frontier is suitable. The side-step part of the operator is performed by a technique based on the continuation proposed at [23]. It uses again second order information and intends to generate two more points at the Pareto Front with an affordable computational cost. Figure 1.6 shows the points obtained by the repeated execution of the HCS2, as a stand-alone procedure, over a convex problem (in convex problems the local optima are global also). In [39], a second type of local search engine is proposed—named as HCS1—but since it does not use gradient information it is not described in this section. Each of these local search engines were combined with the NSGA-II and SPEA2. The four memetic algorithms were tested on two and three objective standard test functions; and those experiments showed that the memetic approach outperformed the plain MOEA, in problems with a moderate number of local Pareto points.

Finally, it is worth mentioning that, in [61, 59, 21, 60], other hybrids can be found, in which heuristic methods are coupled with particular multi-objective continuation methods.

1.6 Reference point methods

In [54], the authors proposed a hybrid technique that combines the robustness of MOGA-II [51] with the accuracy and speed of NBI-NLPQLP. In [69], the proposed LS process employs quadratic approximations for all the objective functions.

An important work which explicitly emphasizes the benefits of using local search to obtain accurate solutions was presented in [63]. The authors combined the NSGA-II with a reference point method. Applying this method, the authors were able to accelerate the convergence of NSGA-II. In this work, the authors proposed a concurrent approach, instead of a serial approach—in which the local search is applied after the population-based algorithm finishes. In the concurrent approach, both techniques interact over the same generation of individuals, at different times of the evolutionary process. This avoids the problem of having to set, *a priori*, a specific number of resources to be spent by each technique.

Then, applying local search to a specific individual $\boldsymbol{y}$ from the population consists of solving the next problem for $\boldsymbol{x}$:

$$\text{Minimize} \quad \max_{i=1}^{m} \frac{f_i(\boldsymbol{x}) - z_i}{f_i^{max} - f_i^{min}} + \rho \sum_{j=1}^{m} \frac{f_j(\boldsymbol{x}) - z_j}{f_j^{max} - f_j^{min}} \tag{1.15}$$

where $\boldsymbol{x}$ is subject to $\boldsymbol{x} \in S$, with S being the feasible region. In this case, to take advantage of the population information into the procedure, f_i^{max} and f_i^{min} are the maximum and minimum values taken from the entire population, at a certain generation, respectively. Also, $\boldsymbol{z}$ is the so-called reference point defined by

$$\boldsymbol{z} := F(\boldsymbol{y}),$$

with $\boldsymbol{y}$ being the individual from the population which has been chosen to be used as a departing point for the local search procedure. A suggested value is

$$\rho = 10^{-2}.$$

Like other gradient-based local searchers, this method allows to use the KKT conditions to terminate the LS. Anyway, the authors proposed to check a secondary stopping condition, when the local search had elapsed more that 25 function calls. This avoids consuming too many resources for the execution of the LS.

Besides, in order to balance the cost of both techniques, the LS is applied with a certain probability

$$p_{ls}$$

at each generation. The function used [63] is defined as follows: starting from zero at the beginning of the evolutionary process ($generation = 0$), the probability of applying LS increases up to 0.01. in

$$0.5N - 1$$

generations. Here, N is the population size. Then, the probability drops until reach zero after spending

$$0.5N$$

generations. This way to manage the probability has several motivation. One is to increase linearly the amount of resources spent by the LS. Another one is to avoid the application of LS at the beginning of the process, which is particularly convenient for the specific LS method chosen in this case. A final reason is to have a restart mechanism in order to prevent loss of diversity during the evolutionary process.

This method was tested with the Zitzler-Deb-Thiele (ZDT) benchmark problems with good results. It showed that the application of this kind of LS was benefits in terms of the speed of convergence and the final accuracy of the solutions obtained. In these experiments, the chosen MOEA to be hybridized was NSGA-II.

1.7 Other Gradient-based Approaches

In [2], the problem of finding an improvement direction for the multi-objective case is stated as a multi-objective problem as well. Therefore, the solution (*i.e.,* the "multi-objective gradient") must be a set of suitable movement directions. When performing a search over any of these directions, 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 of this method is to describe a set of descent directions rather than a single one. An analytical way to calculate this set of directions is also shown. This 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 [2]. A later improvement for CORL (by combining it with other criteria) can be found in [3]. In both cases, the hybridization of CORL is made by a a combination with an Estimation Distribution Algorithm (EDA) called $\mathbb{MIDEA}$ [4]. The hybridization is made in the following manner: at the end of the generational cycle, *i.e.*, after the variation operators have been applied to the population, the algorithms choose a specific set of individuals in order to perform the local search over them. Thus, an improvement on the fitness for each one of them is obtained.

A revision of this method was presented by Harada et al. [22]. They used the ideas introduced in [15] 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 those cases, it is necessary to find different descent directions. The algorithmic complexity of PDM is again polynomial—as its authors refer—because the basic operations of this method consist of solving systems of linear equations. Same as with other gradient-based local search engines, PDM performs successfully on MOPs with no local Pareto fronts. Harada et al. [22] compared PDM against CORL and also against a simple weighted linear aggregating function. They also evaluated a randomized generator of solutions, similar to the mutation mechanism used in Evolution Strategies—they concluded that this last method was 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 memetic algorithm based on PDM; this is left by the authors as future work. In the same work, the authors stated that their method could have scalability issues when more than three objectives are used.

In [62], Shukla introduced the use of two stochastic gradient-based techniques to improve the mutation mechanism of NSGA-II. These two techniques are Schäffler's stochastic method [58] and Timmel's [68] 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 improved, but they also illustrate the local nature of the gradient-based information and its possible limitations. As an example, Figure 1.7 shows the set of points generated by the HCS2, a more sophisticated local search engine which is obviously trapped in false fronts at the same test problem indicated before (ZDT4).

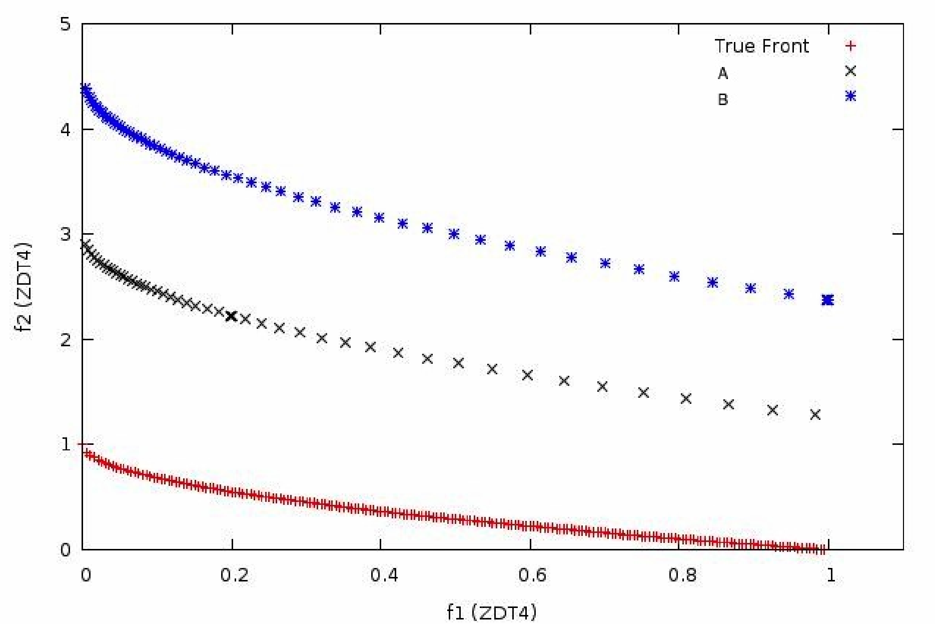

FIGURE 1.7: This figure shows the set of points generated by the repeated application of the operator HCS2, starting from two different points A and B. In this test problem (ZDT4) local Pareto fronts are formed from different points, far away from the global one.

Sequential quadratic programming: In [25] a gradient-based local algorithm (Sequential Quadratic Programming (SQP)), was used in combination with NSGA-II and SPEA [79] to solve the ZDT benchmark suite [76]. 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.

1.8 Approaches based on Nelder and Mead's Algorithm

The Multi-Objective Memetic Algorithms (MOMAs) presented in the above sections, require the gradient information of the functions. Therefore, their use is limited to certain types of problems. This has motivated the development of new approaches that couple direct search methods (i.e., methods that do not require gradient information) with a MOEA. Among the mathematical programming techniques available for this coupling, Nelder and

Mead's algorithm [50] is perhaps the most obvious choice, since it is the most popular direct search method adopted for solving unconstrained optimization problems. Nelder and Mead's method, also known as the Nonlinear Simplex Search (NSS), has been used extensively to solve parameter estimation problems as well as other optimization problems since its inception, in 1965. The search done by the NSS is based on geometric operations (reflection, expansion, contraction and shrinkage) on a set of points, which define an n-dimensional polygon called "simplex". Mathematically, an n-simplex can be defined as follows.

Definition 7 *A simplex or n-simplex Δ is the convex hull of a set of $n+1$ affinely independent points Δ_i $(i = 1, \dots, n+1)$, in some Euclidean space of dimension n.*

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

The NSS expands or focuses the search adaptively on the basis of the topography of the fitness landscape. The full algorithm is defined stating three scalar parameters to control the movements performed in the simplex: reflection (ρ), expansion (χ), contraction (γ) and shrinkage (σ). According to Nelder and Mead [50], these parameters should satisfy:

$$\rho > 0, \qquad \chi > 1, \qquad \chi > \rho, \qquad 0 < \gamma < 1, \qquad \text{and} \qquad 0 < \sigma < 1 \tag{1.16}$$

The nearly universal choices used in the Nelder and Mead algorithm are:

$$\rho = 1, \qquad \chi = 2, \qquad \gamma = \frac{1}{2}, \qquad \text{and} \qquad \sigma = \frac{1}{2} \tag{1.17}$$

At each iteration of the NSS, the $n+1$ vertices Δ_i's of the simplex represent solutions which are evaluated and sorted according to: $f(\Delta_1) \leq f(\Delta_2) \leq \ldots \leq f(\Delta_{n+1})$.

Considering $\mathbf{\Delta} = \{\Delta_1, \Delta_2, \dots, \Delta_{n+1}\}$ as the simplex with the vertices sorted according to the function value, the transformations performed by the NSS into the simplex are defined as:

1. Reflection: $\mathbf{x}_r = (1+\rho)\mathbf{x}_c - \rho\Delta_{n+1}$ (see Figure 1.9).
2. Expansion: $\mathbf{x}_e = (1+\rho\gamma)\mathbf{x}_c - \rho\chi\Delta_{n+1}$ (see Figure 1.10).
3. Contraction:
   (a) *Outside*: $\mathbf{x}_{oc} = (1+\rho\gamma)\mathbf{x}_c - \rho\gamma\Delta_{n+1}$.
   (b) *Inside*: $\mathbf{x}_{ic} = (1-\gamma)\mathbf{x}_c + \gamma\Delta_{n+1}$ (see Figure 1.11).
4. Shrinkage: Each vertex of the simplex is transformed by the geometric shrinkage defined by: $\Delta_i = \Delta_1 + \sigma(\Delta_i - \Delta_1)$, $i = 2, \dots, n+1$, and the new vertices are evaluated—see Figure 1.12.

where $\mathbf{x}_c = \frac{1}{n}\sum_{i=1}^{n} \Delta_i$ is the centroid of the n best points (all vertices except for Δ_{n+1}), Δ_1 and Δ_{n+1} are the best and the worst solutions identified within the simplex, respectively. At each iteration, the simplex is modified by one of the above movements, according to the following rules:

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

In the last few years, some MOMAs that combine the NSS with different state-of-the-art MOEAs have been reported in the specialized literature. In the following, we present several hybrid approaches that have reported significant improvements with respect to the original MOEA adopted.

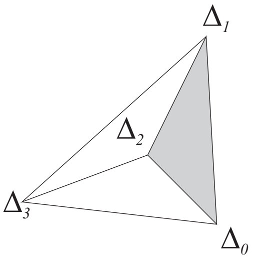

FIGURE 1.8: An n-simplex with $n = 2$

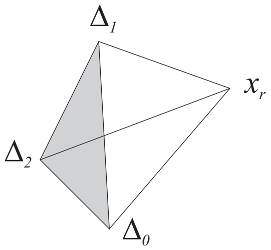

FIGURE 1.9: Reflection

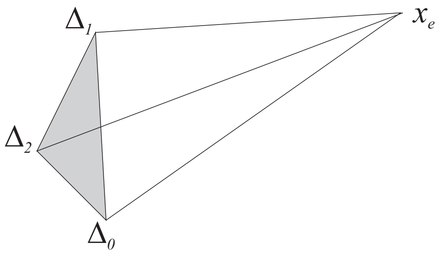

FIGURE 1.10: Expansion

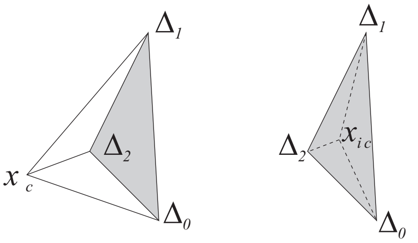

FIGURE 1.11: Inside and outside contraction

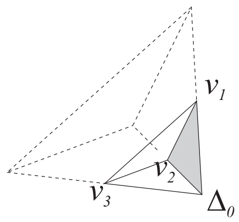

FIGURE 1.12: Shrinkage

1.8.1 A Multi-objective GA-Simplex Hybrid Algorithm

Koduru et al. [35] proposed a hybrid GA using fuzzy dominance and Nelder and Mead's algorithm. The simplex search algorithm is adopted to improve solutions in the population of a GA. This memetic approach is used to estimate the parameters of a gene regulatory network for flowering time control in rice. The memetic algorithm minimizes the difference between the model behavior and real-world data. Because of the nature of the data, a multi-objective approach is stated.

In order to understand the fuzzy dominance relation, the following definitions are introduced. Assuming a minimization problem with n decision variables and considering $\mathcal{X} \subset \mathbb{R}^n$ as the solution space, i.e., the set of all possible solution vectors, the fuzzy i-dominance by a solution is defined as follows.

Definition 8 *Given a monotonically nondecreasing function* $\mu_i^{dom} : \mathcal{X} \to [0,1], i = \{1,\dots,n\}$ *such that* $\mu_i^{dom}(0) = 0$*, solution* $\mathbf{u} \in \mathcal{X}$ *is said to* ***i*-dominate** *solution* $\mathbf{v} \in \mathcal{X}$*, if and only if* $f_i(\mathbf{u}) < f_i(\mathbf{v})$*. This relationship will be denoted as* $\mathbf{u} \prec_i^F \mathbf{v}$*. If* $\mathbf{u} \prec_i^F \mathbf{v}$*, the degree of fuzzy i-dominance is equal to* $\mu_i^{dom}(f_i(\mathbf{v}) - f_i(\mathbf{u})) \equiv \mu_i^{dom}(\mathbf{u} \prec_i^F \mathbf{v})$*. fuzzy dominance can be regarded as a fuzzy relationship* $\mathbf{u} \prec_i^F \mathbf{v}$ *between* $\mathbf{u}$ *and* $\mathbf{v}$ *[44].*

Definition 9 *Solution* $\mathbf{u} \in \mathcal{X}$ *is said to* **fuzzy dominate** *solution* $\mathbf{v} \in \mathcal{X}$ *if and only if* $\forall i \in \{1,\dots,k\}, \mathbf{u} \prec_i^F \mathbf{v}$*. This relationship will be denoted as* $\mathbf{u} \prec^F \mathbf{v}$*. The degree of fuzzy dominance can be defined by invoking the concept of fuzzy intersection [44]. If* $\mathbf{u} \prec^F \mathbf{v}$*, the degree of fuzzy dominance* $\mu^{dom}(\mathbf{u} \prec^F \mathbf{v})$ *is obtained by computing the intersection of the fuzzy relationships* $\mathbf{u} \prec_i^F \mathbf{v}$ *for each* i*. The fuzzy intersection operation is carried out using a family of functions called t-norms, denoted by* $\bigcap$*. Hence,*

$$\mu^{dom}(\mathbf{u} \prec^F \mathbf{v}) = \bigcap_{i=1}^{k} \mu_i^{dom}(\mathbf{u} \prec \mathbf{v}) \tag{1.18}$$

where k *is the number of objective functions.*

Definition 10 *Given a population of solutions* $P \subset \mathcal{X}$*, a solution* $\mathbf{v} \in P$ *is said to be fuzzy dominated in* P *iff it is fuzzy dominated by any other solution* $\mathbf{u} \in P$*. In this case, the degree of fuzzy dominance can be computed by performing a union operation over every possible* $\mu^{dom}(\mathbf{u} \prec^F \mathbf{v})$*, carried out using t-co norms, that are denoted by* $\bigcup$*. Hence, the degree of fuzzy dominance of a solution* $\mathbf{v} \in P$ *in the set* P *is given by,*

$$\mu^{dom}(P \prec^F \mathbf{v}) = \bigcup_{\mathbf{u} \in P} \mu^{dom}(\mathbf{u} \prec^F \mathbf{v}) \tag{1.19}$$

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

$$\mu_i^{dom}(\mathbf{u} \prec_i^F \mathbf{v}) = \begin{cases} 0 & \text{if } f_i(\mathbf{v}) - f_i(\mathbf{u}) < 0, \\ \frac{f_i(\mathbf{v}) - f_i(\mathbf{u})}{p_i} & \text{if } \leq f_i(\mathbf{v}) - f_i(\mathbf{u}) < p_i, \\ 1 & \text{otherwise.} \end{cases} \tag{1.20}$$

where p_i determines the length of the linear region of the trapezoid for the objective function f_i. The t-norm and t-co norms are defined as $x \cap y = xy$ and $x \cup y = x + y - xy$. Both are standard forms of operators [44].

At each generation t, a fraction of the next population P_{t+1} is obtained by performing genetic operators and the rest of the population is stated by using the NSS as it is illustrated in Figure 1.13. At the beginning of each iteration, the fuzzy dominances of all solutions in the current population P_t are calculated according to the equation (1.20). Then, the fuzzy dominance of the population is stored as a two-dimensional array, each entry of which is a fuzzy dominance relationship between two solution vectors. The first part of the population is obtained by evolving a set B of $N-(n+1)$ solutions chosen randomly from the population P_t, where N denotes the population size and n the number of decision variable of the MOP. The subpopulation B is evolved performing genetic operators (crossover and mutation) and the fuzzy dominance relation is used as a measure of fitness during the selection into the GA. The resulting offspring population Q_1 is inserted as part of the next population P_{t+1}.

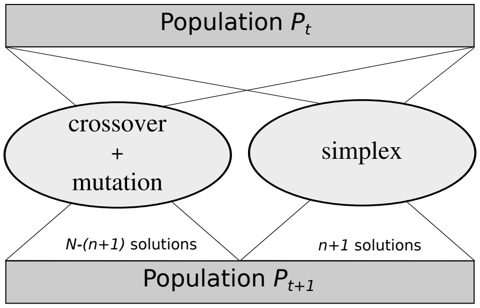

FIGURE 1.13: The offspring population generated by the multi-objective GA-Simplex Hybrid Algorithm

The second part of the population is generated by the NSS. The simplex is built by selecting a sample set S of $n+1$ solutions from the current population P_t and then, the centroid $\mathbf{c}$ of the sample S is calculated. Any solution $\mathbf{u} \in S$ at a distance $||\mathbf{c}-\mathbf{u}|| > \rho_{simplex}$ is rejected and replaced with another one taken in a random way from the population P_t, where $\rho_{simplex}$ represents the radius parameter of the simplex and $|| \cdot ||$ denotes the Euclidean norm. This process is repeated until either all the sample solutions fit within the radius $\rho_{simplex}$, or the total replacements exceed r_{max}. After selecting the initial vertices of the simplex, the NSS is performed for a total of α times. For each solution in the simplex, fuzzy dominance is calculated among solutions of the simplex and the vertices are sorted according to the fuzzy dominance relation. From the solutions obtained by the NSS, a set Q_2 of the best $n+1$ solutions (according to the fuzzy dominance relation) are selected and they are inserted into the next population P_{t+1}. The evolutionary process of this MOMA is carried out during T_{max} generations. Algorithm 7 shows the general framework of the multi-objective GA-Simplex Hybrid Algorithm.

The authors suggested the use of $\alpha = 10$ as the maximum number of iterations for the NSS. The coefficients for the reflection, expansion and contraction movements of the NSS were defined as: $\rho = 1, \chi = 1.5$ and $\gamma = 0.5$, respectively. The shrinkage step was omitted in this memetic approach. The hybrid GA was tested using a population size of $N = 100$ and it was was compared against a well-known state-of-the art MOEA: SPEA. A more detailed description of this algorithm can be found in [35].

1.8.2 A Multi-objective Hybrid Particle Swarm Optimization Algorithm

Koduru et al. [36] hybridized a multi-objective Particle Swarm Optimization (PSO) algorithm with Nelder and Mead's method. The NSS was used as a local search engine for finding nondominated solutions in the neighborhood defined by the solution to be improved. This bio-inspired technique evolves a set of solutions called swarm P for approximating solutions towards the Pareto optimal front. Each particle $\mathbf{x}_i$ in the swarm possesses a flight velocity which is initially set to zero. The swarm is evolved by updating both the velocity $\mathbf{v}_i^{t+1}$ and the position of each particle $\mathbf{x}_i^{t+1}$ according to the following equations:

$$\mathbf{v}_i^{t+1} = w(\mathbf{v}_i^t + c_1 r_1(\mathbf{x}_{pb,i} - \mathbf{x}_t^t) + c_2 r_2(\mathbf{x}_{gb,i} - \mathbf{x}_i^t)) \tag{1.21}$$

and the new particle position is updated according to equation ([30]):

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{1.22}$$

```
Input:
$N$: the population size;
$T_{max}$: the maximum number of generations;
Output:
$P$: the final approximation to the Pareto optimal front;
begin
    $t = 0$;
    Generate a random population $P_t$ of size $N$;
    Evaluate the population $P_t$;
    while $t < T_{max}$ do
        // Select the solutions for performing the evolutionary process;
        $B = x_i \in P_t$ such that: $\mathbf{x}_i$ is randomly chosen from $P_t$ and $|P_t| = N - (n+1)$;
        $Q_1 = \textsc{Mutation}(\textsc{Crossover}(B))$; // Apply genetic operators
        $S = \mathbf{x}_i \in P_t$ such that: $|S| = n+1$; // Defining the simplex
        for $j = 0$ to $j < \alpha$ do
            Execute NSS using the initial simplex $S$;
        end
        Define $Q_2$ as the best $n+1$ solutions (according to fuzzy dominance) found by the simplex search;
        $P_{t+1} = Q_1 \cup Q_2$;
        $t = t + 1$;
    end
    return $P_t$
end
```

Algorithm 7: The Multi-objective GA-Simplex Hybrid Algorithm

where $w \geq 0$ represents the constriction coefficient, $c_1, c_2 \geq 0$ are the constraints on the velocity, r_1, r_2 are two random variables having uniform distribution in the range $(0, 1)$. $\mathbf{v}_i, \mathbf{x}_{pb,i}$ and $\mathbf{x}_{gb,i}$ represent the velocity, the personal best and the global best position for the i^{th} particle, respectively. Since at the beginning of the search, a particle does not have a previous position, the best personal position is initialized with the same position as the particle, i.e., $\mathbf{x}_{pb,i} = \mathbf{x}_i$. To avoid getting stuck in a local minimum, a turbulence factor is implemented into the velocity update (see equation (1.21)), which is similar to a mutation operator in GAs. The modified update equation is given by:

$$\mathbf{v}_i^{t+1} = w(\mathbf{v}_i^t + c_1 r_1(\mathbf{x}_{pb,i} - \mathbf{x}_t^t) + c_2 r_2(\mathbf{x}_{gb,i} - \mathbf{x}_i^t)) + \exp(-\delta t) \cdot u \tag{1.23}$$

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

The nondominated solutions found during the evolutionary process are stored in an external archive denoted as A. This set of nondominated solutions is updated during the evolutionary process by selecting the best N solutions from the union of the current population P and the external archive A, according to the fuzzy dominance relation. In a previous implementation of fuzzy dominance [35], the membership functions $\mu_i^{dom}(\cdot)$ used to compute the fuzzy i-dominances were defined to be zero for negative arguments. Therefore, whenever $f_i(\mathbf{u}) > f_i(\mathbf{v})$, the degree of fuzzy dominance $\mathbf{u} \prec_i^F \mathbf{v}$ is necessarily zero. In this memetic approach, nonzero values are allowed. The membership functions used are trapezoidal, yielding nonzero values whenever their arguments are to the right of a threshold ε. Mathematically, the memberships $\mu_i^{dom}(\mathbf{u} \prec^F \mathbf{v})$ are defined as:

$$\mu_i^{dom}(\delta f_i) = \begin{cases} 0, & \delta f_i \leq -\varepsilon \\ \frac{\delta f_i}{\delta_i}, & -\varepsilon < \delta f_i < \delta_i - \varepsilon \\ 1, & \delta f_i \geq \delta_i - \varepsilon \end{cases} \tag{1.24}$$

where $\delta f_i = f_i(\mathbf{v}) - f_i(\mathbf{u})$. Given a population of solutions $P \subset \mathcal{X}$, a solution $\mathbf{v} \in P$ is said to be fuzzy dominated in P iff it is fuzzy dominated by any other solution $\mathbf{u} \in S$. In this way, each solution can be assigned a single measure to reflect the magnitude by which it dominates the others in a population. Better solutions within a set will have a lower fuzzy dominance value, although, unlike in [35] non-dominated solution may not necessarily be assigned zero values. In order to compare multiple solutions having similar fuzzy dominance values, the crowding distance of NSGA-II is used [11].

At each generation of the multi-objective PSO, NSS is executed. Considering a MOP with n decision variables, the set of solutions P is divided into separate clusters, where each cluster consists of proximally located solutions and it is generated by using a variant of the k-means algorithm [42]. The clusters are disjoint, with $n + 1$ points each one. Each cluster represents the simplex from which the local simplex search is performed. At each iteration of the local search procedure, NSS performs l movements (reflection, expansion or contraction) into the simplex before finishing its execution. The solutions found by the NSS are used to update both the population P and the archive A by using the fuzzy dominance relation. Algorithm 8 shows the general framework of the multi-objective PSO.

```
Input:
$T_{max}$: maximum number of generations;
Output:
$A$: final approximation to the Pareto optimal front;
begin
    $t = 0$;
    Generate a set of particles $P_t$ of size $N$ // using a uniform distribution;
    Initialize all velocities $\mathbf{v}_i^t$, to zero;
    while $t < t_{max}$ do
        Evaluate the set of particles $P_t$;
        Evaluate the fuzzy dominance in the population $P_t$ according to
        equation (1.24);
        Update the archive $A$;
        Update each particle $\mathbf{x}_i \in P_t$ including its personal best and global best;
        Randomly initialize $k$ cluster centers;
        Assign each particle $\mathbf{x}_i$ to a cluster using the $k$-means algorithm;
        For each cluster, apply the simplex search algorithm.;
        Update the velocities $\mathbf{v}_i^{t+1}$ according to equation (1.23);
        Update the positions $\mathbf{x}_i \in P_t$ according to equation (1.22);
        $t = t + 1$;
    end
    return $P_t$
end
```

Algorithm 8: The Multi-objective Hybrid PSO

The authors suggested the use of $k = 9$ as the number of centers for the k-means algorithm, and $l = 2$ for the number of movements (reflection, expansion or contraction) into the simplex. The simplex search was tested using $\rho = 1, \chi = 1.5$ and $\gamma = 0.5$ for the reflection, expansion and contraction, respectively. In this hybrid multi-objective PSO, the use of the shrinkage transformation is omitted. The population size was set $N = 100$ and the external archive was limited to a maximum of 100 particles. The proposed hybrid was used to solve artificial test functions and a molecular genetic model plant problem having between 3 and 10 decision variables and two objective functions. For a more detailed description of this algorithm see [36].

1.8.3 A Nonlinear Simplex Search Genetic Algorithm

Zapotecas and Coello [70] presented a hybridization between the well-known NSGA-II and Nelder and Mead's method. The proposed Nonlinear Simplex Search Genetic Algorithm (NSS-GA) combines the explorative power of NSGA-II with the exploitative power of Nelder and Mead's method, which acts as a local search engine. The general framework of the proposed MOMA is shown in Algorithm 9. This evolutionary approach evolves a population P_t by using the genetic operators of the NSGA-II (Simulated Binary Crossover (SBX) and Polynomial-Based Mutation (PBM)) and then, the local search mechanism is applied. The general idea of the local search procedure is to intensify the search towards better solutions for each objective function and the maximum bungle (sometimes called knee) of the Pareto optimal front. The main goal of the NSS is to obtain Λ, which is defined as:

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

where λ_i is a set of the best solutions found for the i-th objective function of the MOP. Υ is a set of the best solutions found by minimizing an aggregating function which approximates solutions to the knee of the Pareto optimal front.

The local search mechanism is applied at each $\frac{n}{2}$ generations, where n denotes the number of decision variables of the MOP. Initially, the local search is focused on minimizing, separately, the objective functions f_i's of the MOP. Once the separate functions are minimized, an aggregating function is used for approximating solutions to the maximum bungle of the Pareto front. The initial search point from which the local search starts is defined according to the following rules:

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

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

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

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

  $$G(\mathbf{x}_\Delta) = \sum_{i=1}^{k} \frac{|z_i - f_i(\mathbf{x}_\Delta)|}{|z_i|} \tag{1.25}$$

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

  $$g(\mathbf{x}) = ED(\mathbf{F}(\mathbf{x}), \mathbf{z}^\star) \tag{1.26}$$

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

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

1. Reducing the Search Domain: A sample of s solutions which minimize the objective function to be optimized is identified, and then, the average ($\mathbf{m}$) and standard deviation (σ) of these decision variables is computed. Based on that information, the new search space is defined as:
$$\begin{aligned}\mathbf{L}_{bound} &= \mathbf{m} - \sigma \\ \mathbf{U}_{bound} &= \mathbf{m} + \sigma\end{aligned}$$
where $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively. In this work, the authors proposed to use $s = 0.20 \times N$, where N is the population size of the evolutionary algorithm—i.e. 20% of the population size.

2. Building the Vertices: Once the new search domain has been defined, the remaining vertices are determined by using either the Halton [19] or the Hammersley [20] sequence (each has a 50% probability of being selected) in the new bounds $\mathbf{L}_{bound}$ and $\mathbf{U}_{bound}$, previously defined.

Once the simplex is defined, the NSS method is executed during a determined number of iterations defined by the following stopping criteria. The local search is stopped if: 1) it does not generate a better solution after $n+1$ iterations, or 2) if after performing $2(n+1)$ iterations, the convergence is less than ϵ. The knowledge of the local search is introduced to the population of the NSGA-II using the crowding comparison operator [11] using the union of the current population P and the set of solutions found by the local search Λ, i.e. $P \cup \Lambda$.

The authors suggested a population size of $N = 100$, and the simplex was controlled using $\rho = 1, \chi = 2$ and $\gamma = 0.5$ for the reflection, expansion and contraction coefficients, respectively. The shrinkage step is not employed in this approach. The threshold for the convergence in the simplex search was set to: $\epsilon = 1 \times 10^{-3}$. The hybrid algorithm was tested over artificial test functions having between 10 and 30 decision variables, and two and three objective functions. A more detailed description of this hybrid algorithm can be found in [70].

1.8.4 A Hybrid Non-dominated Sorting Differential Evolution Algorithm

Zhong et al. [75] hybridized Nelder and Mead's method with Differential Evolution (DE) [66] by using nondominated sorting. The proposal of Zhong et al. adopts nonlinear simplex search as its local search engine in order to obtain nondominated solutions along the evolutionary process according to the Pareto dominance relation. The sorting strategy adopted in this work, involves the evaluation of the fitness function of each solution, and the dominance relation among the individuals of the population is defined according to their fitness cost. Throughout the search, the nondominated solutions are stored in a separate set A which, at the end of the search, will constitute an approximation of the Pareto optimal set.

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

$$\mathbf{v}_i = \mathbf{x}_i^{best} + F \cdot (\mathbf{x}_{r_0} - \mathbf{x}_{r_1}) + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{1.27}$$

```
Input:
$t_{max}$: maximum number of generations;
Output:
$P$: final approximation to the Pareto front;
begin
    $t = 0$;
    Randomly initialize a population $P_t$ of size $N$;
    Evaluate the fitness of each individual in $P_t$;
    while $t < t_{max}$ do
        $Q_t = \text{MUTATION}(\text{CROSSOVER}(B))$; // Apply the genetic operators of NSGA-II
        $R_t = P_t \cup Q_t$;
        Assign to $P^*$ the $N$ best individuals from $R_t$ // According to the crowding comparison operator;
        if ($t \mod \frac{n}{2} = 0$) then
            Get $\Lambda$ set by minimizing each function of the MOP and using the aggregating function (equation (1.26)).
            $R_t^* = P_t^* \cup \Lambda$;
            Assign to $P_{t+1}$ the $N$ best individuals from $R_t^*$ // According to the crowding comparison operator;
        else
            $P_{t+1} = P^*$;
        end
        $t = t + 1$;
    end
    return $P_t$
end
```

Algorithm 9: The Nonlinear Simplex Search Genetic Algorithm

where $\mathbf{x}_{r_0}, \mathbf{x}_{r_1}, \mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$ are different solutions taken of P_t and $\mathbf{x}_i^{best}$ is a solution randomly chosen from the set of nondominated solutions A. The trial vector $\mathbf{v}_i$ is then used for generating the new solution $\mathbf{x}'_i$ by using binary crossover:

$$\mathbf{x}'_i(j) = \begin{cases} \mathbf{v}_i(j) & \text{if } r < CR \\ \mathbf{x}_i(j) & \text{otherwise} \end{cases} \tag{1.28}$$

where r is a random number having uniform distribution, $j = 1, \ldots, n$ is the j^{th} parameter of each vector and CR represents the crossover ratio. After the whole offspring population Q_t is generated, the nondominated sorting of $P_t \cup Q_t$ is used for obtaining the set of N solutions (N is the number of solutions in P_t) for the next population P_{t+1}.

In the local search mechanism, a simplex S is built by randomly selecting a nondominated solution from A, and the other n points (where n is the number of decision variables) are randomly chosen from the current population P_t. If the population P_t cannot provide enough points to create the simplex, other points are selected from A. After the simplex is built, the vertices of the simplex are stored according to the Non-dominated Sorting Differential Evolution (NSDE). The sorting strategy involves evaluating the fitness value of each solution, and the dominance relation among the solutions in the simplex is built according to their fitness cost. NSS performs any movement in the sorted simplex. The movements in the simplex are performed according to the Nelder and Mead algorithm. However, for the comparisons among the solutions, the dominance relation is used instead of a function cost. The shrinkage step is performed if either inside or outside, a contraction fails. In this case, all the vertices into the transformed simplex S are sorted to obtain the solutions which are nondominated. Considering m as the number of the current nondominated solutions in the simplex, the shrinkage step is performed according to the following description.

If $m > 1$, then there exist different converging directions, which could help to maintain the diversity of the solutions. Then, new simplexes $S_1, S_2, \ldots, S_m$ with m nondominated solutions each as respective guiding points, are generated. The new simplexes are stored within a bounded array. If the total number exceeds the storing space of the array, no more new simplexes are accepted. Then, these simplexes iterate to shrink the Pareto fronts in the array order as it is described in the following; If $m \leq 1$ or $S \in S_1, \ldots, S_m$, set the Pareto point m correspondingly in the simplex S_m as the guiding point x_1. The vertices in the simplex are relocated according to:

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

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

The Euclidean distance among the centroid and the vertices of the simplex is used for assessing the convergence at each simplex. After the convergence has taken place in all simplexes of the array, the nondominated solutions found by the simplex searches are introduced into the population of the evolutionary algorithm according to the nondominated sorting strategy used in NSDE.

The authors suggested a population size of $N = 20 \times k \times n$ where n and k represent the number of decision variables and the number of objective functions of the MOP, respectively. The DE/best/2/bin strategy was used with a crossover ratio $CR = 0.8$ and a weighting factor $F = 0.5$. The nonlinear simplex search was performed using $\rho = 1, \chi = 2, \gamma = 0.5$ and $\sigma = 0.5$ for the reflection, expansion, contraction and shrinkage movements, respectively. The Euclidean distance criterion to assess the convergence was set as 1×10^{-12}. For a more detailed description of this multi-objective evolutionary technique the interested reader can be referred to [75].

1.8.5 A Multi-Objective Memetic Evolutionary Algorithm based on Decomposition

Zapotecas and Coello [73] presented a memetic algorithm using Nelder and Mead's algorithm which was coupled to MOEA/D [74]. The local search engine is based on the Multi-objective Nonlinear Simplex Search Algorithm (MONSS) framework presented in [72]. The memetic evolutionary algorithm exploits the promising neighborhoods of the nondominated solutions found by MOEA/D. Considering P as the set of solutions found by MOEA/D in any generation, this approach assumes that if a solution $\mathbf{p} \in P$ is nondominated, then there exists another nondominated solution $\mathbf{q} \in \Omega$ such that $||\mathbf{p} - \mathbf{q}|| < \delta$ for any small $\delta \in \mathbb{R}_+$. In other words, the probability that $\mathbf{q}$ is nondominated with respect to $\mathbf{p}$ in the neighborhood defined by δ is equal to one, which implies that $\mathbf{q}$ is also nondominated. This property is considered to obtain new nondominated solutions departing from nondominated solutions located in the current population P. Since MOEA/D decomposes a MOP into several scalarization subproblems and such subproblems are solved during the evolutionary process, if all solutions in P are nondominated, then it assumes that the minimum value to each subproblem has been achieved. Considering that at the end of the evolutionary process, the population converges to a particular region of the search space (the place where the nondominated solutions are contained), the performance of the local search engine should be better when the diversity in the population is higher, i.e., when having a low number of nondominated solutions. The local search procedure is applied then, when the percentage of nondominated solutions in P is less than a certain threshold.

Considering $P^\star \subseteq P$ as the set of nondominated solutions found by MOEA/D in any generation, and assuming that all the nondominated solutions in $P^\star$ are equally efficient, the solution $\mathbf{p}_{ini}$ which starts the local search is randomly taken from $P^\star$. Solution $\mathbf{p}_{ini}$ represents not only the initial search point, but also the simplex head from which the simplex is built. Let $\mathbf{w}_{ini}$ be the weighting vector that defines the subproblem for which the initial search solution $\mathbf{p}_{ini}$ is minimum. Let $S(\mathbf{w}_{ini})$ be the neighborhood of the n closest weighting vectors to $\mathbf{w}_{ini}$ (where n is the number of decision variables of the MOP). Since the dimensionality of the simplex depends of the number of decision variables of the MOP, the population size of the MOEA needs to be larger than the number of decision variables. Then, the simplex defined as:

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

is built in two different ways by using a probability $P_s = 0.3$, according to the two following strategies:

i. *Neighboring solutions:* The remaining n solutions $\mathbf{p}_i \in P$ $(i = 1, \dots, n)$ are chosen, such that, $\mathbf{p}_i$ minimizes each subproblem defined by each weighting vector in $S(\mathbf{w}_{ini})$. This is the same strategy employed for constructing the simplex used in MONSS [72].

ii. *Sample solutions:* The remaining n solutions $\mathbf{p}_i \in \Omega$ $(i = 1, \dots, n)$ are generated by using a low-discrepancy sequence. The Hammersley sequence [20] is adopted to generate a well-distributed sampling of solutions in a determined search space. In an analogous way as in [70], this approach uses a strategy based on the genetic analysis of a sample from the current population for reducing the search space. However, here, the average $(\mathbf{m})$ and standard deviation (σ) of the solutions that minimize each subproblem defined by the weighting vectors in $S(\mathbf{w}_{ini})$ are computed. In this way, the new bounds are defined by:

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

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

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

Let $B(\mathbf{w}_{ini})$ be the neighborhood of the T closest weighting vectors to $\mathbf{w}_{ini}$, such that $\mathbf{w}_{ini}$ defines the subproblem for which the initial search solution $\mathbf{p}_{ini}$ is minimum. Let $D(\mathbf{w}_{ini})$ be the $A_r = 5$ closest weighting vectors to $\mathbf{w}_{ini}$. NSS focuses on minimizing a subproblem defined by the weighting vector $\mathbf{w}_{obj}$, which is defined according to the following rules:

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

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

At each iteration of the local search, the vertices of the simplex $\mathbf{\Delta}$ are sorted according to their value for the subproblem that it tries to minimize. In this way, a movement into the simplex is performed for generating the new solution $\mathbf{p}_{new}$ by using Nelder and Mead's method. The new solution generated by the NSS replaces one or more solutions of the population according to: Let $B(\mathbf{w}_{obj})$ and $W = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$ be the neighborhood of the T closest weighting vectors to $\mathbf{w}_{obj}$, and the well-distributed set of all weighting vectors, respectively. The set of weighting vectors is defined by:

$$Q = \begin{cases} B(\mathbf{w}_{obj}), & \text{if } r < 0.9 \\ W & \text{otherwise} \end{cases}$$

where r is a random number having uniform distribution.

The population P is updated by replacing at most R_{ls} solutions from P such that, $g(\mathbf{p}_{new}|\mathbf{w}_i, z) < g(\mathbf{x}_i|\mathbf{w}_i, z)$, where $\mathbf{w}_i \in Q$ and $\mathbf{x}_i \in P$ is the solution that $\mathbf{x}_i$ minimizes the subproblem defined by $\mathbf{w}_i$. The local search is performed by a maximum number of fitness function evaluations E_{ls}. If the nonlinear simplex search overcomes this maximum number of evaluations, the simplex search is stopped and the evolutionary process of MOEA/D continues. The search could be inefficient if the simplex has been deformed so that it has collapsed into a region where there are no local minima. Therefore, if the NSS does not find a minimum value in $n + 1$ iterations, a reinitialization of the simplex is performed.

The authors used $\rho = 1, \chi = 2$ and $\gamma = 0.5$ for the reflection, expansion and contraction movements of the NSS. The population size used was set as $N = 100$ for two-objective problems and 300 for three-objective problems. The action range for the NSS was set as $A_r = 5$. The number of solutions to be replaced was set as $R_{ls} = 15$. The total number of iterations of the NSS was set as $E_{ls} = 300$. The proposed memetic algorithm was tested using artificial functions between 10 and 30 decision variables, with two and three objectives.

1.9 Other Direct Search Approaches

In the last few years, several algorithms that hybridize Nelder and Mead's algorithm with a MOEA have been presented. Such implementations have used solutions from the population of the MOEA as the vertices to define the simplex. Since the solutions in the population are evaluated, the vertices of the simplex do not need to be evaluated. This reduces the number of function evaluations that a MOMA based on the NSS could require. Nevertheless, in the specialized literature there are other direct search methods that do not require a simplex in order to obtain an optimal point of a function, but their use had been somewhat scarce in MOMAs during several years. Most of these methods start the search from a single point (which implies less information of the fitness landscape), and generate a solution for each iteration, whereupon, the convergence to an optimal value could require several function evaluations and the hybridization with a MOEA could be inefficient. However, in recent years, the coupling of this type of direct search methods with a MOMA has attracted a lot of interest from some researchers. In the following sections we present some strategies that have been used for coupling other direct search methods (different to Nelder and Mead's method) with a MOEA.

1.9.1 A Multi-Objective Meta-model Assisted Memetic Algorithm

Zapotecas and Coello [71] presented a strategy which combines a MOEA with a direct search method assisted by a surrogate model. The local search mechanism adopts the Hooke and Jeeves algorithm [24] as its local search engine which is assisted by a surrogate model based on Support Vector Regresion (SVR).

```
Input:
a stopping criterion;
$N$: The number of the subproblems considered in the memetic algorithm;
$W$: A well-distributed set of weighting vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$;
$T$: The number of weight vectors in the neighborhood of each weighting vector;
$R_{ls}$: The maximum number of evaluations for the local search;
$A_{ls}$: The action range for the local search.
Output:
$P$: the final approximation to the Pareto optimal front found by the memetic algorithm.
begin
    Compute the Euclidean distances between any two weighting vectors and then work out the $T$ closest weighting vectors to each weighting vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$ where $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_T}$ are the $T$ closest weighting vectors to $\mathbf{w}_i$;
    Generate an initial population $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ randomly. Set $FV^i = \mathbf{F}(\mathbf{x}_i)$; $\mathbf{z} = (+\infty, \ldots, +\infty)^T$;
    while the stopping criteria is not satisfied do
        Perform an iteration of MOEA/D (see Algorithm 3);
        if the percentage of nondominated solutions in $P$ is less than 50% then
            while there are enough resources and the simplex has not collapsed do
                Select a solution from $P$ as the initial search solution ($\mathbf{p}_{ini}$);
                Build the simplex;
                Select the search direction for the NSS;
                Perform an iteration of NSS for obtaining $\mathbf{p}_{new}$;
                Update the population $P$ using the new solution $\mathbf{p}_{new}$;
            end
        end
    end
end
```

Algorithm 10: The Multi-Objective Memetic Evolutionary Algorithm based on Decomposition

Multi-Objective Meta-model Assisted Memetic Algorithm (MOMAMA) generates a sample S of size $2N$ (where N is the population size) which is randomly distributed in the search space using the Latin hypercube sampling method [43]. The surrogate model is trained using the set of solutions $D = S$ which is evaluated with the real fitness function. The initial population P_0 is defined by N solutions randomly chosen from S. The nondominated solutions found throughout the search are stored in an external archive A. The external archive is always maintained with at most N nondominated solutions, and it is pruned by performing Pareto ranking [11]. A technique based on clustering is used for selecting the best nondominated solutions when the external archive contains more than N solutions.

At each iteration t, recombination takes place between each individual $\mathbf{x}_i \in P_t$ and an individual which can be chosen from either R_t or A, according to the next rule.

$$\begin{aligned} parent^1 &= \mathbf{x}_i \in P_t \quad \forall i = 1, \ldots, N \\ parent^2 &= \begin{cases} \mathbf{y} \in R_t, & \textbf{if } \left(g < 1 - \frac{|A|}{2N}\right) \\ \mathbf{y} \in A, & \textbf{otherwise} \end{cases} \end{aligned} \tag{1.29}$$

where g is a uniformly distributed random number within $(0, 1)$ and $\mathbf{y}$ is a solution randomly chosen from A or R_t. After the parents are recombined, the mutation operator is applied to each child for obtaining the offspring population Q_t. The next population P_{t+1} is then stated by selecting N individuals from $P_t \cup Q_t$ according to Pareto ranking [11].

The local search mechanism uses the surrogate model in order to find new solutions nearby the solutions provided by the MOEA. The local search is guided by a set W of n_w weighting vectors which define a scalar optimization problem using the following augmented weighting Tchebycheff problem.

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{i=1,\ldots,k} \{w_i |f_i(\mathbf{x}) - z_i^\star|\} + \alpha \sum_{i=1}^{k} |f_i(\mathbf{x}) - z_i^\star| \tag{1.30}$$

where α is a sufficiently small positive scalar and $\mathbf{z}^\star$ represents the utopian vector.

The Hooke and Jeeves algorithm approximates solutions to the Pareto optimal set by solving the n_w Tchebycheff problems. For each weighting vector $\mathbf{w}_j \in W$, a set of solutions λ_j which consists of all solutions evaluated into the meta-model is found. The initial search point $\mathbf{x}_s^1$ for solving the problem corresponding to the weighting vector $\mathbf{w}_1$, is stated as the $\mathbf{x}^\star \in P_t \cup A$ that minimizes equation (1.30). The remaining initial search points $\mathbf{x}_s^j$ $(j = 2, \ldots, n_w)$ are defined by the local optimal solution found by the Hooke and Jeeves algorithm for the weighting vector $\mathbf{w}_{j-1} \in W$.

Let $\Lambda = \bigcup_{j=1}^{n_w} \lambda_j$ be the set of solutions found so far by the local search mechanisms. Considering that the probability that, given a nondominated solution $\mathbf{q}^*$, there exists another nondominated solution $\mathbf{p}$ in its neighborhood is equal to one, i.e.:

$$P(\exists \mathbf{p} \in \mathbb{R}^n : ||\mathbf{q}^* - \mathbf{p}|| < \delta \text{ and } \mathbf{q}^* \not\prec \mathbf{p}) = 1 \tag{1.31}$$

for any small $\delta \in \mathbb{R}_+$, the proposed approach generates more approximate solutions by using DE [66].

The initial population is given by $\mathcal{G}_0 = \Lambda$. Each new individual is stored in an external archive $\mathcal{L}$ according to the dominance rule. The archiving strategy can make that the set of solutions $\mathcal{L}$ increases or decreases its size. The next population for the DE algorithm is then defined by $\mathcal{G}_{g+1} = \mathcal{L}$.

Since all the solutions in the archive $\mathcal{L}$ are nondominated, we can say that DE has converged (at least locally) when it has obtained N different nondominated solutions from the evolutionary process. That is:

$$\textbf{if } |\mathcal{L}| = N \textbf{ then} \qquad \text{stop the DE algorithm} \tag{1.32}$$

The solutions set R_t obtained by the local search mechanism is given by $R_t = \mathcal{L}$. However, this stopping criteria is not always satisfied. Thus, R_t set can be defined by selecting N individuals from $\Lambda \cup \mathcal{L}$ using Pareto ranking [11] after a certain number of iterations. The final set R_t is then evaluated using the real fitness function and it is used for generating the offspring population Q_t of the memetic algorithm. After applying the local search, the surrogate model is retrained by using the data set D with the set of local optimal solutions Q_t, i.e., the training set is defined by $D = D \cup Q_t$. Algorithm 11 shows the general scheme of the proposed MOMAMA. A more detailed description of this evolutionary approach can be found in [71].

```
Input:
$T_{max}$: Maximum number of generations;
Output:
$A$: Final approximation to the Pareto optimal front;
begin
    $A = \emptyset$;
    Generate $S$ of size $2N$ // using the Latin Hyper-cubes method;
    EVALUATE($S$) // using the real fitness function;
    $P = \{x_i \in S\}$ such that: $x_i$ is randomly chosen from $S$ and $|P_t| = N$;
    $R_t = S \setminus P_t$;
    $A =$UPDATEARCHIVE($R_t, A$);
    $D = S$;
    while $t < T_{max}$ do
        $A =$UPDATEARCHIVE($P_t, A$);
        $Q_t =$CREATEOFFSPRING($P_t, R_t, A_t$); // apply genetic operators
        EVALUATE($Q_t$); // using the real fitness function
        $D = D \cup Q_t$;
        Retrain the meta model using the training set $D$;
        $P_{t+1} =$SELECTNEXTPOPULATION($P_t, Q_t$); // using Pareto ranking
        $R_{t+1} =$SURROGATELOCALSEARCH($P_t, A$); // using the Hooke and Jeeves
        algorithm, and DE into the surrogate model
        $t = t + 1$;
    end
end
```

Algorithm 11: The Multi-Objective Meta-model Assisted Memetic Algorithm

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

Koch et al. [34] presented a hybrid algorithm which combines the exploratory properties of the S-Metric Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) [1] with the exploitative power of the Hooke and Jeeves algorithm [24] which is used in the local search engine. SMS-EMOA optimizes a set of solutions according to the S-metric or Hypervolume indicator [78], which measures the size of the space dominated by the population. This performance measure is integrated into the selection operator of SMS-EMOA which aims for maximization of the S-metric and thereby guides the population towards the Pareto optimal front. A $(\mu + 1)$ (or steady-state) selection scheme is applied: At each generation, SMS-EMOA discards the individual that contributes least to the S-metric value of the population. The invoked variation operators are not specific for the SMS-EMOA but are taken from the literature, namely PBM and SBX, with the same parametrization as in the NSGA-II [11]. At each iteration, the Hooke and Jeeves algorithm performs an exploratory move along the coordinate axes. Afterwards, the vectors of the last exploratory moves are combined to a projected direction that can accelerate the descent of the search vector. When the exploratory moves lead to no improvement in any coordinate direction, step sizes are reduced by a factor η. The search terminates after a number of predefined function evaluations or, alternatively, when the step size falls below a constant value $\varepsilon > 0$.

The Hooke and Jeeves was conceived for minimizing SOPs, therefore, its use to deal with MOPs is not possible without modifications. Koch et al. adopted a scalar function by using the weighting sum approach developed in [12]. Besides, the proposed MOMA introduces a probability function $p_{ls}(t)$ for extending the idea presented in Sindhya et al. [63] who linearly oscillated the probability for starting the local search procedure. The probability function adopted in this work is given by:

$$p_{ls}(t) = \frac{p_{\max} \cdot \Phi(t \mod (\alpha\mu))}{\Phi(\alpha\mu - 1)} \tag{1.33}$$

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

```
Input:
$T_{max}$: Maximum number of generations;
Output:
$A$: Final approximation to the Pareto optimal front;
begin
    $t = 0$;
    Generate a population $P_t$ of size $N$; // using uniform distribution
    Evaluate the population $P_t$;
    while $t < T_{max}$ do
        Select $\mu$ parents of $P_t$;
        Create population $Q_t$ with $\lambda$ offspring;
        for $i$=1 to $\lambda$ do
            Choose random variable $r \in [0, 1]$;
            if $r \leq p_{ls}(t)$ then
                Local search for $Q_t[i]$;
            end
        end
        Evaluate $\lambda$ offspring;
        Create population $P_{t+1}$ out of $P_t$ and $Q_t$;
        $t = t + 1$;
    end
end
```

Algorithm 12: The hybrid SMS-EMOA

Using the same outlined Algorithm 12, Koch et al. hybridized SMS-EMOA with other mathematical programming techniques. The multi-objective Newton method [14] and the step descent method [15] are also hybridized with the SMS-EMOA. Koch et al. emphasize thed importance of the probability function p_{ls} that controls the frequency of the local search during the optimization process. Three different functions using equation (1.33) and a constant probability p_{ls} were adopted. The hybrid variants using equation (1.33) obtained a value of $\alpha = 0.5$ as proposed by Sindhya et al. [63] and the next functions were used.

1. $p_{ls}(t)$ with $\Phi(x) = x$ (in equation (1.33))
2. $p_{ls}(t)$ with $\Phi(x) = x^2$ (in equation (1.33))
3. $p_{ls}(t)$ with $\Phi(x) = \log(x)$ (in equation (1.33))

4. $p_{ls}(t)$ with $\Phi(x) = 0.01$

Each hybridization with the above probability functions was tested on the ZDT test suite- The hybrid SMS-EMOA is started with a population size of $N = 100$, the SBX recombination operator proposed by [63] was adopted with a probability 1.0, and the polynomial mutation operator was adopted with probability 1.0. According to the results reported, the hybrid using the multi-objective Newton method achieved better results than those obtained by both the hybrid SMS-EMOA using Hooke and Jeeves algorithm, and using the step descent method. More details of this hybridization can be found in [34].

1.10 Conclusions

This chapter has provided a detailed description of the hybridization of MOEAs with mathematical programming methods. The main motivation for this sort of coupling is, evidently, to improve the performance that any of these two types of techniques can provide, when considered separately. Our discussion has included the use of both gradient-based and non-gradient-based methods for the local search engine. In order to make the chapter self-contained, the most important required background has also been included.

The development of hybrid MOEAs raises several challenges which include the design of efficient and effective local search engines (particularly in continuous search spaces), the proper balance between the global and the local search engines, and the appropriate choice of the local search engine to be adopted, among others [7].

However, the review of approaches provided in this chapter clearly illustrates the interest that hybrid MOEAs have attracted in the last few years. Such interest is expected to raise in the next few years, since this is, indeed, a very promising research line within evolutionary multi-objective optimization due to the potential benefits that these hybrid MOEAs could bring (particularly, when dealing with complex real-world problems).

Acknowledgements

The last author gratefully ackowledges support from CONACyT project no. 103570.

Acronyms

SOP	Single-objective Optimization Problem
EC	Evolutionary Computation
KKT	Karush-Kuhn-Tucker
MOEA	Multi-Objective Evolutionary Algorithm
MOP	Multi-objective Optimization Problem
EA	Evolutionary Algorithm
LS	Local Search
GA	Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
SPEA	Strength Pareto Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
SPEA2	Strength Pareto Evolutionary Algorithm 2
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOMA	Multi-Objective Memetic Algorithm
CDMOMA	Cross Dominant Multi-Objective Memetic Algorithm
PMA	Pareto Memetic Algorithm
MOGLS	Multi-Objective Genetic Local Search
M-PAES	Memetic Pareto Archived Evolution Strategy
PDMOSA	Pareto Domination Multi-Objective Simulated Annealing
CMODE	Coevolutionary Multi-Objective Differential Evolution
MDD	Multi-objective Descent Direction
CORL	Combined-objectives Repeated Line Search
EDA	Estimation Distribution Algorithm (EDA)
PDM	Pareto Descent Method
ZDT	Zitzler-Deb-Thiele

DTLZ	Deb-Thiele-Laumanns-Zitzler
SQP	Sequential Quadratic Programming
DE	Differential Evolution
SVR	Support Vector Regresion
MOMAMA	Multi-Objective Meta-model Assisted Memetic Algorithm
SBX	Simulated Binary Crossover
PBM	Polynomial-Based Mutation
PSO	Particle Swarm Optimization
MA	Memetic Algorithm
NSS	Nonlinear Simplex Search
NSS-GA	Nonlinear Simplex Search Genetic Algorithm
NSDE	Non-dominated Sorting Differential Evolution
MONSS	Multi-objective Nonlinear Simplex Search Algorithm
SMS-EMOA	S-Metric Selection Evolutionary Multi-objective Optimization Algorithm
EMOA	Evolutionary Multi-objective Optimization Algorithm

Bibliography

[1] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[2] 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)*, volume 1, pages 755–762, New York, USA, June 2005. ACM Press.

[3] 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)*, volume 1, pages 627–634, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[4] 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.

[5] Martin Brown and R.E. Smith. Effective use of directional information in multi-objective evolutionary computation. In *Genetic and Evolutionary Computation (GECCO) 2003*, volume Volume 2723/2003 of *Lecture Notes in Computer Science*, pages 778–789. Springer Berlin / Heidelberg, 2003.

[6] 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.

[7] 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.

[8] 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.

[9] I. Das and J. E. Dennis. Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[10] Richard Dawkins. *The Selfish Gene.* Oxford University Press, 1990.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions Evolutionary Computation*, 6(2):182–197, 2002.

[12] Kalyanmoy Deb and Tushar Goel. A hybrid multi-objective evolutionary approach to engineering shape design. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, EMO '01, pages 385–399, London, UK, UK, 2001. Springer-Verlag.

[13] Matthias Ehrgott. *Multicriteria Optimization.* Springer, Berlin, 2nd edition edition, June 2005.

[14] J. Fliege, L. M. Graña Drummond, and B. F. Svaiter. Newton's method for multiobjective optimization. *SIAM J. on Optimization*, 20(2):602–626, May 2009.

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

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

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

[18] 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.

[19] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, December 1960.

[20] J. M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86:844–874, 1960.

[21] 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)*, volume 1, pages 813–820, London, UK, July 2007. ACM Press.

[22] 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.

[23] Claus Hillermeier. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach.* Birkhäuser Basel, 2000.

[24] Robert Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, 1961.

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

[26] 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.

[27] 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.

[28] 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.

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

[30] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[31] J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[32] 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.

[33] Joshua D. Knowles and David W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., Julio 1999. IEEE Service Center.

[34] Patrick Koch, Oliver Kramer, Günter Rudolph, and Nicola Beume. On the hybridization of sms-emoa and local search for continuous multiobjective optimization. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 603–610, New York, NY, USA, 2009. ACM.

[35] Praveen Koduru, Sanjoy Das, Stephen Welch, and Judith L. Roe. Fuzzy Dominance Based Multi-objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 356–367, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[36] Praveen Koduru, Sanjoy Das, and Stephen M. Welch. Multi-Objective Hybrid PSO Using ϵ-Fuzzy Dominance. In Dirk Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 853–860, London, UK, July 2007. ACM Press.

[37] 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.

[38] 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.

[39] 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.

[40] Adriana Lara, Oliver Schütze, and Carlos A. Coello Coello. *EVOLVE: A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, volume 447 of *Studies in Computational Intelligence*, chapter On Gradient-based Local Search to Hybridize Multi-objective Evolutionary Algorithms, page 440. Springer.

[41] 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.

[42] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[43] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[44] J.M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345 –377, mar 1995.

[45] Kaisa Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Boston, Massachuisetts, 1999.

[46] Kaisa M Miettinen. Some methods for nonlinear multi-objective optimization. In E. et al Zitzler, editor, *Evolutionary Multi-Criterion Optimization*, volume 1993/2001 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin, Heidelberg, 2001.

[47] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[48] H. Mukai. Algorithms for multicriterion optimization. *Automatic Control, IEEE Transactions on*, 25(2):177–186, 1980.

[49] 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.

[50] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7:308–313, 1965.

[51] 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.

[52] Singiresu S. Rao. *Engineering Optimization.* John Wiley & Sons Inc., 3rd edition, 1996.

[53] A. Ravindran, K.M. Ragsdell, and G.V. Reklaitis. *Engineering Optimization. Methods and Applications.* John Wiley & Sons, Inc., Hoboken, New Jersey, USA, 2006.

[54] 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.

[55] R. S. Rosenberg. *Simulation of genetic populations with biochemical properties.* PhD thesis, University of Michigan, Ann Harbor, Michigan, EE. UU., 1967.

[56] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[57] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms.* PhD thesis, Vanderbilt University, 1984.

[58] 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.

[59] O. Schütze. *Set Oriented Methods for Global Optimization.* PhD thesis, University of Paderborn, 2004. <http://ubdata.uni-paderborn.de/ediss/17/2004/schuetze/>.

[60] 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.

[61] 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.

[62] Pradyumn Kumar 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.

[63] 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.

[64] 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.

[65] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[66] Rainer M. Storn and Kenneth V. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, Berkeley, CA, March 1995.

[67] B. Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering*, 28:1849–1871, 2004.

[68] G. Timmel. Ein stochastisches suchverfahren zur bestimmung der optimalen kompromisslösung bei statistischen polykriteriellen optimierungsaufgaben. Technical report, TH Illmenau, 1980.

[69] 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.

[70] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Proposal to Hybridize Multi-Objective Evolutionary Algorithms with Non-Gradient Mathematical Programming Techniques. In *Parallel Problem Solving from Nature–PPSN X*, volume 5199, pages 837–846. Springer. Lecture Notes in Computer Science, September 2008.

[71] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature–PPSN XI*, volume 6238, pages 576–585, Kraków, Poland, September 2010. Springer, Lecture Notes in Computer Science.

[72] Saúl Zapotecas Martínez and Carlos A. Coello Coello. MONSS: A Multi-Objective Nonlinear Simplex Search Algorithm. Technical Report TR:2011-09-09, Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV-IPN), México, D.F., MÉXICO, September 2011.

[73] Saúl Zapotecas Martínez and Carlos A. Coello Coello. A Direct Local Search Mechanism for Decomposition-based Multi-Objective Evolutionary Algorithms. In *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, pages 3431–3438, Brisbane, Australia, June 2012. IEEE Press.

[74] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.

[75] Xiang Zhong, Wenhui Fan, Jinbiao Lin, and Zuozhi Zhao. Hybrid non-dominated sorting differential evolutionary algorithm with nelder-mead. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 1, pages 306 –311, December 2010.

[76] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[77] 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, 2002.

[78] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.

[79] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Noviembre 1999.