Centro de Investigacion y de Estudios
Avanzados
del Instituto Politecnico Nacional

Unidad Zacatenco

Departamento de Computación

El punto a punto de las técnicas de búsqueda local para algoritmos de optimización multiobjetivo

Tesis que presenta

Sergio Jesús Alvarado García

para obtener el Grado de

Maestro en Ciencias

en Computación

Director de la Tesis: Dr. Oliver Steffen Schütze

México, Distrito Federal **Diciembre 2012**

Centro de Investigacion y de Estudios
Avanzados
del Instituto Politecnico Nacional

Zacatenco Campus

Computer Science Department

On Pointwise Iterative Local Search Techniques for Evolutionary Multiobjective Optimization

Submitted by

Sergio Jesús Alvarado García

as the fulfillment of the requirement for the degree of

Master in

Computer Science

Advisor: Dr. Oliver Steffen Schütze

México, Distrito Federal **December 2012**

Abstract

In the real world one is often faced with problems that have multiple objectives that must be optimized concurrently leading to the so-called *multiobjective optimization problems* (MOPs). The main difficulty of these problems is that they do not have only one single solution, but instead consist of a set of solutions presented as different trade-offs among the objectives. One possible and widely accepted way to solve MOPs numerically is to use memetic algorithms, i.e., evolutionary algorithms coupled with local search mechanisms. The goal of this thesis is to extend a recently developed local search strategy, the Directed Search (DS) method. This new method, should be first developed to work as a standalone algorithm, and will further on be integrated into state-of-the-art multi-objective evolutionary algorithms leading to a new memetic strategy. One of the main ideas of this work is to use DS without explicitly computing the objective's gradient. Instead, the neighborhood will be explored leading to a new finite difference approach. The proposed algorithm has shown to increases the convergence rate of the multiobjective evolutionary algorithms.

Resumen

En el mundo real frecuentemente enfrentamos problemas que poseen múltiples objetivos, los cuales deben ser optimizados de manera concurrente llevándonos a los denominados *problemas de optimización multiobjetivo*(POMs). La dificultad de este tipo de problemas radica en que generalmente no se tiene una única solución posible, sino que por el contrario, existen un conjunto de soluciones que optimizan de diferente manera a cada uno de los objetivos. Una de las posibles y ampliamente aceptadas formas aceptadas de resolver POMs es el uso de lo que se conoce como algoritmos meméticos, v.g., algoritmos evolutivos a los que se les acoplan mecanismos de búsqueda local. El objetivo de esta tesis es el de extender un método de búsqueda local recientemente desarrollado: la búsqueda dirigida, o *Directed Search* (DS). Este nuevo algoritmo, inicialmente se utilizará como un método capaz de encontrar soluciones de un PMO por sí mismo, y posteriormente será utilizado en un algoritmo memético adoptando alguno de los algoritmos evolutivos multiobjetivo que se encuentran en el estado del arte a fin de producir un nuevo algoritmo memético. Una de las ideas fundamentales de este trabajo es el de utilizar el DS sin necesidad de calcular explicitamente el gradiente de las funciones objetivo. En vez de eso, se explorará el vecindario, a fin de producir un nuevo método de diferencias finitas. Como resultado, el algoritmo generado ha demostrado mejorar la convergencia de los algoritmos evolutivos multiobjetivo.

Acknowledgment

First I want to thank to CONACyT for the economic support provided in order to apply conclude the master's degree. Also I want to thank the CINVESTAV for the opportunity of became one of their students.

I really want to thank my advisor Dr. Oliver Schütze for all their support in this thesis work and guidance through the elaboration of this project, I know that it was not easy but I really appreciate all the patience and effort in order to help me to prepare this work.

I also like to thank Dr. Luis Gerardo de la Fraga and Dr. Carlos A. Coello Coello to read support me and support me in the construction in the final version of this thesis.

Finally I would like to thank to my family in special to my mother, because without all your sacrifices this part of my life could not exists. I also thank to my sister and my father for all the support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Some engineering problems in the real world demand to optimize several objectives at the same time, leading to so-called multiobjective optimization problems (MOPs). As an example in the design of a car one is (among many other objectives) interested in a high comfort for the passengers and at the same time in a low production cost of the vehicle, leading in this case to a bi-objective problem.

One of the main problems of the treatment of the MOPs is that there exist more than one solution that can be considered to be optimal. The set of solutions of a MOP, the so-called Pareto set (PS), typically forms a $(k-1)$-dimensional object, where k is the number of objectives involved in the MOP. The knowledge of the entire Pareto set is of particular interest for the decision maker of the underlying application since it gives him/her all trade off solutions and is hence important for the decision making process.

Nowadays, there exist several methods for the numerical treatment of a MOP. The oldest ones are known as *scalarization methods* [Ehrgott, 2005], which transform the MOP into a problem with one single objective (SOP). If the entire Pareto set is of interest, one can e.g. get an approximation by solving a sequence of SOPs. Next, there are population-based evolutionary strategies; widely used algorithms that are able to find an entire approximation of the PS in one run. The main problem of these methods resides in their slow convergence and their restriction to moderate dimensions (dimension of the parameter space $n < 50$). A possible improvement of these stochastic strategies are the *memetic algorithms* which combine the evolutionary methods along with local search techniques (mainly coming from mathematical programming), in order to increment the convergence rate of the resulting algorithm.

Problem

The scope of this work is to extend a recently developed local search technique, the *Directed Search* (DS) method presented by [Schütze et al., 2010]. The DS can be used as a standalone algorithm in order to perform a line search movement until one solution in the PS is reached. The main limitation of this method is that it requires the gradient information or a numerical approximation of it. The uses of a state-of-the-art algorithm to calculate the approximation of the gradient can lead us to a high computational effort, e.g. the approximation obtained by the finite difference method costs at least nk function evaluations, where n refers to the number of parameters and k is the number of objectives. Hence, a reduction in the number of function evaluations is needed. Next, using the DS inside an EA opens the possibility that the information of the individuals of the population can be used in order to obtain an approximation of the gradient.

Objectives

The objectives of this work are as follows:

- To design a numerical method based on the DS for the treatment of MOPs as a standalone algorithm that does not require explicitly gradient information.
- To integrate the developed algorithm into a multiobjective evolutionary algorithm (MOEA).
- To introduce the possible use of the DS into constrained MOPs.

Organization of the thesis

The organization of this thesis is as follows:

In Chapter 2, some concepts needed to understand the main ideas of this work are presented, in addition to some of the classical methods used to solve MOPs. Also some of the state-of-the-art MOEAs are given, and the concept of a *memetic algorithm* is introduced.

Chapter 3 presents the formulation of the new algorithms developed in this thesis work such as the gradient-free Directed Search method. Furthermore, results from applied the standalone algorithm in some MOP are presented.

In Chapter 4, the incorporation of the *Discrete Directed Search* (DDS) as a local search technique into two state-of-the-art methods is presented. Next, a new idea to decompose the direction d used in the DS algorithm is proposed, such that the

convergence rate of the DDS method is incremented.

Chapter 5 presents the results obtained by using the DS algorithm in order to solve some particular multiobjective control problems. First, the DS is used to correct back systems that deteriorate in time; next, novel ideas to work with constrained multiobjective control problems are presented.

Chapter 6 contains the conclusions and some possible future ideas to be developed from this work.

Chapter 2

Background

2.1 Optimization

Optimization is the process which tries to minimize or maximize the *objectives* of a given problem, e.g., to maximize the profits of a manufacturing process by changing the materials used in the process. But there is not a unique algorithm that can solve all kinds of optimization problems. In general, it depends on the type of the problem, and commonly it is left to the users to select the appropriate algorithm.

2.1.1 Notations

Decision variables

The *decision variables* (also called *parameters*) in terms of optimization refer to quantities that can control the optimization problem in order to obtain some kind of results. For instance, in some chemical processes, a change in the temperature of a boil can lead us to an increment in the pressure.

Formally the *decision variables* are represented as column vectors constructed with the n variables:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}.$$

Objective function

An *objective* is the term to name those quantitative measures used in the performance of the system under study. The objective function describes the co-relation existing between the results of the problem and the parameters. In mathematical notation,

the objective function is defined as:

$$f : G \subset \mathbb{R}^n \to \mathbb{R} \tag{2.1}$$

Constraints

Sometimes the problem requires that some conditions do not take certain values or that they must stay between some levels, e.g. the profits of a project must stay within a certain amount of cash. This type of restrictions are called the constraints of the problem. In optimization, there exist two types of constraints: equality and inequality constraints.

In mathematical notation, the constraints can be expressed as:

$$\begin{aligned} h(x) &= 0, \quad i = 1, \ldots, l \\ g(x) &\leq 0, \quad j = 1, \ldots, m \end{aligned} \quad . \tag{2.2}$$

Gradient vector

As a non-formal definition, the gradient typically refers to the derivative of a a vectorial function. Sometimes the gradient can be used to represent the slope of a curve. In mathematical terms the gradient is defined using the nabla operator ∇. Each of the i entries is formed for the n partial derivatives of the f function:

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right]^T . \tag{2.3}$$

Jacobian matrix

Having a vector of objective functions such that $F : G \to \mathbb{R}^k$, the Jacobian (denoted as $J(x)$ or $D(x)$) can be constructed using a k-row vector, where each row represents the gradient of the f_k function:

$$J(x) = \begin{bmatrix} \nabla f_1(x)^T \\ \nabla f_2(x)^T \\ \cdots \\ \nabla f_k(x)^T \end{bmatrix} \in \mathbb{R}^{k \times n}, \tag{2.4}$$

formulated with its partial derivatives the Jacobian matrix can also be described as:

$$J(x) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \cdots & & & \\ \frac{\partial f_k(x)}{\partial x_1} & \frac{\partial f_k(x)}{\partial x_2} & \cdots & \frac{\partial f_k(x)}{\partial x_n} \end{bmatrix},$$

Hessian matrix

The Hessian is the name given to the matrix formed with then second-order partial derivatives, and is commonly is used to describe the local curvature of a function. Having a vectorial function $F : G \to \mathbb{R}$, the Hessian matrix is related to the Jacobian matrix as:

$$H(x) = J(\nabla f(x))$$

Local and global minimizers

Having a point $x^* \in G$ and an objective function $F : G \to \mathbb{R}$, x^* is a *global minimizer* of F if:

$$f(x^*) \leq f(x), \forall x \in G, \tag{2.5}$$

where G is the feasible region and $G \subset \mathbb{R}^n$. It is said that x^* is a local minimizer if:

$$f(x^*) \leq f(x), \forall x \in N, \tag{2.6}$$

where N is a neighborhood of x^*.

2.2 Multiobjective optimization

In the real world it is possible to find problems that require to optimize a certain number of objectives at the same time. The main issue in this kind of optimization problems resides in trying to find a solution that optimizes all the objectives at the same time. In other words, it is possible that an optimal solution in one of the objectives could be the worst possible solution for another one. Next, the existence of multiple objectives could lead us to find not only one solution, instead, it is possible to find a set of optimal solutions. As an example in the design of a car one is (among many other objectives) interested in a high comfort for the passengers and at the same time in a low production cost of the vehicle. This leads in this case to the bi-objective problem:

$$\begin{pmatrix} \text{maximize} & \text{comfort} \\ \text{minimize} & \text{cost} \end{pmatrix}.$$

2.2.1 Formal definition of a MOP

Instead of taking a single objective, one is given k different objective functions that require to be optimized at the same time, the problem is converted into a multiobjective optimization problem (MOP). For the unconstrained case, a MOP can be defined as:

Definition 1. *Given a vector x that exist in a region G where $G \subset \mathbb{R}^n$ (n is the parameter number), and $F(x)$ a vector of objective functions, such that*

$$F : G \to \mathbb{R}^k, \qquad F(x) = (f_1(x), f_2(x), ..., f_k(x)) \ ,$$

where k is the number of objective functions.

Then we can define a MOP as:

$$\min_{x \in G} F(x). \tag{MOP}$$

Here a particular case of a MOP is when $k = 1$. This case is known as a single objective optimization problem (SOP), where G can be expressed as:

$$G = \{x \in \mathbb{R}^n | h_i(x) = 0, i = 1 \ldots m, g_i(x) \leq 0, i = 1 \ldots l\}. \tag{2.7}$$

2.2.2 Pareto dominance

In a MOP the concept of an optimal solution turns out to be ambigous, so in order to explain which solutions are better than others, the concept of optimality for a MOP was proposed by [Edgeworth, 1881] and later generalized by [Pareto, 1896], proposing the notion of *Pareto dominance* (some authors call it *Edgeworth-Pareto Optimality*).

To compare two solutions of a MOP, the dominance relation described by Pareto is defined as follows:

Definition 2. *Given two points $x, y \in G$. y is said to be dominated by x ($x \prec y$) if $f_i(x) \leq f_i(y), i = 1, 2, ..., k$ and $f_j(x) < f_j(y), j \in 1, 2, ..., k$.*

In Figure 2.1 the value of f_2 in points A and C are the same, but the value of f_1 of A is less than the one from C, so we can confirm that $A \prec C$. In the same figure it is visible that B dominates C, but A and B do not dominate each other.

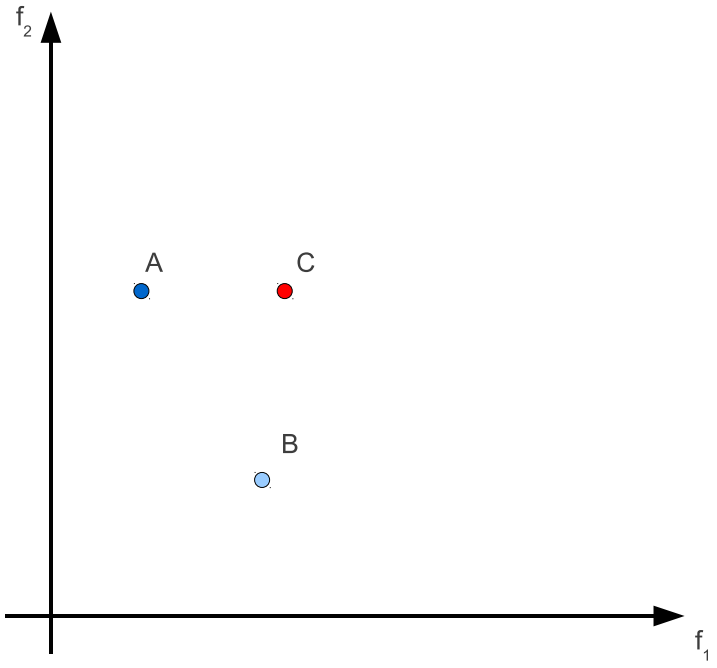


Figure 2.1: Example that illustrates Pareto dominance

2.2.3 Pareto front

The points A and B in Figure 2.1 do not dominate each other. More generally, in a MOP it is possible that there exists a set of solutions such that all elements of this set do not dominate each other. Hence, they are the "best" solutions that can be found. From this idea it is possible to present the definition of *Pareto optimality*.

Definition 3. *Given two points $x, y \in G$, and a vector function $F : G \subset \mathbb{R}^n \to \mathbb{R}^k$, then a point $y \in G$ is called Pareto optimal, if there does not exist a $x \in G$ which dominates y .*

The set of all Pareto optimal solutions of a MOP is called the *Pareto set*, and can be formalized as:

$$PS = \{x \in G | \nexists y \in G \qquad y \prec x\} \tag{2.8}$$

Figure 2.2 presents the so-called *Pareto front* $F(PS)$, which is the set of images of the points of the *Pareto Set*.

2.2.4 Weak Pareto Optimality

Another concept widely used in multiobjective optimization is *weak Pareto optimality*. It is possible to say that a vector is weakly Pareto optimal if there does not exist another vector for which all the components dominate it. Formally, weakly Pareto optimality can be formally defined as follows:

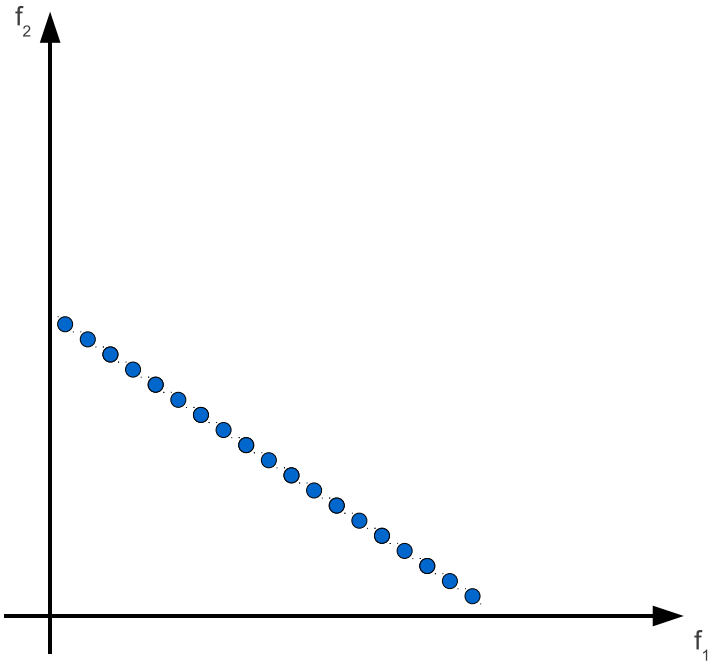


Figure 2.2: Example of a Pareto front

Definition 4. *Given a point $x^* \in G$, is called weakly Pareto optimal if does not exist a point y, such that $f_i(y) < f_i(x), \forall i = 1, 2, ..., k$.*

In Figure 2.3, the concept of weak Pareto optimality is presented. Here, all the points are weakly Pareto optimal, but only the yellow squares can be considered Pareto optimal.

2.2.5 Karush-Kuhn-Tucker point

If all the objectives in an unconstrained MOP are differentiable the theorem of Kuhn and Tucker [Kuhn and Tucker, 1951] states a necessary condition for Pareto optimality:

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

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(x^*) = 0. \tag{2.9}$$

This theorem claims that the zero vector can be expressed as a convex combination of the gradients of the objective functions, but this theorem does not state a sufficient condition for Pareto optimality. For the constrained case we refer e.g. to [Miettinen, 1999].

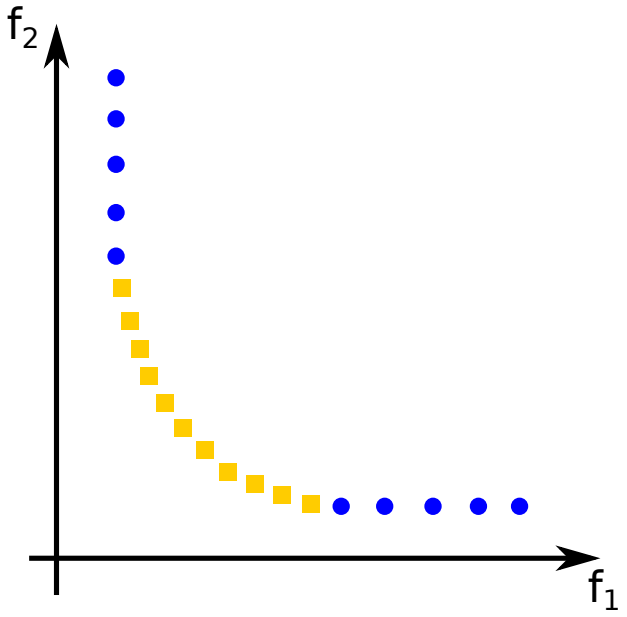


Figure 2.3: Pareto points and weak Pareto points

Definition 5. *A point $x \in \mathbb{R}^n$ is called a Karush-Kuhn-Tucker (KKT) point if there exist scalars $\alpha_1, ..., \alpha_k$ such that $\sum_{i=1}^{k} \alpha_i = 1 \geq 0$ and Equation (2.9) is satisfied.*

2.3 Classical Methods

The first approaches to solve MOP were typically based on trying to convert the problem into an auxilar SOP, and then to use numerical methods to find the optimal solution for this auxiliar problem. Since the solution of a SOP consists typically of one point, one application of the method will lead ideally to one Pareto optimal solution. An approximation of the entire Pareto set can be obtained by solving a clever sequence of SOPs, see, e.g., [Miettinen, 1999], [Das and Dennis, 1998], [Fliege and Svaiter, 2000], [Ehrgott, 2005].

2.3.1 Weighted sum method

One of the oldest algorithms used to solve a MOP is the *weighted sum method* (for reference see [Ehrgott, 2005]). This method aggregates all objectives f_i into one auxiliary objective F_w by considering a weighted sum of all the f_i's. To be more precise, given weights $w_i, i = 1, ..., k$, $w_i \geq 0$, and $\sum w_i = 1$, the auxiliary objective

is as follows:

$$F_w(x) = \sum_{i=1}^{k} w_i f_i(x). \tag{2.10}$$

In general, it is not a trivial task to choose w_i, because *a priori* is not known which are the weights that lead us to one or another solution.

Further, the method has problems to find Pareto optimal soluctions x such that the image $F(x)$ is located on concave parts of the Pareto front. See [Dennis and Schnabel, 1987] for a more thorough discussion.

2.3.2 The ϵ-constraint method

Another widely used approach that can solve an MOP is the ϵ-constraint method ([Ehrgott, 2005]). The main idea of this method is to optimize one objective while the other objectives are transformed into constraints with a permissible error ϵ. In this manner, a MOP can be reformulated as:

$$\begin{array}{ll} \min & f_i(x) \\ x \in G & \\ f_j(x) \leq \epsilon_j & j \neq i \end{array}. \tag{2.11}$$

This method is capable of finding several solutions of a MOP by using different values of ϵ. Also, it is important to mention that this method can be used in non-convex functions. The choice of the values of ϵ_j depend of the kind of problem at hand.

2.4 Stochastic Methods

Bio-inspired heuristic algorithms are widely used methods adopted to solve MOPs. Evolutionary Algorithms (EAs) are population-based algorithms that have their foundation on the evolution of species (for more information see [Coello et al., 2007], [Deb, 2001]). There is plenty of evidence that these stochastic methods are a good alternative in problems where the gradient methods can not find a global solution. So, in general, EAs are used to perform a global search in all the objectives. This leads to a good approximation of the PS constructed by the individuals of the population. The main disadvantage of EAs is their low convergence rate when generating the Pareto front of a MOP, specially in the treatment of high-dimensional problems.

The main idea of EAs is to depart from a randomly generated population, and take a subset of individuals consisting of the "best ones" generally selected according to a fitness function (commonly the values of their objectives are taken); in particular, when dealing with a MOP it is usual to select the individuals according to Pareto

dominance.

After the selection process, a new population is constructed by recombining the best individuals (the most common operators used to recombining individuals are the crossover and the mutation operators). The generation of individuals continues until some stopping criteria are reached. The pseudocode in Algorithm 1 illustrates the way in which EAs work.

Algorithm 1 Pseudocode of an evolutionary algorithm

Input: n, g, F
Output: P_g

1: Randomly initialize the population P_g with n individuals.
2: Evaluate the fitness of each individual in the population using the function F.
3: **while** Stop condition is not accomplished **do**
4: $g = g + 1$
5: Select P'_g from $P_{(g-1)}$ using the fitness of individuals.
6: Apply crossover operator to $P'_g \rightarrow P''_g$
7: Apply mutation operator to $P''_g \rightarrow P'''_g$
8: Set $P_g = P'''_g$
9: **end while**

2.4.1 MOEA/D

MOEA/D is a stochastic method introduced by [Zhang and Li, 2006]. It is presented as an alternative to solve MOPs through the decomposition of the original problem into N subproblems. The algorithm solves the N subproblems at the same time by evolving a population of solutions. MOEA/D emphatizes the solution of the subproblems by using only the information allocated in their neighborhood. The pseudocode of MOEA/D is described in Algorithm 2.

The decomposition of the problem can be performed using three different decomposition methods. The first method used to decompose a MOP is the weighted sum approach. Having a convex weight $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$, i.e. $\lambda_i \geq 0$ with $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$, the optimal solution can be found by solving the scalar optimization problem:

$$\max_x g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x). \tag{2.12}$$

Tchebycheff decomposition is another method used in MOEA/D. Here, the objective function of each N-th subproblem is defined as:

$$\min_x g^{te}(x|j, z^*) = \max[\lambda_i^j |f_i(x) - z_i^*|] \in \mathbb{R}^k, \tag{2.13}$$

where $\lambda_1, ..., \lambda_N$ is a set of evenly spread weight vectors and z^* is the ideal vector.

The last decomposition method used in the algorithm is based on the Normal Boundary Intersection method proposed by [Das and Dennis, 1998]. The main motivation of this approach is try to find intersection points of the top boundary. Having the ideal vector $z*$ and the weight vector λ, each subproblem can be defined as:

$$\begin{aligned} \min_{x,\, d} \; & g^{bi}(x|\lambda, z^*) = d \\ & z^* \in G \qquad \cdot \\ \text{subject to } & z^* - F(x) = d\lambda \end{aligned} \tag{2.14}$$

Algorithm 2 Pseudocode of MOEA/D

Input: a stopping criteria, MOP, the number of subproblems N, N weight vector $\lambda_1, \ldots, \lambda_N$, the number of weight neighborhood T
Output: EP

```
Set EP = ∅
Find the T closest weight vector of each vector.
Set B(i) = i_1, ..., i_T, with the T closest vectors to λ_i.
Generate a N random vectors x_1, ..., x_N.
Set FV^i = F(x^i).
Initializa z
while Stopping criteria not accomplished do
  for i = 1, ..., N do
    Select x_k, x_l from B(i)
    Generate y using genetic operators on x_k and x_l.
    Apply and improvement to y and generate y'.
    Update z
    Update of Neighboring Solutions
    Remove from EP the vectors dominated by F(y').
    Add y' to EP.
  end for
end while
```

2.4.2 NSGA-II

The Nondominated Sorting Genetic Algorithm II (NSGA-II) is an algorithm proposed by [Deb et al., 2002]. It presents two ideas which try to improve the common performance of a traditional MOEA.

The *non-dominated sort* is the first mechanism used in the NSGA-II. It sorts the individuals according to Pareto non-domination. A rank value is created according

to the number of individuals that dominate each solution. The non-dominated solutions receive a rank equal to one, and the others receive a rank value according to how many subsets they dominate. Figure 2.4 shows the fronts created according to different rank values.

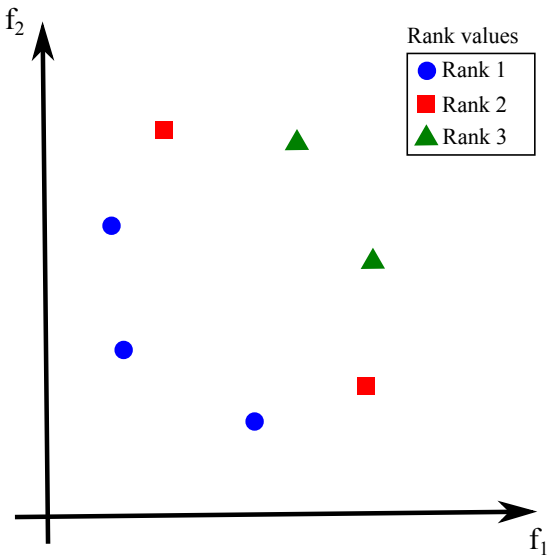


Figure 2.4: Example of non-dominated sort.

To preserve a good spread of the final solutions, the NSGA-II uses a mechanism called *crowding distance*. This distance measures the average size of the cuboid formed with the points that enclose a solution i in the population. In Figure 2.5 the cuboid of the i-th solution is represented by a dashed line.

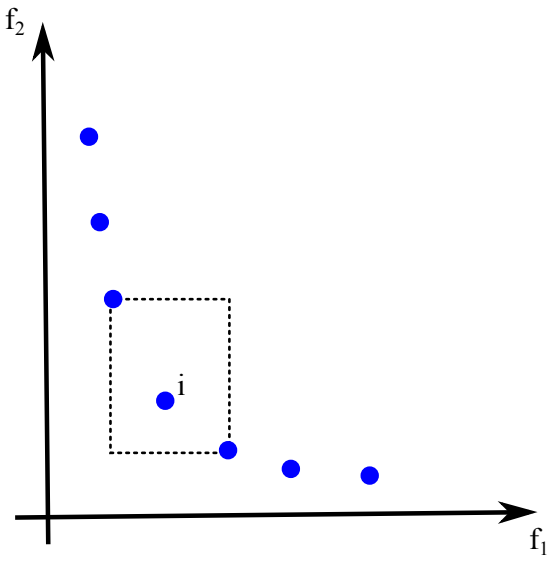


Figure 2.5: Example of crowding distance.

The main idea behind the NSGA-II is to use a selection that preserves the individuals with the lowest rank value. When almost all the solutions exist in the first rank, the algorithm selects the solutions with the highest value of the crowding distance. Algorithm 3 describes the NSGA-II evolves the solutions.

Algorithm 3 Pseudocode of the NSGA-II

Input: number of generations G, number of individuals N
Output: final population P_G

1: Randomly generate the population P_0
2: Calculate $F(P_0)$
3: Apply genetic operators in P_0 to generate Q_0
4: $i = 0$
5: **while** $i < G$ **do**
6: Set $R_i = P_i \cup Q_i$
7: Calculate the rank value of R_i
8: Calculate the crowding distance of R_i
9: Select the N individuals with the lowest rank and highest crowding distance (P_i).
10: Apply genetic operators in P_i to generate Q_i
11: Set $i = i + 1$
12: **end while**

2.5 Memetic Algorithms

The concept of *memetic algorithms* (*MAs*) (introduced by [Moscato, 1989]), refers to heuristic strategies coupled with local search techniques.

The term *memetic* is based on the english term *"meme"* presented by [Dawkins, 1989]. This term is used as the "transmission unit" in the context of cultural evolution. Quoting Dawkins:

"Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation."

In terms of computational algorithms the idea of "memes" is used to denote the improvement of particular individuals along with some mechanism of global cooperation and competition with other solutions in the population.

The main idea of *MAs* is to perform a local search improving an inital solution by searching within its neighborhood for a "better" solution that improves the initial one. Once the local search finds the "best" or at least a better solution in the neighborhood it replaces the original solution; this process is repeated until the local search does not find a solution in the neighborhood that improves the current solution. Figure 2.6 presents the behaviour of a local search technique.

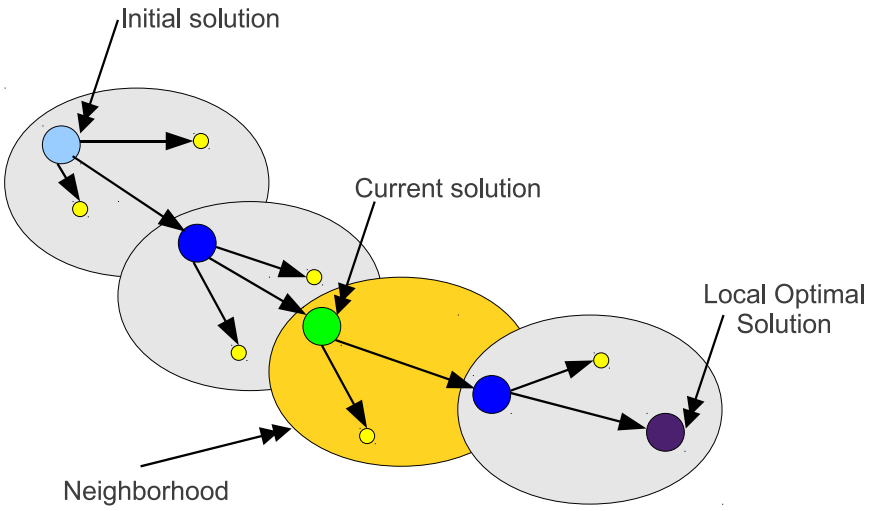


Figure 2.6: Outline of a general local search procedure.

Algorithm 4 presents a generic memetic algorithm. It is important to remark that

the time in which the local search is used can be redefined according to the structure of the EA and/or the optimization problem to be solved.

Algorithm 4 Pseudocode of a memetic algorithm

Input: n, g, F
Output: P_g

1: Randomly initialize the population P_g with n individuals.
2: Evaluate the fitness of each individual in the population using the function F.
3: **while** Stop condition is not accomplished **do**
4: $g = g + 1$
5: Select P'_g from $P_{(g-1)}$ using the fitness of individuals.
6: Apply evolutionary operators to $P'_g \rightarrow P''_g$
7: Apply local Search in the neighborhood of P''_g $(P''_g \rightarrow P'''_g)$
8: Select P_g from $\left(P_{(g-1)}, P'_g, P''_g, P'''_g\right)$
9: **end while**

2.6 The Directed Search method

The Directed Search method [Schütze et al., 2010] is a local search technique used to perform a steering movement in a direction d in image space. In order to find the next point via a line search, a direction $\nu \in \mathbb{R}^n$ needs to be calculated. In this method, an iterative scheme that performs such a movement in objective space is proposed. Figure 2.7 represents how the direction d is mapped from objective space into parameter space in the direction ν.

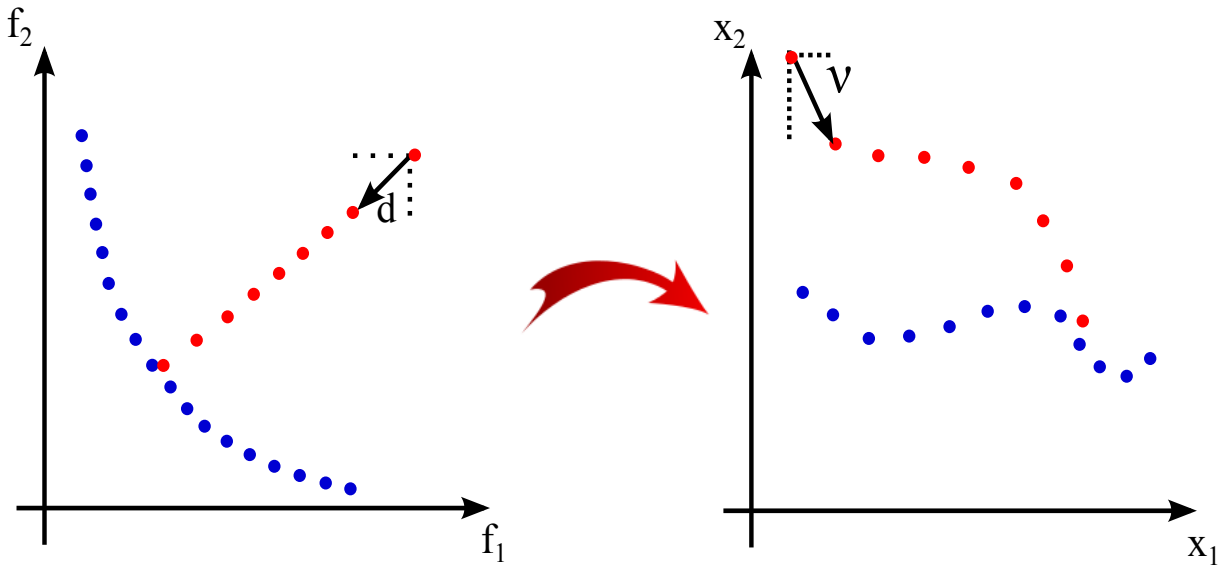


Figure 2.7: The Directed Search method.

Assuming an initial point $x_0 \in \mathbb{R}^n$ and the direction $d \in \mathbb{R}^k$, it is possible to calculate the direction ν such that $y_0 = x_0 + t\nu$, where $t \in \mathbb{R}^+$ is the step size. For ν it should hold:

$$\lim_{t \to 0} \frac{f_i(y_0) - f_i(x_0)}{t} = \langle \nabla f_i(x_0), \nu \rangle = d_i, \ i = 1, ..., k. \tag{2.15}$$

Using the Jacobian of F, Equation (2.15) can be reformulated in matrix notation as:

$$J(x_0)\nu = d \tag{2.16}$$

Since typically the number of parameters is higher than the number of objectives, to calculate the direction ν it is necessary to solve an underdetermined system of equations. To prevent getting many solutions to the problem caused by the underdetermined system, the solution with the lowest 2-norm can be viewed as the greedy solution in this context. Hence, using Equation (2.16), this leads to:

$$\nu = J(x_0)^+ d, \tag{2.17}$$

where $A^+ \in \mathbb{R}^{n \times k}$ denotes the so-called pseudoinverse. In case that A is maximal, the pseudoinverse can be calculated as $A^+ = A^T(AA^T)^{-1}$ [Nocedal and Wright, 2006].

Using the previous result, the problem is the same as numerically solving the following initial value problem:

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x} &= J(x(t))^+ d \end{aligned} \tag{2.18}$$

Assuming that d is a "descent direction" in objective space (i.e., $d_i \leq 0$ for all $i = 1, \ldots, k$ and there exists an index j such that $d_j < 0$), the numerical solution of Equation (2.18) can be considered as a hill climber for a MOP.

When an endpoint of the curve described in (2.18) is reached, it is possible to detect it under certain assumptions: If the number of parameters n is at least as large as the number of objectives k, and all the gradients at x_0 are linearly independent from each other; it is possible to detect along the points in the curve when the value of $\kappa(J(x))$ increase over a certain tolerance *tol* in order to find when the objectives start to conflict between them. For the condition number of the Jacobian, it holds:

$$\kappa(J(x)) = \sqrt{\frac{\lambda_{max}\left[J(x)^T J(x)\right]}{\lambda_{min}\left[J(x)^T J(x)\right]}} \to \infty \quad \text{for} \quad x \to x^*, \tag{2.19}$$

where $\lambda_{max}(A)$ and $\lambda_{min}(A)$ denote the largest and smallest eigenvalues of matrix A, and x^* is a point in the end in the curve of Equation (2.18).

With the stopping criteria already defined, it is possible to state the algorithm for the Directed Search (see Algorithm 5).

Algorithm 5 Pseudocode of the Directed Search

Input: starting point $x_0 \in \mathbb{R}^n$ with $rank(J(x_0)) = k$, $tol \in \mathbb{R}^+$, convex weight $\alpha_0 \in \mathbb{R}^k$.

1: $i := 0$
2: **while** $\kappa_2(J(x_i)) < tol$ **do**
3: compute $\nu_i = J(x_i)^+ d_i$
4: compute $t_i \in \mathbb{R}^+$
5: set $x_{i+1} = x_i + t_i \nu_i$
6: choose $d_{i+1} \in \mathbb{R}^k$
7: set $i := i + 1$
8: **end while**

2.6.1 Approximating the Jacobian

The main problem that can be presented in the DS method can be the requirement of the Jacobian matrix of the functions. One possible solution to break this dependency is the use of a numerical approach to approximate the derivatives, using the finite differences method, the j_i, $i = 1, \ldots, n$ column vectors of the Jacobian can be approximated as:

$$j_i(x_0) = \frac{f(x_0 + he_i) - f(x_0)}{h},$$

where $x_0, I \in \mathbb{R}^n$ and e_i is the i-th unit vector $h \in \mathbb{R}^+$.

Finite differences can obtain good results in low dimensions, but the computational cost is $O(nk)$ in terms of additional function calls, and this cost increases as the number of the parameters gets higher.

2.7 Other methods

2.7.1 Normal Boundary Intersection

Das and Dennis [Das and Dennis, 1998] proposed the Normal Boundary Intersection (NBI) technique. This method tries to find points along the PF making use of a *convex hull of individual minima (CHIM).*

Definition 6. *Let x_i^* be the k global minimizers of the function $F : \mathbb{R}^n \to \mathbb{R}^k$, set $F_i^* = F(x_i^*)$, and having the shadow minima F^*. Let $\Phi \in \mathbb{R}^{k \times k}$ be the matrix whose column vectors is $F_i^* - F^*$, also called the shifted pay-off matrix. Let equation $B =$*

$\left\{\beta : \beta \in \mathbb{R}^k, \sum_{i=1}^{n} \beta_i = 1, \beta_i \geq 0\right\}$ *represent the set of "convex combination vectors". Then the set of point in* $\mathbb{R}^k$ *that are convex combinations of* $F_i^* - F^*$ *i.e.* $\{\Phi\beta : \beta \in B\}$, *is referred to as the convex hull of individual minima.*

Having a convex combination vector $\beta \in B$, a point in the CHIM ($\Phi\beta$). Let $\Phi\beta + t\hat{n}$, with $t \in \mathbb{R}$ represent the set of points and the normal vector of the CHIM ($\widetilde{n}$) that point towards the origin. It is possible to calculate the intersection between $\hat{n}$ and the PF by solving the problem:

$$\begin{array}{c} \max t \\ x \in G \\ \text{subject to } \Phi\beta + t\hat{n} = F_i^* - F^* \end{array} . \tag{2.20}$$

2.7.2 Descent direction of Fliege and Svaiter

In [Fliege and Svaiter, 2000] a steepest descent method is proposed. Having an initial point $x \in \mathbb{R}^n$, the matrix A is defined as:

$$A = J(x)$$

and the function $f_x : \mathbb{R}^n \to \mathbb{R}^k$ by:

$$f_x(\nu) = \max\{(A\nu)_i | i = 1, \ldots, k\}.$$

Considering the unconstrained minimization problem described as:

$$\begin{array}{c} \min f_x(\nu) + \frac{1}{2}\|\nu\|^2 \\ \text{subject to } \nu \in \mathbb{R}^n \end{array} \tag{2.21}$$

.

Assuming that the objective function is proper, closed and strongly convex, it is possible to reformulate Equation (2.21) into:

$$\begin{array}{c} \min \alpha + \frac{1}{2}\|\nu\|^2 \\ \text{subject to } (A\nu)_i \leq \alpha, i = 1, \ldots, k \end{array} . \tag{2.22}$$

Lemma 1. *Let* $\nu(x)$ *and* $\alpha(x)$ *be the solution and the optimum value of problem* (2.22), *respectively.*

1. *If* x *is Pareto critical, then* $v(x) = 0 \in \mathbb{R}^n$ *and* $\alpha(x) = 0$.
2. *If* x *is not Pareto critical, then* $a(x) < 0$. *It should hold:*

$$\begin{array}{c} f_x(\nu(x)) \leq -\frac{1}{2}\|\nu\|^2 < 0 \\ (J(x)\nu(x))_i \leq f_x(\nu(x)), i = 1, \ldots, m \end{array} \tag{2.23}$$

3. *The mappings* $x \to v(x)$ *and* $x \to \alpha(x)$ *are continuous.*

2.7.3 The Multi-Agent Collaborative Search

In [Vasile and Zuiani, 2011] a memetic algorithm is implemented. The *Multi-Agent Collaborative Search* (MACS) method uses several mechanisms to find local solutions. The algorithm generates the population through stochastic mechanisms (crossover and mutation), then it takes the individuals that are Pareto dominated and applies them local search strategies (individualistic actions) in order to improve such candidate solutions. The principal individualistic actions are based on algorithms to such as hill climbers, interpolation and extrapolation between individuals and differential evolution ([Storn and Price, 1997]).

2.7.4 The Hill Climber with Sidestep

The *Hill Climber with Sidestep* (HCS) proposed by [Lara et al., 2010] is a novel local search idea to perform a hillclimber movement in order to reach the PF. It uses a random point x_l^2 in the neighborhood of x_l^1 to construct a direction $\nu_{acc} \in \mathbb{R}^n$, such that:

$$\nu_{acc} = \begin{cases} x_l^1 - x_l^2 & \text{if} \quad x_l^1 \prec x_l^2 \\ x_l^2 - x_l^1 & \text{otherwise} \end{cases}.$$

Next, the algorithm uses the descent direction in order to perform a line search technique that throws a new candidate solution. While the solutions dominate each other it means that these solutions exist "far away" from the PF. But, when the solutions found are mutually non-dominated and the process starts to repeat N_{nd} many times, this indicates that the point is already near to the PF, and hence the algorithm will try to move along the PF.

The gradient free version of HCS, uses all the information obtained in the N_{nd} iterations in order to find a direction that describes a predictor step like movement, that is supposed to be orthogonal to the PF. Having $i_0 \in \{1, 2\}$, we can define the predictor step as:

$$\alpha = \frac{1}{N_{nd}} \sum_{k=1}^{N_{nd}} s_{l_k} \frac{x_{l_k}^2 - x_{l_k}^1}{\left\| x_{l_k}^2 - x_{l_k}^1 \right\|}. \tag{2.24}$$

The N_{nd} is the number of iterations used to recopilate information. It is important to mention that N_{nd} only present the cases when x_l^1 and x_l^2 do not dominate each other. Hence, s_l can be defined as:

$$\nu_{acc} = \begin{cases} 1 & \text{if} \quad f_{i_r}(x_l^2) \prec f_{i_r}(x_l^1) \\ -1 & \text{otherwise} \end{cases}. \tag{2.25}$$

where $i_r \in i_0$ is selected randomly.

Chapter 3

The Discrete Directed Search method

The use of the Directed Search method as described in the previous chapter makes use of the gradient of the objectives. Hence, with MOPs without such a type of information, the use of an approximation of the gradient becomes an important tool. The use of the DS method can become a problem when it is used in high-dimensional problems (e.g. when $n \gg 50$), because the computational cost of the approximation is expensive (it requires $n \times k$ function evaluations). In some cases it can be more effective to choose a stochastic local search technique.

The main motivation in this work is to try to improve the Directed Search by decreasing the computational cost of the approximation when it is used on objectives whithout gradient information.

3.1 The method

Having an initial solution $x_0 \in \mathbb{R}^n$ and r search directions $\nu_i \in \mathbb{R}^n, i = 1, \ldots, r$, define the matrix $\mathcal{F}(x_0) \in \mathbb{R}^{k \times r}$ as follows:

$$\mathcal{F}(x_0) = (\langle \nabla f_i(x_0), \nu_j \rangle)_{\substack{i=1,\ldots,k \\ j=1,\ldots,r}} . \tag{3.1}$$

Hence, each entry m_{ij} of the matrix $\mathcal{F}$ is defined by the directional derivative of each objective f_i in direction ν_j, $m_{ij} = \nabla_{\nu_j} f_i(x_0)$. Then it holds:

Proposition 1. *Let $x_i, \nu_i \in \mathbb{R}^n, i = 1, \ldots, r$, $\lambda \in \mathbb{R}^r$ and $\nu = \sum_{i=1}^{r} \lambda_i \nu_i$. Then*

$$DF(x)\nu = \mathcal{F}(x)\lambda. \tag{3.2}$$

Proof. It is:

$$\mathcal{F}(x)\lambda = \begin{pmatrix} \langle \nabla f_1(x), \nu_1 \rangle & \dots & \langle \nabla f_1(x), \nu_r \rangle \\ \vdots & \vdots & \vdots \\ \langle \nabla f_k(x), \nu_1 \rangle & \dots & \langle \nabla f_k(x), \nu_r \rangle \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_r \end{pmatrix}, \tag{3.3}$$

and

$$J(x)\nu = J(x)\left(\sum_{i=1}^{r} \lambda_i \nu_i\right) = \sum_{i=1}^{r} \lambda_i \begin{pmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_k(x)^T \end{pmatrix} \nu_i. \tag{3.4}$$

Hence for the l-th component of both products it holds:

$$(\mathcal{F}(x)\lambda)_I = \sum_{i=1}^{r} \lambda_i \langle \nabla f_i(x), \nu_i \rangle = (J(x)\nu)_l, \tag{3.5}$$

and the desired identity follows. □

Hence, instead of solving Equation (2.16), one can solve the equation:

$$\mathcal{F}(x)\lambda = d, \tag{3.6}$$

and define:

$$\nu = \sum_{i=1}^{r} \lambda_i \nu_i. \tag{3.7}$$

Now, the entries m_{ij} of matrix $\mathcal{F}$ are required. Having the starting point x_0 the direction ν_j can be defined as follows:

$$\nu_j = \frac{x_j - x_0}{\|x_j - x_0\|_2}, t_j = \|x_j - x_0\|_2, \qquad j = 1, \dots, r \tag{3.8}$$

Using Equation (3.8) the entries m_{ij} can be approximated by:

$$\begin{aligned} m_{ij} &= \langle \nabla f_i(x_0), \nu_j \rangle = \lim_{t \to 0} \frac{f_i(x_0 + t\nu_j) - f_i(x_0)}{t} \\ &\approx \frac{f_i(x_0 + t\nu_j) - f_i(x_0)}{\|x_j - x_0\|_2}, \qquad i = 1, \dots, k, \quad j = 1, \dots, r \end{aligned}. \tag{3.9}$$

Now that the new approximation is constructed, it is desirable to calculate the numerical error. From the discussion above it holds that the computational error is defined as follows:

$$\langle \nabla f_i(x_0), \nu_j \rangle = \frac{f_i(x_j) - f_i(x_0)}{\|x_j - x_0\|_2} + O\left(\|x_j - x_0\|_2\right) \tag{3.10}$$

It is important to remark that with this approximation, the search direction can be computed without any additional function evaluation.

Until now it is impossible to know if there exists a solution for Equation (3.6), so it is important to investigate the behavior of the problem according to the selection of r and the ν_i's. For this, it is advantageous to write $\mathcal{F}(x)$ as follows:

$$\mathcal{F}(x) = J(x)V, \tag{3.11}$$

where $V = (\nu_1, \ldots, \nu_r) \in \mathbb{R}^{n\times r}$ represents the matrix where each row is given by the search direction ν_i.

If $\text{rank}(J(x)) = k$ (which is given for a non-boundary point x), it is known from linear algebra that:

$$\text{rank}(J(x)) = k \Rightarrow rank(\mathcal{F}(x)) = rank(V). \tag{3.12}$$

If, on the other hand, x is a boundary point (and hence, $\text{rank}(J(x)) < k$), then it follows by the rank theorem of matrix multiplication that also $rank(\mathcal{F}(x)) < k$ regardless of the choice of V (i.e., regardless of the number r and the choice of the search directions ν_i).

This indicates that the condition number of $\kappa_2(\mathcal{F}(x))$ can be used to check numerically if a current iterate is already near to an endpoint of (2.18) (in the page 19). Equation (3.12) indicates that the ν_i's should be chosen such that they are linearly independent. If in addition the search directions are orthogonal to each other, a straightforward calculation shows that:

$$V \text{orthogonal} \Rightarrow \kappa_2(\mathcal{F}(x)) = \kappa_2(J(x)). \tag{3.13}$$

In that case, the condition number $\kappa_2(\mathcal{F}(x))$ can indeed be used as a stopping criteria.

It is $V_+^{(r)} \to V_r$ for $r \to n$. This is due to the fact that VV^T (i.e., the orthogonal projection on to span $\{V_1, \cdots, V_r\}$) converges to the identity matrix $I_n \in \mathbb{R}^{n\times n}$. One way to see this is as follows: Let $V_1, \cdots, V_n$ be an orthogonal basis of the $\mathbb{R}^n$, whereby $V_1, \cdots, V_r$ are identical to the column vectors of V. Further, let $x \in \mathbb{R}^n$. Since $\{V_1, \cdots, V_r\}$ is a basis of $\mathbb{R}^n$, there exist scalars $N_1, \cdots, N_n \in \mathbb{R}$ such that $x = \sum_{i=1}^{n} N_i V_i$. Then:

$$\begin{aligned} VV^T &= \sum_{i=1}^{r} V_i V_i^T \left(\sum_{j=1}^{n} N_i V_i \right) \\ &= \sum_{i=1}^{r} \sum_{j=1}^{n} N_j \langle V_i, V_j \rangle = \sum_{i=1}^{r} N_i V_i \end{aligned} . \tag{3.14}$$

Hence, it holds:

$$\left\|VV^T x - I_n x\right\| = \left\|\sum_{i=r+1}^{n} N_i V_i\right\|, \tag{3.15}$$

which diminishes for increasing value of r. For $r = n$ it holds:

$$\begin{aligned} V_+^{(n)} &= V\mathcal{F}(x)^+ d = VV^T J(x)^T \left(J(x)VV^T\right)^{-1} d \\ &= J(x)^T \left(J(x)J(x)^T\right)^{-1} d = J(x)^+ d = V^+ \end{aligned}. \tag{3.16}$$

Example 1. Using the bi-objective model proposed in [Köppen and Yoshida, 2007]:

$$\begin{aligned} F &: \mathbb{R}^n \to \mathbb{R}^2 \\ f_i(x) &= \|x - a_i\|_2^2, \quad i = 1, 2, \end{aligned} \tag{3.17}$$

where $a_1 = (1, \ldots, 1)^T, a_2 = (-1, \ldots, -1)^T \in \mathbb{R}^n$. The Pareto set is given by the line segment between a_1 and a_2, i.e.,

$$\mathcal{P} = \{x \in \mathbb{R}^n \ : \ x_i = 2\alpha - 1, i = 1, \ldots, k, \alpha \in [0, 1]\} \tag{3.18}$$

Let $r = 2$ and $v_1 := e_i$ and $v_2 := e_j$, $i \neq j$, where e_i denotes the i-th canonical vector. Then, it is

$$\mathcal{F}(x) = \begin{pmatrix} x_i - 1 & x_j - 1 \\ x_i + 1 & x_j + 1 \end{pmatrix} \tag{3.19}$$

It is $\det(\mathcal{F}(x)) = 1/(2(x_i - x_j))$, and hence,

$$\det(\mathcal{F}(x)) = 0 \quad \Leftrightarrow \quad x_i = x_j, \tag{3.20}$$

by which it follows that it is $rank(\mathcal{F}(x)) = 2$ for all $x \in \mathbb{R}^n \backslash B$, where $B := \{x \in \mathbb{R}^n \ : \ x_i = x_j\}$ (note that $\mathcal{P} \subset B$).

Since B is a zero set in $\mathbb{R}^n$, the probability is one that for a randomly chosen point $x \in \mathbb{R}^n$ the matrix $\mathcal{F}(x)$ has full rank, and hence, that Equation (3.1) has a unique solution. To be more precise, it is $\nu = \lambda_1 e_i + \lambda_2 e_j$, where

$$\begin{aligned} \lambda = \mathcal{F}^{-1}(x)d &= \frac{1}{\det(\mathcal{F}(x))} \begin{pmatrix} x_{j+1} & -x_j + 1 \\ -x_i - 1 & x_j - 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \\ &= \frac{1}{2(x_i - x_j)} \begin{pmatrix} x_j(d_1 - d_2) + d_1 + d_2 \\ x_i(d_2 - d_1) - d_1 - d_2 \end{pmatrix}. \end{aligned} \tag{3.21}$$

Note that this holds regardless of the number $n \geq 2$ of the parameter dimension.

The above considerations show that already for $r = k$ search directions ν_i, $i = 1, \ldots, r$, one can find a descent direction $\tilde{\nu}$ by solving Equation (3.1).

However, by construction it is $\nu \in \text{span}\{\nu_1, \ldots, \nu_k\}$ which means that only a k-dimensional subspace of the $\mathbb{R}^n$ is explored in one step. One would expect that the more search directions ν_i are taken into account, the better the choice of $\tilde{\nu}$ is. This is indeed the case: For $r > k$, we suggest to choose analog to (2.17)

$$\nu_+^{(r)} := \sum_{i=1}^{r} \lambda_i \nu_i, \quad \text{where} \quad \lambda = \mathcal{F}(x_0)^+ d \tag{3.22}$$

The following discussion gives a relation between $\nu_+^{(r)}$ and ν_+ for non-boundary points x for the case that the ν_i's are orthonormal: It is

$$\nu_+ = J^+(x)d = J(x)^T(J(x)J(x)^T)^{-1}d \tag{3.23}$$

and

$$\begin{aligned} \lambda = \mathcal{F}(x)^+ d &= V^T J(x)^T (J(x) \underbrace{VV^T}_{I} J(x)^T)^{-1} d \\ &= V^T \underbrace{J(x)^T (J(x)J(x)^T)^{-1} d}_{\nu_+} = V^T \nu_+ \end{aligned} \tag{3.24}$$

and hence

$$\nu_+^{(r)} = \sum_{i=1}^{r} \lambda_i \nu_i = \sum_{i=1}^{r} \langle \nu_i, \nu_+ \rangle \nu_i \tag{3.25}$$

For instance, when choosing $\nu_i = e_{j_i}$, Equation (3.25) gets simplified:

$$\nu_+^{(r)} = \sum_{i=1}^{r} \nu_{+,j_i} e_{j_i}, \tag{3.26}$$

i.e., $\nu_+^{(r)}$ has only r entries which are identical to the corresponding entries of ν_+. In both cases $\nu_+^{(r)}$ gets closer to ν_+ with increasing number r and for $r = n$ it is $\nu_+^{(r)} = \nu_+$.

Finally the developed algorithm requires the calculation of the optimal step size for each iteration. Hence, the step size calculation presented by [Mejia and Schütze, 2010] was used. In the original algorithm the step size was calculated for initial value problems, so in consideration that the DS can be stated as that type of problem, it could

be implemented along with it.

The step size calculates the gain $\gamma(x_i)$ of each iteration in the Directed Search:

$$\gamma(x_i) = \frac{(f(x_i) - f(x_{i-1}))^T d}{d^T d}. \tag{3.27}$$

From above, it is possible to approximate the curvature σ_i of the solution curve of the IVP.

$$\sigma_i = \frac{\dfrac{\gamma(x_i) - \gamma(x_{i-1})}{x_i - x_{i-1}} - \dfrac{\gamma(x_{i-1}) - \gamma(x_{i-2})}{x_{i-1} - x_{i-2}}}{x_i - x_{i-2}}. \tag{3.28}$$

Finally, the step size can be formulated as follows:

$$t = \frac{1}{\|\sigma_i\|}. \tag{3.29}$$

As mentioned above, the proposed algorithm is used to calculate the step size that is going to be used in the DDS method. Along with the calculation of the optimal length of the step size, this algorithm also restricts the length of the step size t such that the movement of the line search stays close enough to the original direction d.

In order to maintain a good direction, the angle β is introduced. β measures the angle formed between the direction d and the line segment formed with the point x_i and the point after the line search has been performed x_{i+1}. It is desirable that the angle β is as small as possible in order to make the correction step be easier to perform.

Figure 3.1 illustrates how the tolerance of β is used. The point A represents the point with the maximal value of t and point B represents the point using a step size after k iterations of the algorithm, t_k. The value of the step size has been reduced in order to fit at the β tolerance.

In the step size algorithm a parameter $\rho \in (0, 1)$ is used to reduce the final length of the step size in each iteration. In Algorithm 6, it is possible to see that at each iteration, the value of the step size t is reduced by a factor ρ.

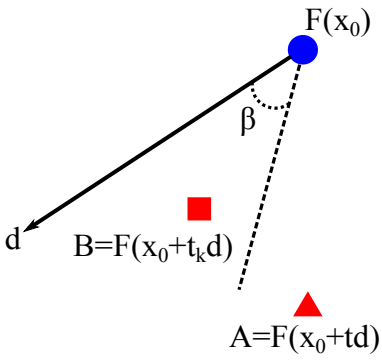


Figure 3.1: Graphical explanation of the β angle.

Algorithm 6 Pseudocode of the step size control.

Input: β, x_i, ρ, ν, d
Output: t

compute t as in (3.29)
$y = x_i + t\nu$
$F_x = F(x_i)$
$F_y = F(y)$
while $\angle(F_x - d, F_x - F_y) > \beta$ **do**
    $t = t\rho$
    $y = x_i + t\nu$
    $F_x = F(x_i)$
    $F_y = F(y)$
end while
return t

3.2 Comparison between the DS method and the DDS method

If the DS and the DDS methods are compared, the first significant point is that the DDS do not need gradient information. DDS is a method that uses the neighborhood information of the current iteration in order to construct an approximation of the gradient. It is important to note that the error can be ignored.

A key feature of the DDS is that it was originally developed in order to work along an EA, and is also known that in an EA the information of the individuals comes for "free". Hence, an important task to construct a memetic algorithm can be the selection of the individuals that are going to be used in order to construct the Jacobian approximation. Figure 3.2 presents the idea of taking the r directions, which correspond to the $r = 2$ individuals in the neighborhood of x_0 ($N(x_0)$).

Another important issue about the DDS is that even if a model does not have gradient information, the algorithm can use the approximation of the Jacobian with an affordable computational cost (it requires only the function evaluations for the step size control).

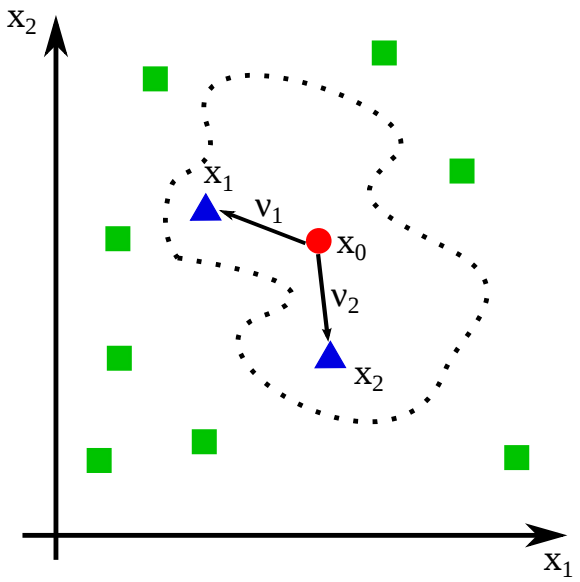


Figure 3.2: Example of neighborhood in a memetic algorithm.

Thus, the main advantages of the DDS method in comparison with the DS can be stated as follows:

- It can work on problems where a function evaluation is computational expensive.
- It can work on problems where the exact gradient information is hard to find.

3.3 Numerical Results

In order to illustrate the performance of the DDS in a multiobjective problem, an academic function is used here [Schütze et al., 2011]:

$$F(x) = \begin{pmatrix} \|x - a_1\|_2^2 \\ \|x - a_2\|_2^2 \\ \vdots \\ \|x - a_k\|_2^2 \end{pmatrix}, \tag{3.30}$$

where $x \in \mathbb{R}^n$ and $a_i \in \mathbb{R}^n, i = 1, \ldots, k$.

This experiment were performed in order to compare the DS and the DDS algorithms. To make a fair test both algorithms started with the same initial random point x_0. The parameters used in the tests are shown in Table 3.1.

Parameter	**Value**
n	100
k	2
a_i	$a_1 = (1, 1, \ldots, 1), a_2 = (-1, -1, \ldots, -1)$
r	5, 10, 15, 20, 25
tol	10^2
d	$(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$

Table 3.1: Parameters used in the comparission between DS and DDS.

In order to fulfill the requirement of the DDS that the search directions ν_i are linearly independent, the following mechanism is presented. First, a random individual $x_j \in \mathbb{R}^n$ is calculated, such that a QR decomposition of x_j can be obtained. Next, the j column vector is taken from the Q matrix, such that the column vectors are also linearly independent with each other. Hence, these solutions also represent the best possible case which is obtained when the solutions are orthogonal with respect to each other. Next, having a small step size t_n (in this case 10^{-6}) the construction of the neighbors took place. Each neighbor solution is used to construct the matrix $\mathcal{F}(x)$ as:

$$\nu_i = x_0 + t_n v_q, i = 1, \cdots, r, \tag{3.31}$$

where v_1 refers to the r column vector taken from Q matrix.

Table 3.2 presents the number of function evaluations, and the number of iterations needed to reach the tolerance *tol* adopted. In the table is also presented the number of Jacobian matrices that the DS method requires.

Parameter	**DS**	**DDS**				
	-	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$
Function evaluations	$100 + 10J(x)$	2130	1921	1617	1709	1641
No. of iterations	10	217	129	82	69	55

Table 3.2: Results from comparing DS and DDS.

From the results presented in Table 3.2, it can be pointed out that when the value of r increases, the number of function evaluations is significantly reduced, and also the number of iterations of the algorithm is decreased. Hence, the main problem with the use of the DDS is to try to define the maximal number of neighbors that is allowed to take from an EA. To calculate the results, the exact gradient of the function was used, but in order to allow a fair comparison of both algorithms Table 3.3 presents the function evaluations required in order to approximate the Jacobian matrix by two state-of-the-art methods. The first method presented is finite differences, and the other approximation is calculated by using automatic differentiation [Rall, 1981] approach.

Algorithm	**Function Evaluations**
DS(Finite differences)	2000
DS(Automatic Differentiation)	100

Table 3.3: Comparision of the gradient approximations.

In Figures 3.3-3.8 the graphical results obtained from this experiment are presented. From the graphical results, it is possible to state that the behavior of the DDS in comparison to the DS becomes the same as long as the r value increases. Also, it is important to remark that the DDS performs "bigger" steps when the value of r is incremented. Here, it is important to investigate the possible balance that exists between the value of r and the convergence rate, because it is possible that a better choice of the direction d can greatly increment the convergence rate of the algorithm. Finally, it is possible to confirm that the selection of the neighbors of the DDS is not a trivial task, and the selection must be reviewed when a new model is presented in order to be solved.

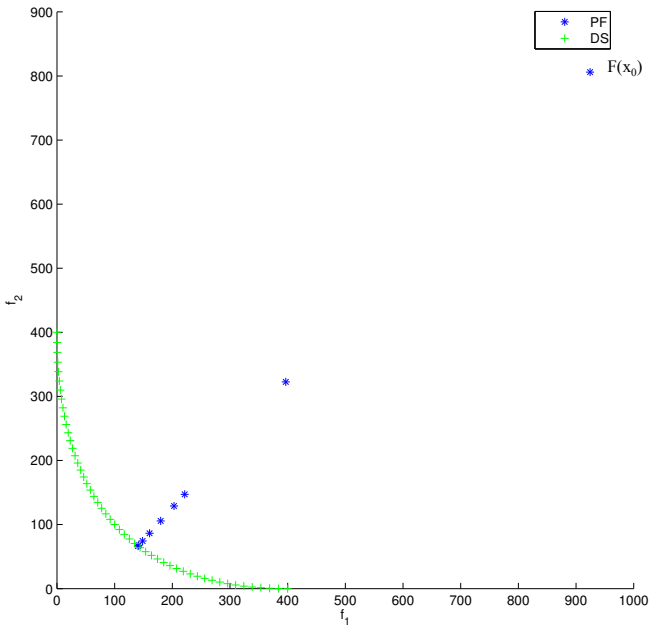


Figure 3.3: Results for the DS method.

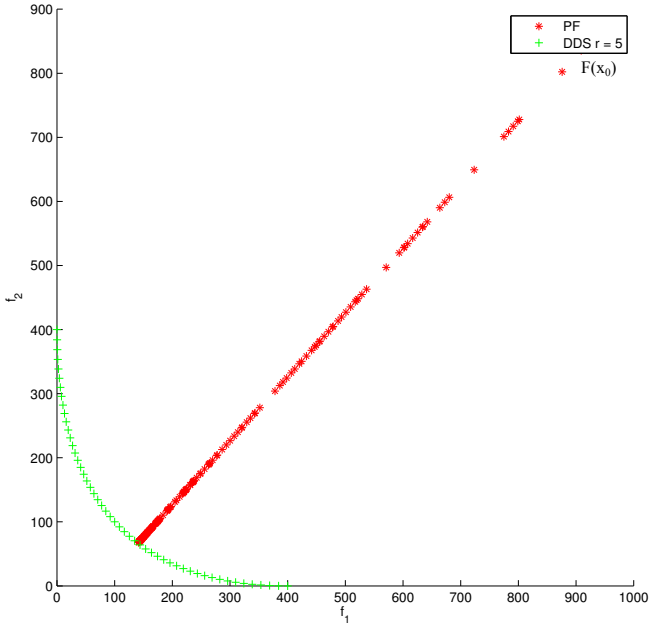


Figure 3.4: Results for the DDS method with $r = 5$.

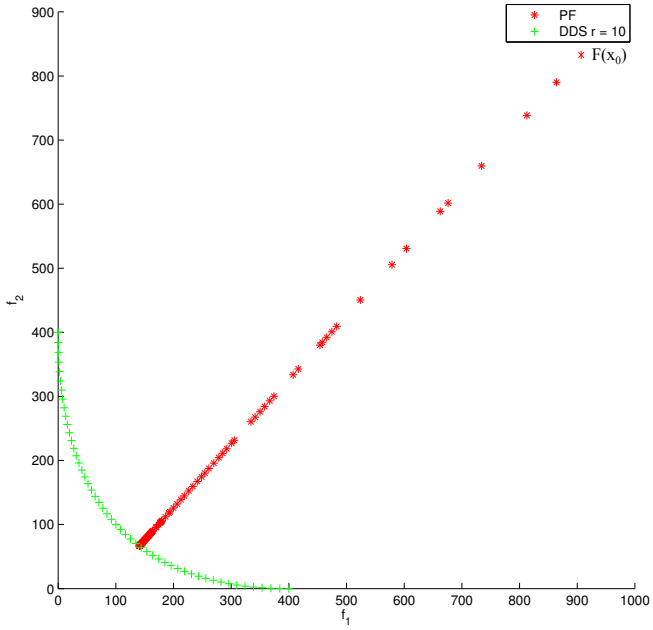


Figure 3.5: Results for the DDS method with $r = 10$.

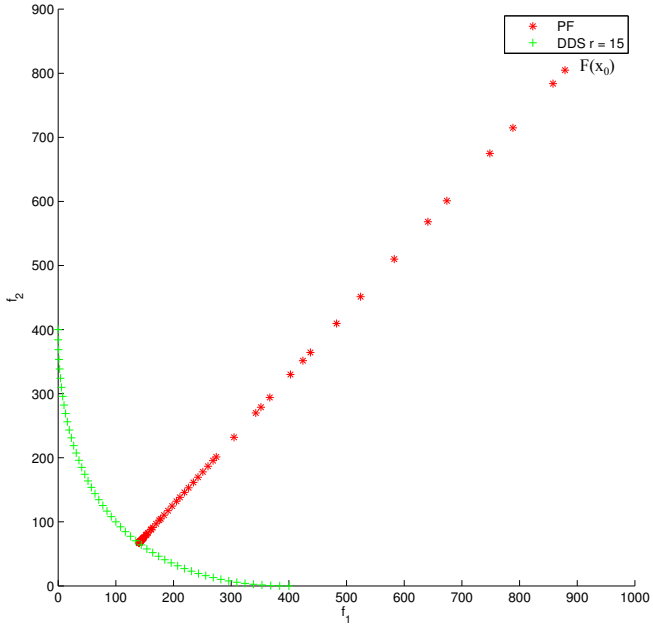


Figure 3.6: Results for the DDS method with $r = 15$.

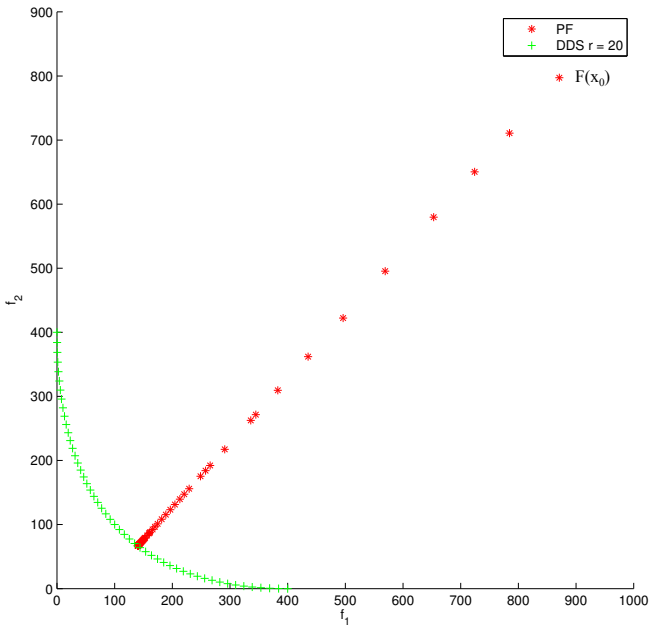


Figure 3.7: Results for the DDS method with $r = 20$.

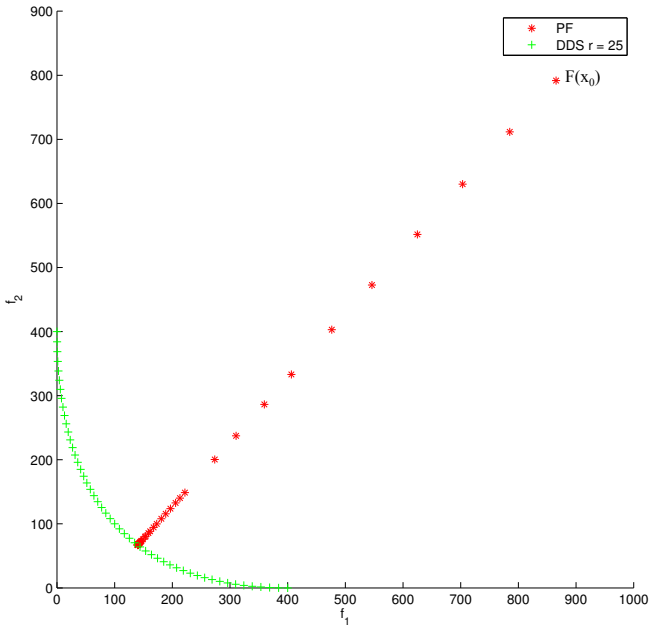


Figure 3.8: Results for the DDS method with $r = 25$.

Chapter 4

Integrating the DDS into MOEAs

As mentioned in Chapter 2, the main idea behind the memetic algorithms (MA) is to try to improve the approximations obtained by the MOEA by the use of local search techniques. To make an approximation of the PF that is well distributed along the front certainly is not an easy task. Generally, the function evaluations required by an average MOEA significantly increases as the problem becomes complicated.

It is important to remark that in general to find the right balance between the local search technique and the evolutionary operators of a memetic algorithm is not a trivial task . Hence, an "incorrect" choice of the design parameters could lead to worse results than the ones obtained by the MOEA used as a standalone algorithm.

4.1 Design parameters of the MAs

Along with the common parameters of a MOEA, there exist other design parameters in MAs that must be considered, e.g. which individuals must be selected in order to be improved by the local search operator. If the local search is applied too early in the MOEA, it is possible that a waste of resources is presented because some of the individuals selected for the local search can not pass the selection mechanism and they will be discarded in the next generation of the algorithm.

Figure 4.1 shows the effect explained above. Having a randomly selected individual I_1, a local search technique is performed and the I_1' individual is created by it. Considering that the individuals are selected by Pareto dominance and only five individuals are chosen from the whole population, it is shown that only the set of individuals that exist inside the region B will be selected in the next generation. Here, the I_1' do not accomplish the minimum requirements to survive the selection and the local search can be considered as a waste of function evaluations, and even a waste of computational resources.

Thus, the ideal behavior of an individual that was affected by the local search is

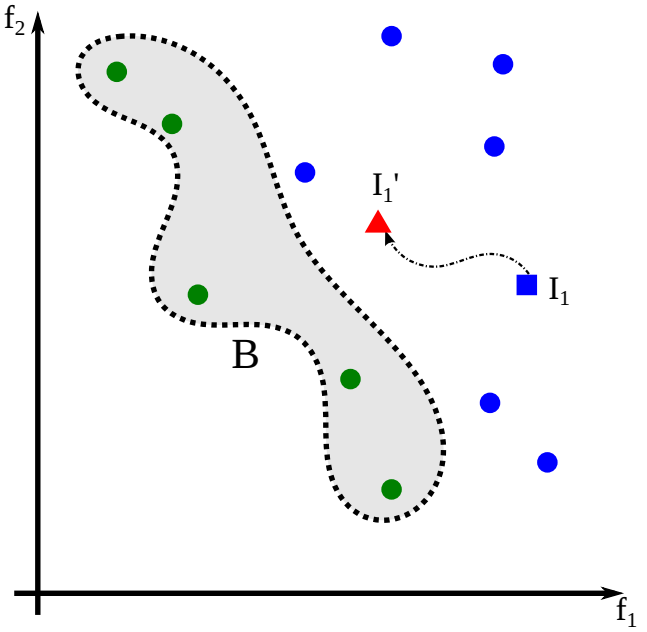


Figure 4.1: Worst case scenario for the application of a local search technique.

that it survives the selection in order to generate children in the following generation. Hence, if the local search leads to a solution near to the PF, there is a great probability that the children generated exist in the neighborhood of the original solution. Based on this behavior, it is expected that the convergence rate will be significantly increased due to the local search. Another possible advantage that comes from a local search operator is shown when the individuals are near to the PF. At this point, the convergence rate becomes significantly slower for the evolutionary operators in comparison with the individuals that exist far away from the PF. In this case, a local search technique can be used in order to reach some space that cannot be easily reached by the evolutionary operators.

In Figure 4.2 the individual A is affected by the local search operator leading to the individual A'. Since almost all the population resides near the PF, the improvement that comes from the evolutionary operators can be smaller. In the figure, the individuals B and C are the ones generated by these operators. On the other hand the individuals B' and C' are the ones generated from A' and it is expected that these solutions dominate both A and B. The behavior described above can only take place if the cost of the local search operator is significantly less than the one from the main evolutionary operators. At this point the main goal of the DDS is to take advantage of the information that already exists in the population.

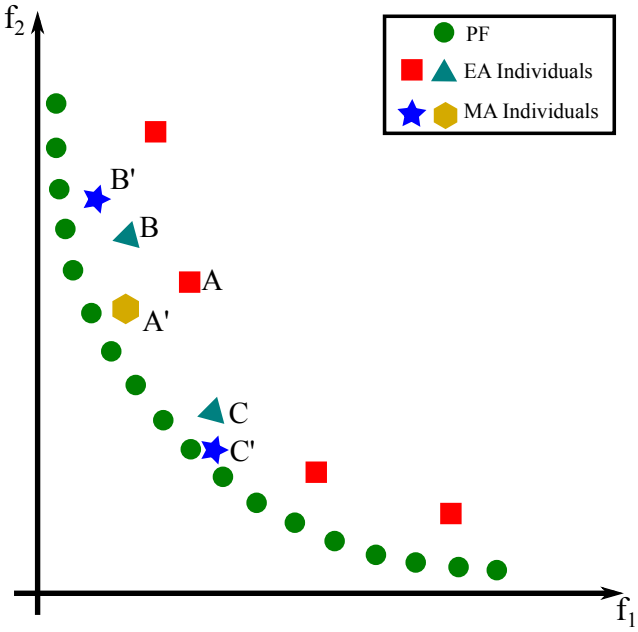


Figure 4.2: Local search improves the solutions near the PF.

4.2 DDS method as local searcher within a MOEA.

In some of the state-of-the-art algorithms, gradient information is used in order to perform a descent local search. The individuals obtained in this type of local search are expected to exist inside the descent cones of the functions [Lara et al., 2010], [Brown and Smith, 2005]. The descent cone represents the set of directions in which an improvement according to all objectives is possible.

It is known that the size of the descent cone depends on the directions that the objectives gradients are pointed at. When a solution x_0 is far enough from the PF, it is assumed that the gradients are almost aligned and the descent cone is almost equal to a half space. But, as long as the solution is getting closer to the PF, the gradients of the objectives are pointing in different directions and the cone starts reducing its size.

As an important feature to consider is that the DS/DDS can not only perform a movement following the descent cone. Instead, these methods can perform a movement in any of the 2^k directional cones by only changing the components of direction d. Figure 4.3 shows that changing the sign of one or more components of the direction vector d leads to the directions d_i, $i = 1, \ldots, 4$. Using these directions it is possible for the DS/DDS method to visit any of the directional cones by calculating the direction in parameter space ν_i, $i = 1, \cdots, 4$.

The next issue to take into consideration is which individuals of the population can be used to construct the matrix $\mathcal{F}(x)$. As mentioned in Chapter 3 such solutions

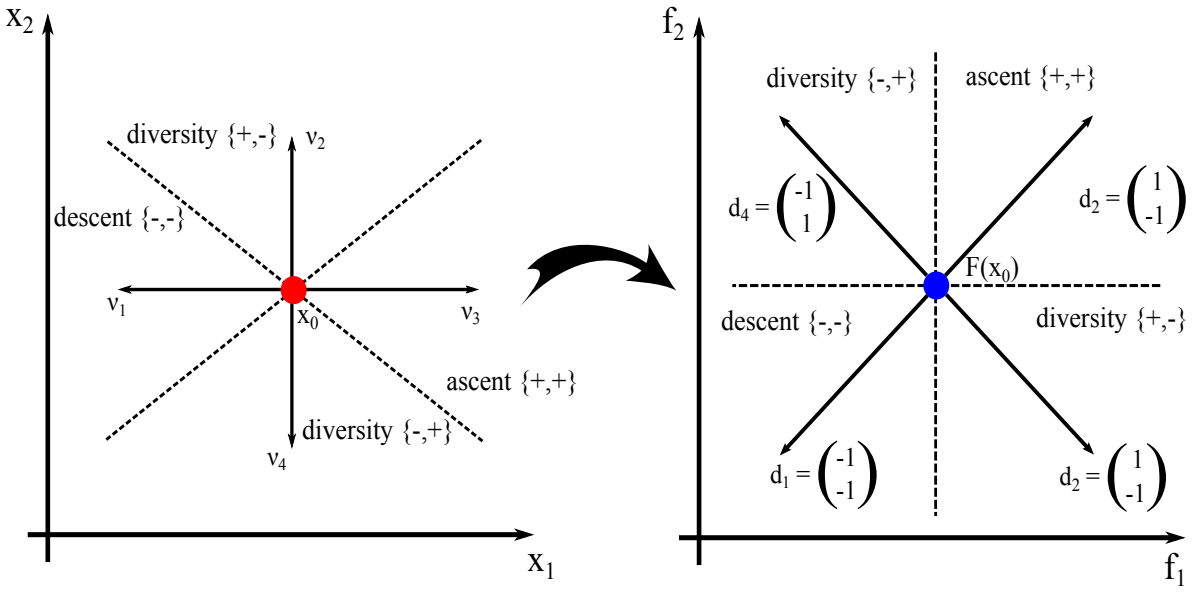


Figure 4.3: Relation between the descent cones and the direction d.

must accomplish certain conditions, in order to consider them as good candidates for the DDS method. The selection of the individuals that are going to take part in the DDS is not a trivial task and generally depends on the problem and the MOEA under consideration.

It is necessary to pay attention to the selection step of the algorithm because if random solutions are taken from the population perhaps a bad approximation of the gradient will be constructed and hence the DDS method can not obtain a solution as "good" as expected.

At this point it is possible to compare the approximation obtained with the DDS method with respect to the finite differences method. Here, the approximation of the finite differences approach by the use of information that is already present in the population is not possible, i.e., using a biobjective MOP with $n = 30$, it would be necessary to take around 60 individuals of the population to generate a "free" approximation of the Jacobian, but for the DDS, having $r = 2$ would be the minimal condition to generate a search direction ν.

The DDS method can save even more function evaluations by using two possible options. In Chapter 3, in page 28, it was mentioned that the step size uses a corrector step in order to follow the direction d. But, in case of using an EA, it is not strictly necessary to perform such a movement because it is not critical for the local operator to perform a restricted line movement, so it is possible to afford a small error in the direction. By this, the value of β is increased such that an admissible error can be used, e.g. $\beta = 0.5$, and also the correction step was eliminated in the implementation

of the MA.

4.3 NSGA-II/DDS

NSGA-II (see Chapter 2 in page 14), is a widely used algorithm to solve MOPs. The main advantage in taking this algorithm as a base line for our MA, is that in the NSGA-II it is possible to know for "free" which individuals of the population are non dominated at each generation.

Another point in favor of the NSGA-II is that it creates many sets of individuals at each generation. Each set of individuals is constructed by applying the evolutionary operators over the current population. At the end all the sets are coupled together before the selection mechanism is applied in order to generate the next generation of individuals. The main idea for the MA is to generate a new subpopulation L_i and merge it with the other subpopulations as described above. Next, because all subpopulations are merged into a single population, the selection mechanism is applied to all individuals no matter how many subpopulations were created in previous steps. It is desirable that the local search is applied before the evolutionary operators have taken place. This process is described in Algorithm 7.

Algorithm 7 Pseudocode of the NSGA-II/DDS

Input: number of generations G, number of individuals N
Output: final population P_G

1: Randomly generate the population P_0
2: Calculate $F(P_0)$
3: Apply genetic operators in P_0 to generate Q_0
4: Set $R_0 = \emptyset$
5: $i = 0$
6: **while** $i < G$ **do**
7: Apply the DDS on R_i and generate L_i
8: Set $R_i = P_i \cup Q_i \cup L_i$
9: Calculate the rank value of R_i
10: Calculate the crowding distance of R_i
11: Select the N individuals with the lowest rank and highest crowding distance (P_i).
12: Apply genetic operators in P_i to generate Q_i
13: Set $i = i + 1$
14: **end while**

Finally when the proper time to apply the local search inside the MA is already defined, the selection of the individuals to apply the local search is the next step in the

analysis. As mentioned before, the NSGA-II sorts the whole population into so-called "fronts" using the rank value of each individual, so it can be an option to take the individuals with the lowest rank value among all the population. Such individuals have a high probability to survive the selection process and it can be assumed that this individuals will attract the rest of the population, leading to an increment in the convergence rate.

4.3.1 Numerical Results for NSGA-II/DDS

As discussed before the application of the local search operator also depends on the kind of problem that is going to be solved. In this section the ZDT1 problem [Zitzler et al., 2000] was used in order to test the new memetic algorithm. The main goal of this experiment is to observe how the individuals inside a MOEA are affected after local search is applied to some of them.

For this experiment the local search operator was applied to the individuals that have a rank value equal to 1. It is expected that the solutions with lower rank value can survive selection and, therefore, can pass the information through recombination.

Finally, the last important point to be resolved is the use of the neighborhood information. Hence, a simple approach can help in the selection of the individuals that exist near a certain solution x_0. For each solution x_0 choose elements x_i from the current population such that $\|x_i - x_0\|_2 \leq \epsilon$, where ϵ is a problem dependent threshold. Then, the individuals with the smallest norm are selected as the neighbors of x_0. Graphically in Figure 4.4 it can be observed that this method pretends to form a neighborhood N around the solution x_0. The parameters used for the experiment are described in Table 4.1. In order to maintain a fair experiment, a fixed number of evaluations was used for both algorithms. The goal for this experiment is to demonstrate the improvement of the rate of convergence, and thus, a small number of function evaluations was selected.

In the particular case of the ZDT1 problem, the NSGA-II can converge to the PF in less than 10000 function evaluations. So a 10% of the total number of function evaluations required to approximate the PF was used in order to observe the behavior of the algorithm while it has not yet reached the set of interest.

In order to approximate the same behavior that DDS presents as a standalone algorithm, the direction d is set with the same value as the one defined in the experiments performed in Chapter 3, i.e.:

$$d = (-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}) \tag{4.1}$$

In order to obtain a fair competition, the memetic algorithm and the NSGA-II standalone algorithm performed over 30 independent runs, using the same random

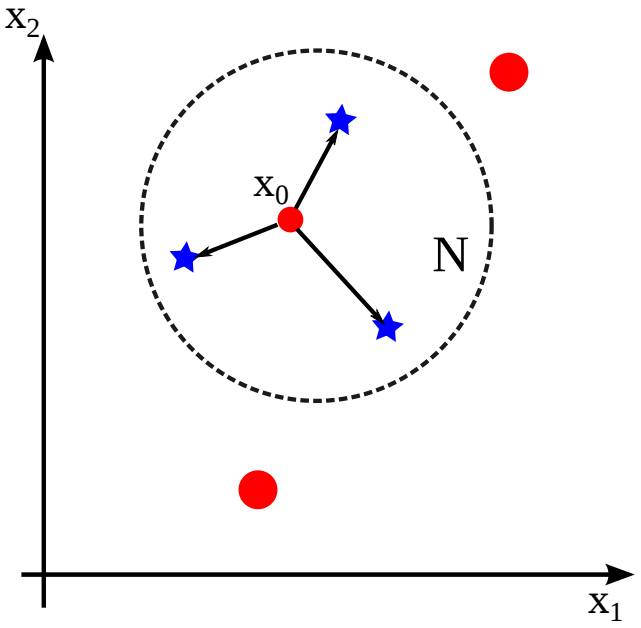


Figure 4.4: Neighborhood used in NSGA-II/DDS.

Identifier	**Value**	**Description**
n	30	Number of decision variables
N	100	Number of individuals
E_{max}	2000	Maximum number of function evaluations
r	7	Number of neighbors for DDS
P_c	0.95	Crossover probability
P_m	0.0333	Mutation probability
g_l	20	Frequency of generation to apply the DDS
N_l	3	Individuals affected by DDS

Table 4.1: Parameters for the NSGA-II /DDS

seed in both algorithms. Four metrics were measured to compare the performance of the algorithms: the Inverted Generational Distance (IGD), the Generational Distance (GD), the Δ_p indicator and finally the hypervolume.

The statistics can be seen in Table 4.2, and from the results it is clear that even when the NSGA-II has the minimum values in IGD and in the GD indicators, the memetic algorithm, in the mean, produces better results with respect to spread and

the convergence rate.

As the first experiment using the DDS inside a MOEA, DDS has demonstrated that at least for the ZDT1 problem it can increase the convergence rate of the algorithm. These results can be used in order to discuss an idea about the values of the main parameters of the MA, e.g. the size of the neighborhood and the number of individuals that must be affected.

Statistic	Algorithm	Indicators			
		IGD	GD	Δ_p	Hypervolume
Mean	NSGA-II	0.2277816835	0.1923195486	0.2326735132	0.9328004813
	NSGA-II/DDS	**0.1782101893**	**0.1594371667**	**0.1891265487**	**0.946121494**
Min	NSGA-II	0.0894245635	0.1065269724	**0.1091867267**	**0.8860861093**
	NSGA-II/DDS	**0.0885838994**	**0.0751261388**	0.1153974934	0.9071797267
Max	NSGA-II	**0.398520959**	**0.2801037669**	**0.398520959**	0.9764002544
	NSGA-II/DDS	0.2639838353	0.2769408556	0.2769408556	**0.9829203408**
Median	NSGA-II	0.2254322047	0.1909437101	0.2324421287	0.9354864752
	NSGA-II/DDS	**0.1718001804**	**0.1506737425**	**0.1757497566**	**0.9422848267**

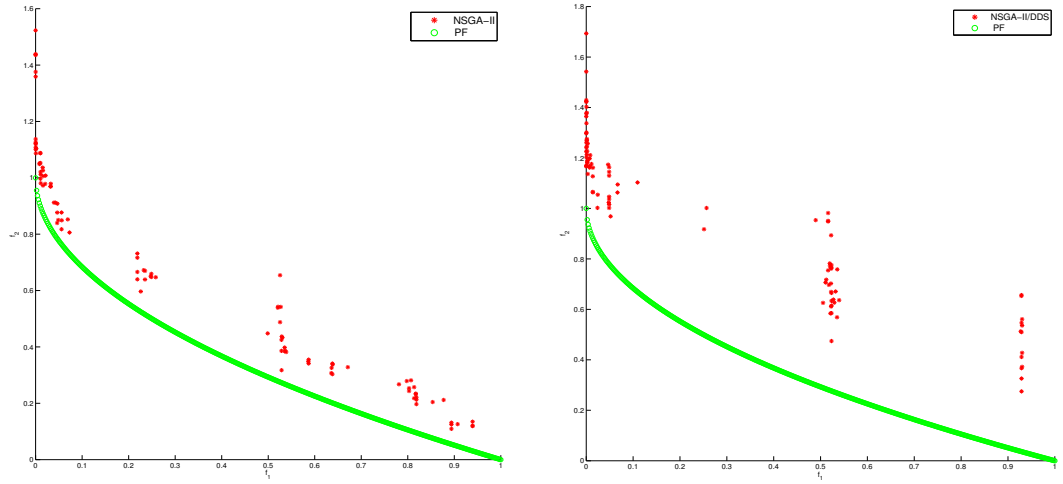
Table 4.2: Results for the NSGA-II /DDS

Figure 4.5 presents the best approximations obtained by the algorithms when the Δ_p indicator is measured. Here, it is important to mention that for this particular case, both approximations were obtained with the same seed value. Hence, graphically it is shown that the evolutionary operators win against the local search operator, such that the approximation obtained by the standalone algorithm has a better spread and better convergence rate than the one obtained by the MA. It is possible, however, that this behavior can be changed by fine tuning the parameters of the MA.

The worst cases are presented in Figure 4.6. Graphically it can be seen that the NSGA-II/DDS presents better convergence rate than that for the NSGA-II. In most of the experiments this type of behavior is repeated, and the MA wins in the average.

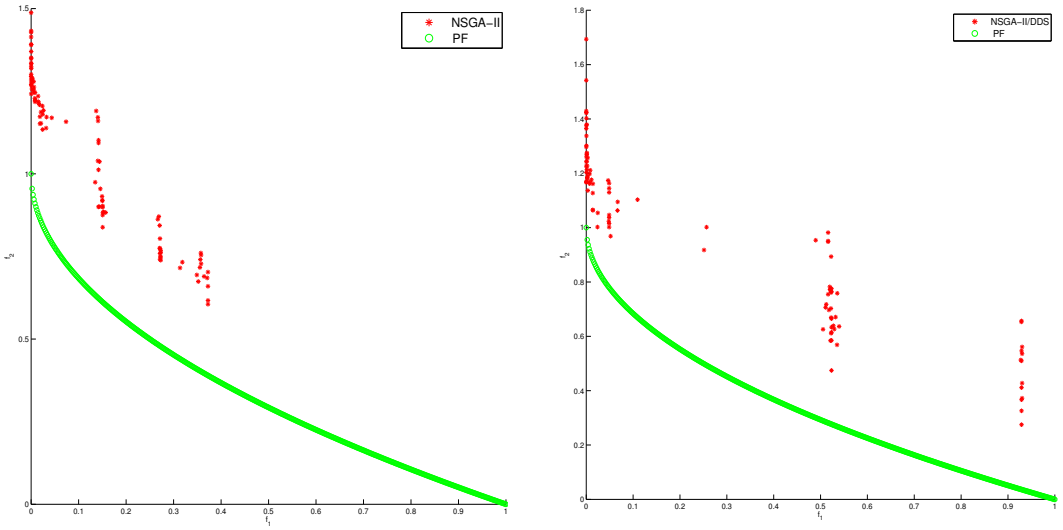
This experiment confirms that when there exists a proper choice of the parameters that control the memetic algorithm, the convergence rate is increased. The parame-

ters used in the NSGA-II can be used as a starting point for the implementation of other memetic algorithms.



(a) NSGA-II (b) NSGA-II/DDS

Figure 4.5: Best runs for the algorithms on the ZDT1 problem.



(a) NSGA-II (b) NSGA-II/DDS

Figure 4.6: Worst runs for the algorithms on ZDT1 function.

4.4 MOEA/D/DDS

After the CEC'09 competition, the MOEA/D algorithm has become one of the most important algorithms used to solve MOPs that exist in the state-of-the-art. As described in Section 2, it decomposes the MOP into several scalar subproblems and solves them simultaneously. Since the algorithm solves simultaneously all the subproblems, one can expect that by affecting a subset of solutions, in the long run, the other solutions that keep interactions with its neighbors can also be affected by interchanging information in order to improve themselves.

In MOEA/D the neighbors of one solution are explicitly stored in an array. This detail could be interesting to exploit because such information can be taken in order to integrate the DDS method. Further, this idea could help the selection of the neighborhood of the solution. It is important to remark that MOEA/D takes as neighbors the solutions that have weights close to the weight that is associated to the current individual.

In order to use the DDS algorithm a direction d is needed. In particular, for this method, a "reference" point was used. Here, the zero vector $\overrightarrow{0} \in \mathbb{R}^k$ was set as the reference point. Using this reference point the algorithm will try to spread the solutions when the MOEA/D is approaching to the PF. Hence, for each individual x_0 in the population it is possible to define the direction d using a reference point z:

$$d = z - \frac{F(x_0)}{\|F(x_0)\|_2} \tag{4.2}$$

The DDS was located before the evolutionary operators have taken place, in other words, the local search technique could only be applied to new individuals. In order to let the MOEA/D perform its own global search, the DDS technique was only applied after every g generations.

Unlike in the NSGA-II in the MOEA/D there does not exist a mechanism that remarks which individuals are non-dominated. So in this case the DDS is applied to individuals that are randomly selected. Algorithm 8 defines how the MA was developed. Here, the frequency k_{ls} defines in which generation the local search is going to be applied. Also, the parameter h_{ls} is presented. This parameter is used to restrict the local search to certain number of individuals in the population.

Algorithm 8 Pseudocode of the MOEA/D/DDS.

```
Set the weight vectors λ_i.
Set the neighborhoods B(i) = i_1, ..., i_T.
Generate initial population P_0.
Set reference point z.
Set the external population EP = ∅.
Set g = 1.
repeat
  for i = 1, ..., N do
    Select two indexes k, l from B(i).
    Generate y_i using genetic operators over x_k, x_l .
    Generate y'_i using the subproblem improvement heuristic on y_i .
    if mod(g, k_ls ) = 0 and mod(i, h_ls ) = 0 then
      Generate y''_i appliying DDS on y'_i.
      Set y'_i = y''_i.
    end if
    Update reference point z*.
    Remove from EP all the vectors dominated by y_I.
    Add y_i if no vectors in EP dominates it.
  end for
  Set g = g + 1
until Stopping criterion is satisfied
Report E_p
```

4.4.1 Numerical results

Since MOEAD/D is the selected algorithm for this experiment it is reasonable to test the MA with the problems included in the CEC'09 contest benchmark [Zhang et al., 2009b]. In these experiments an adapted version of MOEA/D proposed by [Zhang et al., 2009a] was used. This specific version uses a utility function π_i in order to distribute the subproblems.

The original DS algorithm was developed to work only with unconstrained problems. It is worth nothing, however, that there were some problems with some of the CEC'09 functions. The main problem consists in that sometimes after the DDS performed the iterations some of the values of x vector go outside of the allowable range and make the algorithm to reach some unfeasible results. So, it becomes necessary to implement some mechanism that bounds the limits of the variables.

In order to restrict the DDS to stay inside the bounds of the variables, a new stopping criterion was added to the original algorithm. When the generated solution overpasses the limits of the variables, the step size is reduced in order to fit such restrictions until all variables remains in its feasible region.

In these experiments the algorithm was run 30 times for each of the test problems. These runs were performed by using the Tchebycheff and NBI decompositions of the MOEA/D (30 iteration for each decomposition). Table 4.3 shows the parameters used in the runs. Here, in order to compare the results four metrics were used: hypervolume, Inverted Generational Distance (IGD), Generational Distance (GD) and Δ_p. In the MA, the local search was performed using the DDS method and also the DS method in order to compare both algorithms.

Identifier	**Value**	**Description**
n	30	Number of parameters
N	300	Number of subproblems
N_{EP}	100	Size of external population
G	50000	Max number of function evaluations
r	7	Number of neighbors for DDS
P_m	1/n	Mutation probability
g_l	15	Frequency of generation to apply the DDS
N_l	10	Individuals affected by DDS

Table 4.3: Parameters for the MOEA/D/DDS

In Tables 4.4, 4.5, 4.6 and 4.7, the average values of the indicators obtained after 30 runs that were obtained by applying the algorithm over the test functions UF1-UF7 are presented. Hence, the letters in brackets represent the decomposition used in the algorithm, e.g. TD refers to Tchebycheff decomposition.

It is visible that the results from the DDS are very promising. The memetic algorithm with the DDS presents better Δ_p results on four of the seven problems that were tested. Here, each of the gradients of the DS method uses the function evaluations defined in the automatic differentiation approach ($5k$).

For the particular case of the UF2 the MA presents a worse performance in comparison with the standalone MOEA/D because for this particular function the convergence of the algorithms in almost all the parts of the PF, takes place at the earliest generations (around $g = 100$). When the DDS method tries to point to the PF to a solution that is already "near", this can be considered as a waste of resources, so it would be interesting to find a method that allows the algorithm to detect the parts of the PF that really need to apply the local search technique in order to approximate

to the PF.

From Table 4.7 it is possible to conclude that the expected behavior for the DDS has been accomplished. Here, it is possible to affirm that the hypervolume value is incremented because the solutions slightly reduce their distance to the original PF. As mentioned above, the results of the local search techniques can be affected by the choice of reference point, so in a future work a direction d (in Equation (4.2)) that also helps the good spreading of the solutions can be stated.

In general from the numerical results of these experiments it is possible to see that the DDS increases the performance of the MOEA, but in some cases the increment in the convergence rate can lead to solutions that converge to a certain region of the PF, e.g in function UF2 and the DDS.

IGD				
Function	Algorithm			
	MOEA/D(NBI)	MOEA/D(TD)	MOEA/D/DS(TD)	MOEA/D/DDS(TD)
UF1	0.0572622262	0.0347917234	0.0260356595	**0.0143866214**
UF2	0.1252927650	0.0208602479	**0.0199435559**	0.0238102230
UF3	0.2634246836	0.0919392204	0.0813866009	**0.0287657808**
UF4	**0.0764105874**	0.0787618084	0.080434227	0.1242333148
UF5	**0.7158476404**	0.7553129089	0.7842737725	0.7279528880
UF6	0.4768850725	0.2886370163	0.2989744918	**0.2778251599**
UF7	0.1235368942	0.0267808352	0.0275765137	**0.0129843756**

Table 4.4: Results for the IGD indicator in problems UF1-UF7.

GD				
Function	Algorithm			
	MOEA/D(NBI)	MOEA/D(TD)	MOEA/D/DS(TD)	MOEA/D/DDS(TD)
UF1	0.0138066143	0.0304481000	0.0237297276	**0.0123029218**
UF2	**0.0021363890**	0.0155243552	0.0114669723	0.0100961147
UF3	**0.0255686142**	0.0785433072	0.0636126413	0.0282090741
UF4	0.0747297508	0.0838012798	0.0851082982	**0.0253206580**
UF5	**0.6012714682**	0.8435331768	0.8807595923	0.7913894440
UF6	**0.0782684283**	0.1374295793	0.0.1402271371	0.1095751695
UF7	**0.0076582949**	0.0190092861	0.0141336054	0.0094960721

Table 4.5: Results for the GD indicator in problems UF1-UF7.

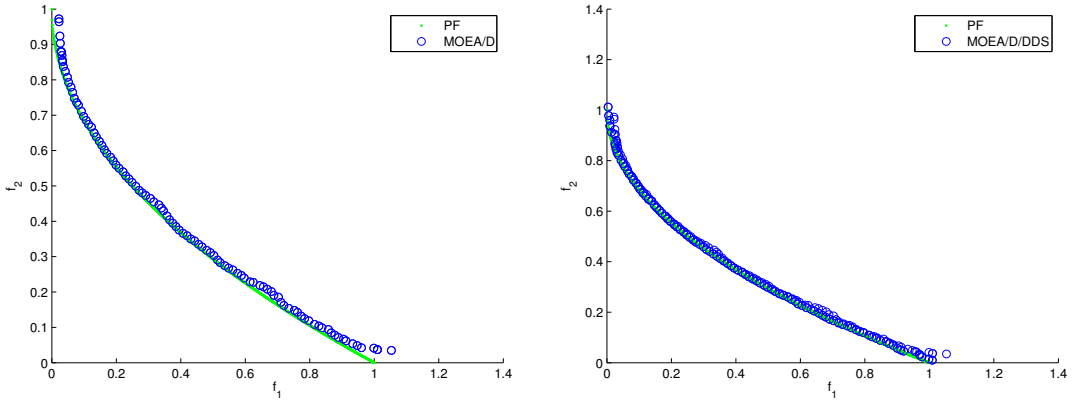
Δ_p				
Function	Algorithm			
	MOEA/D(NBI)	MOEA/D(TD)	MOEA/D/DS(TD)	MOEA/D/DDS(TD)
UF1	0.0572622262	0.0352910483	0.0267699625	**0.0145801900**
UF2	0.1252927650	0.0208602479	**0.0199435559**	0.0238102230
UF3	0.2634246836	0.0960016721	0.0854409444	**0.0321980680**
UF4	**0.0767216658**	0.0838012798	0.0851082982	0.1246050193
UF5	**0.7339885104**	0.8445128539	0.8807595923	0.8014984378
UF6	0.4802285764	0.3055477151	0.314770258	**0.2970146178**
UF7	0.1235368942	0.0276249002	0.0283612276	**0.0131369537**

Table 4.6: Results for the Δ_p indicator in problems UF1-UF7.

Hypervolume				
Function	Algorithm			
	MOEA/D(NBI)	MOEA/D(TD)	MOEA/D/DS(TD)	MOEA/D/DDS(TD)
UF1	0.9339361501	0.9538326792	0.9599056217	0.**9757925643**
UF2	0.8915056434	0.9739430907	**0.9741609691**	0.9692053716
UF3	0.6832457349	0.8721615163	0.8825773787	**0.9642096892**
UF4	0.8770614516	**0.8804023001**	0.8782569225	0.8750131924
UF5	0.7542892280	**0.7587095853**	0.7540902708	0.7586473790
UF6	0.5912642076	0.6846642933	0.6721417613	**0.6968876079**
UF7	0.8968660617	0.9712112769	0.972680329	**0.9818931461**

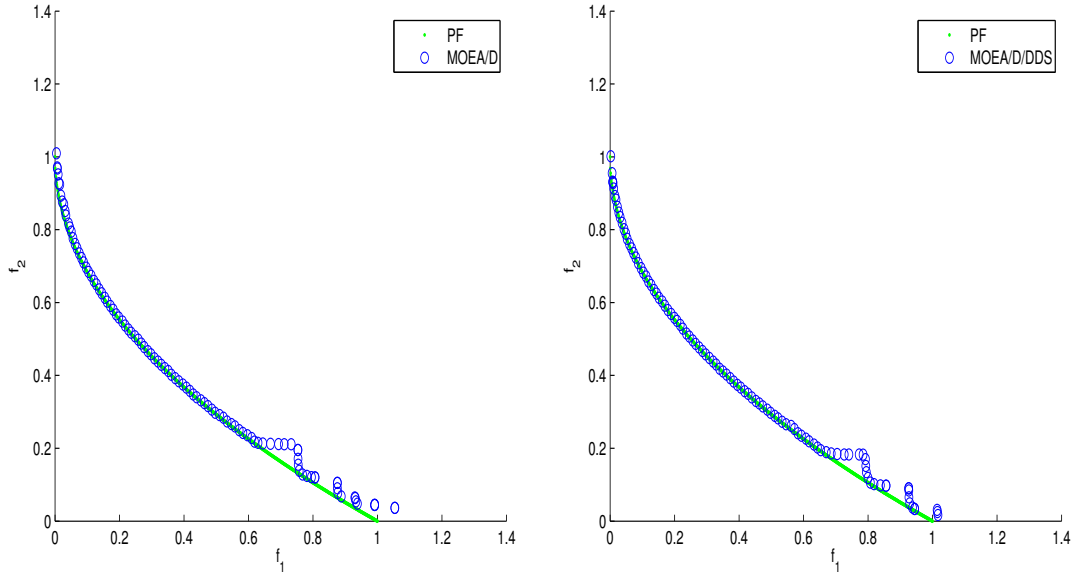
Table 4.7: Results for the Hypervolume indicator in problems UF1-UF7.

Figures 4.7 to 4.13 present the best approximations obtained by the MOEA/D/DDS and the MOEA/D. Graphically it can be observed the difference between the algorithms is not significant in UF2 and UF5, where the memetic algorithm present lower values for the indicators. Hence, in the best cases for the function UF4 the DDS helps the algorithm to get closer to the PF in comparison to the standalone algorithm.



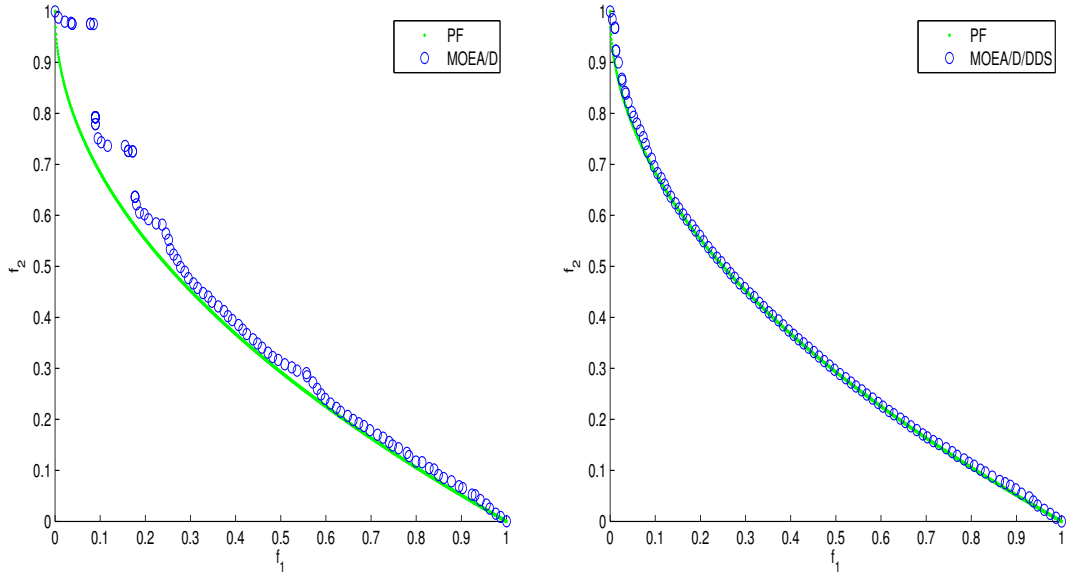
a)MOEA/D b)MOEA/D/DDS

Figure 4.7: Graphical results in problem UF1.



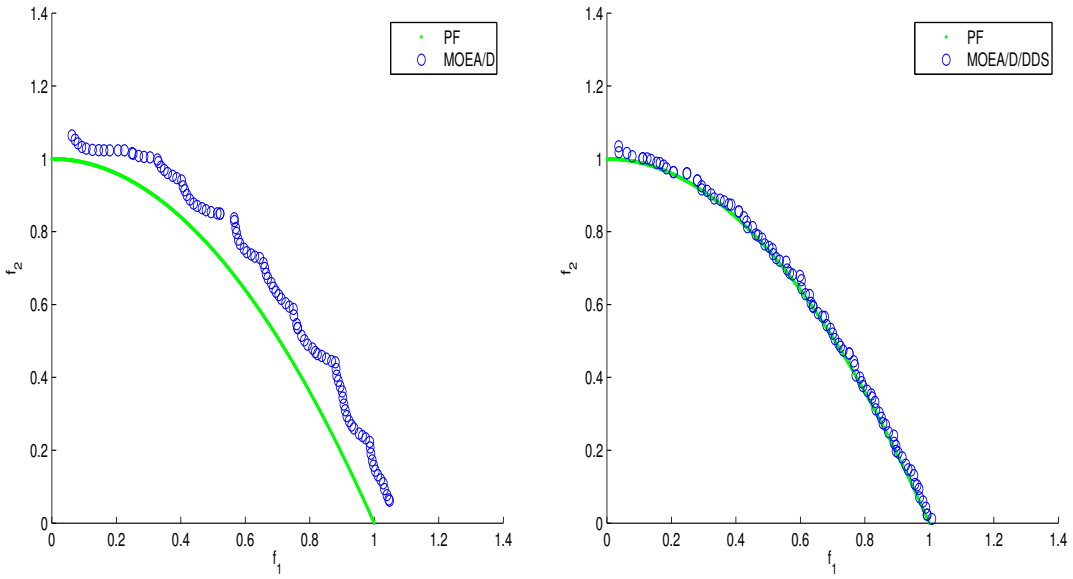
a)MOEA/D b)MOEA/D/DDS

Figure 4.8: Graphical results in problem UF2.



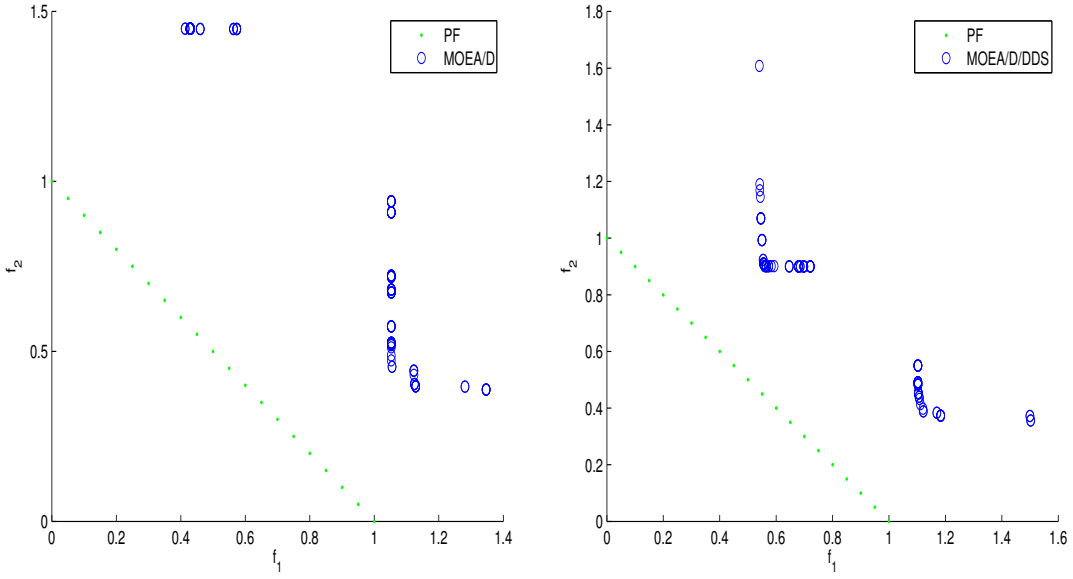
a)MOEA/D b)MOEA/D/DDS

Figure 4.9: Graphical results in problem UF3.



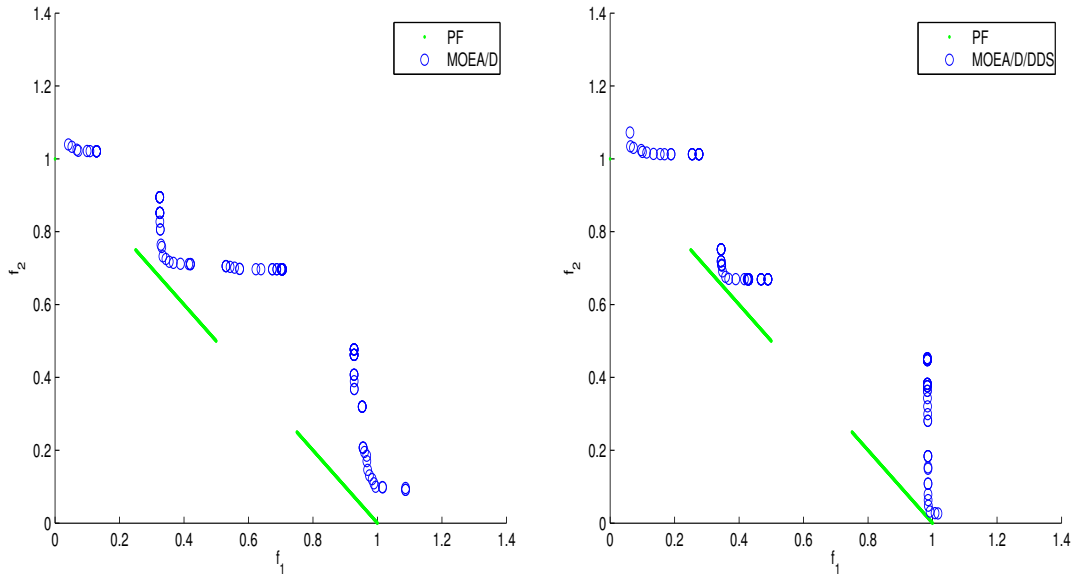
a)MOEA/D b)MOEA/D/DDS

Figure 4.10: Graphical results in problem UF4.



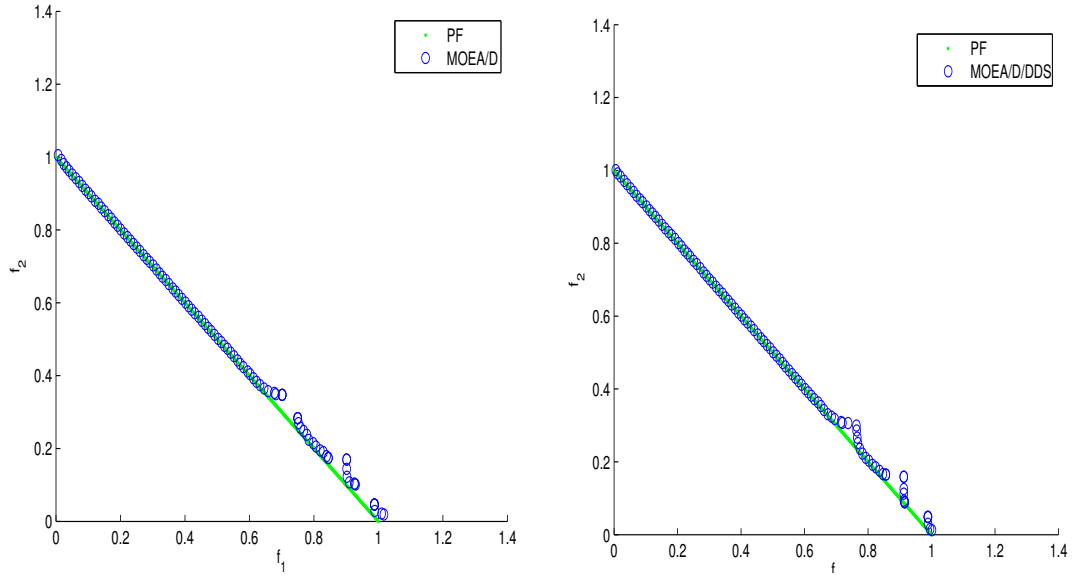
a)MOEA/D b)MOEA/D/DDS

Figure 4.11: Graphical results in problem UF5.



a)MOEA/D b)MOEA/D/DDS

Figure 4.12: Graphical results in problem UF6.



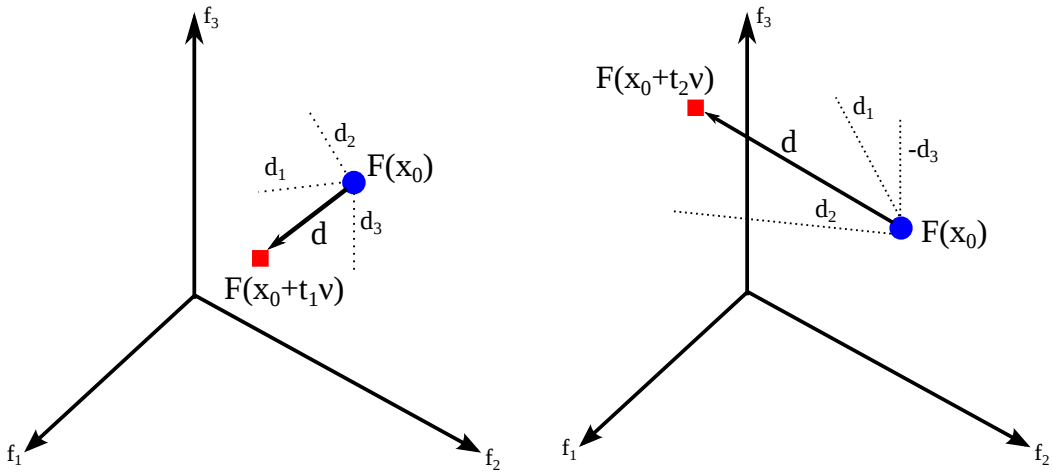
a)MOEA/D b)MOEA/D/DDS

Figure 4.13: Graphical results in problem UF7.

4.5 Decomposing the direction of the DDS

When the experiments with the MOEA/D were performed over the functions UF8-UF10, the DDS algorithm did not present the results such that were expected as in the previous section. Having the current solution in objective space $F(x_0)$ and the objective value of the solution calculated by the local search $F(x_s)$, the 2-norm of the difference between these two points was minimal (around 10^{-9}) even when the solutions are "far" from the PF.

After a step by step review over the original MA it was found that in the DDS the improvement of the method was restricted because in most cases the step size of the algorithm was reduced to a insignificant value ($t \approx 10^{-12}$). The main issue behind the reduction of the step size was caused because one of the three direction points in the opposite direction when the value of t is "normal" and by this the angle β (see Chapter 3 in page 28) overpasses its threshold. Figure 4.14 illustrates the co-relation between the sign of the components of the direction d and the length of the step size. In Figure 4.14a a step size t_1 (that can be considered small) is used to calculated a new candidate solution that points in the same direction than d. Meanwhile, in Figure 4.14b when a larger step size t_2 is used ($t_2 \gg t_1$), the new solution presents the component d_3 in the opposite direction than d.



(a) t_1 (b) t_2

Figure 4.14: Problem that occurs when using a large step size.

From the discussion above it was shown that it is preferable to take only two of the three components of the direction d and use it to direct the movement in order to help the DDS method to perform larger steps. But *apriori* in the end the local search must find an improvement in all the objectives and not only in a subset of them.

The main goal of this approach is to generate many candidate solutions that steer in a different subspace of direction d. Finally, all the candidate solutions are merged in order to calculate a final solution, that increments the performance of the original solution.

As explained above, many solutions are generated in the firsts steps of this algorithm, and it is assumed that each of these solutions improves a subset of the objectives functions, but at the end the algorithm must combine these solution into a single one. In order to perform such an operation, the mutation operator proposed in the Differential Evolution (DE) algorithm [Storn and Price, 1997] was used.

In the DE mutation operator the weighted difference between two vectors is taken and added to a third vector. Figure 4.15 illustrates an example in a two dimensional problem. Here, the difference of the x_2 and x_3 vectors helps x_1 to get closer to the optimal solution x^*.

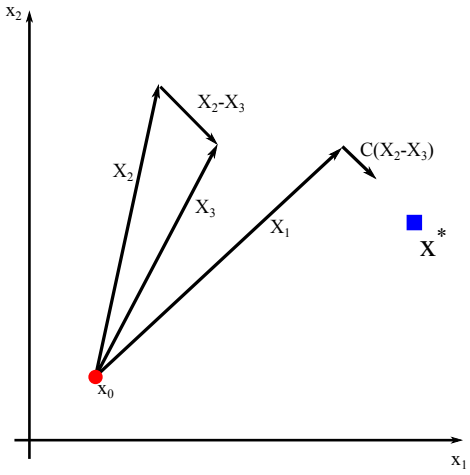


Figure 4.15: Example of the DE mutation operator for a bi-objective problem.

Having $x_j, j = 1, \cdots, 3 \in \mathbb{R}^n$, then a new candidate solution can be defined as:

$$v = x_1 + C(x_2 - x_3), \tag{4.3}$$

where $C \in [0, 2]$ is a constant that controls the amplification rate of the differential variation.

The first step to generate the three solutions required by the mutation operator,

is that the function F is decomposed into three different directions such that:

$$\begin{aligned} F &= f_{d1} \cup f_{d2} \cup f_{d3} \\ &= \begin{pmatrix} f_i \\ \cdots \\ f_{k1} \end{pmatrix} \cup \begin{pmatrix} f_j \\ \cdots \\ f_{k2} \end{pmatrix} \cup \begin{pmatrix} f_k \\ \cdots \\ f_{k3} \end{pmatrix}, \end{aligned} \tag{4.4}$$

where f_j represent the j entries of the function F.

Since the objective function was decomposed, the direction d used in the DS algorithm is also separated into three new directions adopting the same j components used for the decomposition of the objective function. Here, the new directions are defined as:

$$\begin{aligned} d &= d_{d1} \cup d_{d2} \cup d_{d3} \\ &= \begin{pmatrix} d_i \\ \cdots \\ d_{k1} \end{pmatrix} \cup \begin{pmatrix} d_j \\ \cdots \\ d_{k2} \end{pmatrix} \cup \begin{pmatrix} d_k \\ \cdots \\ d_{k3} \end{pmatrix}. \end{aligned} \tag{4.5}$$

Having the decomposed directions, DDS can perform a steered movement (as described in 3) in each of the new directions. Finally, after the DDS has completed the line search, three solutions are generated $x_{dj}, j = 1 \cdots, 3$ and these solutions are merged:

$$x_{i+1} = x_{d1} + C(x_{d2} - x_{d3}). \tag{4.6}$$

Using this new method the algorithm can performs a line search using "larger" steps, and in the objective value $F(x_{i+1})$ of the new calculated solution is expected to exists in the neighborhood of $F(x_{dj})$. In Figure 4.16 it is shown that the new candidate solution x_{i+1} is located inside the neighborhoods of the three solutions x_{dj}. Here, the dotted circles represent the neighborhoods of each solution and the shaded area symbolizes the neighbor where the new candidate solution is expected.

The main advantage of decomposing the direction d is that even when the value k is increased the direction of the DS can be decomposed into three subspace d_{dj}. Further, in order to calculate the solutions x_{dj} it is only needed a single approximation of the gradient. Algorithm 10 presents the pseudocode of the decomposition DDS.

Algorithm 9 Pseudocode of the decomposition DDS.

1: Calculate f_{dj} as in (4.4).
2: Calculate d_{dj} as in (4.5).
3: Generate x_{dj} by applying the DDS.
4: Generate x_{i+1} as in (4.6).

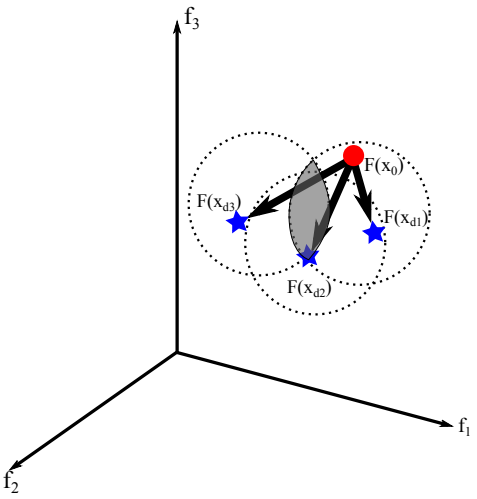


Figure 4.16: Candidate solution expected in the neighborhood of the solutions x_{dj}.

4.5.1 Numerical results for the decomposition DDS

In order to continue the previous experiments, the decomposition algorithm was tested in the three objectives problems of the CEC'09 benchmark. Here, the algorithm was run 30 times for each of the test problems, using two different decompositions, i.e. Tchebycheff and NBI decompositions. Table 4.8 presents the parameters used in each of the run of the algorithm.

Identifier	Value	Description
n	30	Number of decision variables
N	300	Number of subproblems
N_{EP}	100	Size of external population
G	50000	Max number of function evaluations
r	7	Number of neighbors for DDS
P_m	1/n	Mutation probability
g_l	15	Frecuency of generation to apply the DDS
N_l	10	Individuals affected by DDS
C	0.7	Constant for DE mutation operator

Table 4.8: Parameters for the decomposition MOEA/D/DDS

The MOEA/D/DDS with a decomposed direction shows an increases in the performance with respect to the original algorithm. The results from Tables 4.9 to 4.12, show that the new algorithm increments the Δ_p indicator in two of the three test functions. As in the experiments of the previous section, the cost of the gradient approximation in the DS was set in $5k$. All the MAs obtain an increment in the convergence rate and towards the end.

IGD			
Algorithm	Problems		
	UF8	UF9	UF10
MOEAD(NBI)	0.0905479809	0.2204627194	1.0931730426
MOEAD(TD)	0.1030686364	0.1418144915	1.565850743
MOEAD/DS(TD)	0.1049478908	0.1507459676	1.6558603659
MOEAD/DDS(TD)	0.1011438193	**0.1338730213**	1.2965367009
MOEAD/DS(NBI)	**0.0848439888**	0.2490652369	1.0024615247
MOEAD/DDS(NBI)	0.0876753754	0.2350480389	**0.9753865032**

Table 4.9: Results for the IGD indicator for problems UF8-UF10.

Figures 4.17 to 4.19 illustrate the comparison among the best solutions found with the Tchebycheff decomposition. From the pictures it is clear that even when the DDS shows a good approximation in problem UF9, in the average cases the solutions were attracted to the reference point z that for this particular experiment was chosen as the zero vector. Hence, all the solutions are concentrated in the middle part of the objective space, and the approximation loses its spread along the PF.

GD			
Algorithm	Problems		
	UF8	UF9	UF10
MOEAD(NBI)	**0.0516827256**	0.0807498948	1.2717375658
MOEAD(TD)	0.3805004696	0.7176044379	2.5226477893
MOEAD/DS(TD)	0.4853651901	0.8267701070	2.6729793238
MOEAD/DDS(TD)	0.2977168144	0.6800938809	2.4076450278
MOEAD/DDS(NBI)	0.0462688125	**0.0710153193**	1.083548091
MOEAD/DDS(NBI)	0.0475805638	0.0760597811	**1.0272677578**

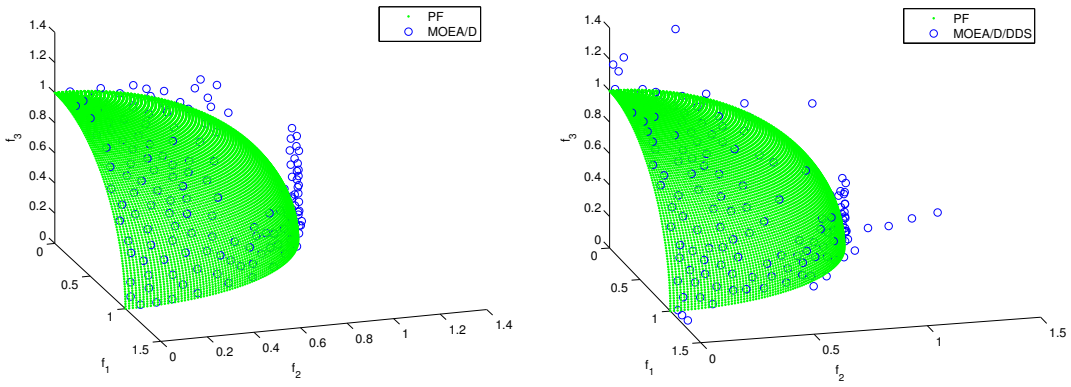
Table 4.10: Results for the GD indicator for problems UF8-UF10.

Δ_p			
Algorithm	Problems		
	UF8	UF9	UF10
MOEAD(NBI)	0.0910747238	**0.2204627194**	1.2742476511
MOEAD(TD)	0.3835674553	0.7176044379	2.5226477893
MOEAD/DS(TD)	0.4859597077	0.8267701070	2.6729793238
MOEAD/DDS(TD)	0.3043166328	0.6800938809	2.4076450278
MOEAD/DDS(NBI)	**0.0850173032**	0.2490652369	1.1169212468
MOEAD/DDS(NBI)	0.0876753754	0.2350480389	**1.0586193275**

Table 4.11: Results for the Δ_p indicator for problems UF8-UF10.

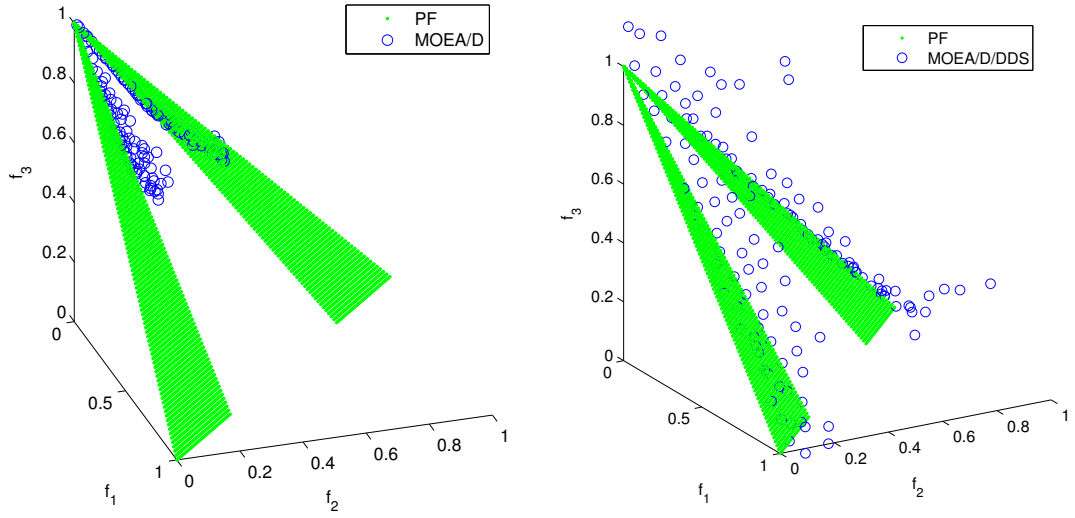
Hypervolume			
Algorithm	**Problems**		
	UF8	UF9	UF10
MOEAD(NBI)	0.996156917	0.9767867371	0.8571690877
MOEAD(TD)	0.9983952876	0.9968842351	0.8820561865
MOEAD/DS(TD)	0.9983183730	0.9968600779	0.8758373866
MOEAD/DDS(TD)	**0.9984354931**	0.9971625358	**0.8838777544**
MOEAD/DDS(NBI)	0.9964002215	0.9721411587	0.8633398313
MOEAD/DDS(NBI)	0.9962463045	**0.9748012254**	0.8584407906

Table 4.12: Results for the hypervolume indicator for problems UF8-UF10.



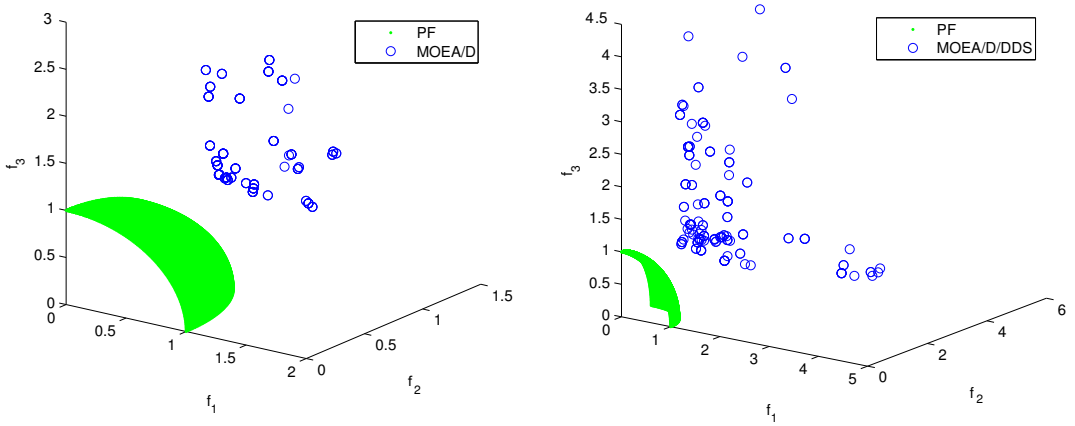
a)MOEA/D b)MOEA/D/DDS

Figure 4.17: Graphical results in problem UF8.



a)MOEA/D b)MOEA/D/DDS

Figure 4.18: Graphical results in problem UF9.



a)MOEA/D b)MOEA/D/DDS

Figure 4.19: Graphical results in problem UF10.

Chapter 5

Integration of DS into Multiobjective Control Problems

As mentioned before the movement in the objective space that the DS can perform is not restricted to only the descent directions of the objective functions. Sometimes in multiobjective control problems (MOCPs) this feature can be used in order to develop a controller that helps the system to move from a certain region in objective space to another, in order to adjust the parameters in the system.

The DS also can be used to follow a certain path in the conditions of the system, e.g. make the system to preserve certain trajectory in objective space such that it does not overpass any of the restrictions of the system. In this section, the features described above are used in two possible applications of the DS into some particular MOCP.

5.1 Application of DS to deteriorated MOCP

Control processes are generally affected by external parameters that deteriorates the parameters of the system, leading to values that get worse as time goes by. Perhaps the most promising idea is to replace the deteriorated parts of the model such that the system returns to an "ideal" state, but in the real world this could lead to a waste of money and, in some cases, a waste of time as well.

Here, the incorporation of a controller procedure could not provide the same results as in the ideal case, but it could at least approximate those results, with the advantage that neither of the parts are replaced.

In this section, an alternative to the replacement of the parts is presented togheter with a the proposal that uses the DS method as a controller algorithm inside a MOCP. The main idea of these experiments resides in demonstrating that the DS can actu-

ally respond to the effect of deterioration of certain parameters. Also, the DS must demonstrate that is possible to return the objective values almost to their original values with a low computational cost.

Figure 5.1 presents a general active control schema used in order to manipulate a multiobjective control problem, which was presented by Avigad and Eisenstadt [Avigad and Eisenstadt, 2010]. Having two parameters $x, y \in \mathbb{R}$, the controller measures the error $e = \|F^*(x) - F(x)\|$ where F^* represents the optimal value that the objectives take before the deterioration is applied, and F represents the current value of the objectives. The controller can only adjust the values of y in the system while the values of x are deteriorated in time.

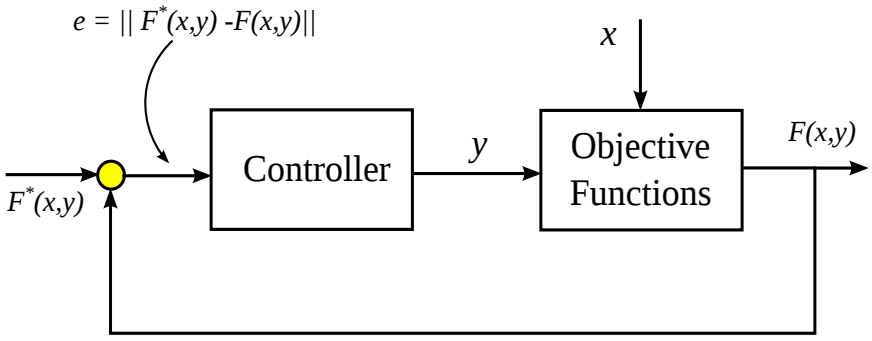


Figure 5.1: Active control scheme.

For this system it is possible to define that the main idea behind the deterioration problems is to minimize the error through the time. Further, the error can be defined as:

$$e = \int_0^t \|F^*(x) - F(x_0)\| \tag{5.1}$$

Having y^0 which represents the value of y after the deterioration, Avigad and Eisenstadt proposed a proportional controller such that at each time the value of y is defined as:

$$y = y^0 + k_c e \tag{5.2}$$

where e is defined as in Equation (5.1), and k_c is the proportional controller gain.

In order to calculate the k_c value, Avigad and Eisenstadt proposed to use the NSGA-II. It is clear that some computational resources can be saved by directly

reaching the point in the objective space that has the least error e.

As mentioned before the main task that the DS must resolve is to control the variables in the system in order to stay as close as possible from the "optimal" condition that was present before deterioration starts. In terms of time, each period T some of the variables are deteriorated by a certain rate. This happens when the DS must manipulate the other variables of the system in oder to minimize the error of the Equation (5.1). Here, some modifications to the original algorithm of the DS must be performed in order to work with this type of problems.

Having an initial optimal solution x_0 the direction ν is calculated as in Equation (2.17) in page 19, but in particular the j entries of ν are set to zero, in order to simulate the behavior of the deteriorated variables. Then the system can be reformulated as:

$$\nu = \begin{pmatrix} \nu_1 \\ \nu_2 \\ 0 \\ 0 \\ \dots \\ \nu_n \end{pmatrix} = \begin{pmatrix} J_{1,1}\ J_{1,2}\ 0\ 0\ \cdots J_{1,n} \\ \vdots \\ J_{k,1}\ J_{k,2}\ 0\ 0\ \cdots J_{k,n} \end{pmatrix} \begin{pmatrix} d_1 \\ \vdots \\ d_k \end{pmatrix}, \tag{5.3}$$

where the j columns of the gradient are also set to zero.

In particular for this problem, it is important to discuss two more features for the algorithm: the direction d and the stopping criterion. When the system does not present deterioration it is possible to obtain the optimal objective value of a point by $f^* = F(x^*)$. After the time t, the point f^* is moved away from the PF, and it becomes a new point $x_0 = x^* + D_t T$ and its image is $f_0 = F(x_0)$, where D_t refers to deterioration factor. Here, the direction d can be defined as:

$$d = \frac{f^* - f^2}{\|f^* - f^2\|} \tag{5.4}$$

In previous application of the DS method, the condition number was used as a stopping criterion, but here it is not possible to measure how close the current solution is from the optimal solution by using such criterion. Hence, the algorithm must stop when the error of Equation (5.1) overpasses a threshold $\epsilon \in \mathbb{R}$. The pseudocode in Algorithm 10 illustrates all the modifications that were introduced in order to use the DS as a controller.

Algorithm 10 Pseudocode of DS in deterioration control problems.

Input: x^*, x_0
Output: x_i

calculate the error e as in Equation (5.1)
set $x_i = x_0$
while $e \geq \epsilon$ **do**
  calculate direction d as in Equation (5.4)
  calculate the gradient of the function $J(x_0)$
  set to zero the column vectors that correspond to the deteriorated variables
  calculate $\nu = J(x)^+ d$
  set the elements of ν to zero, which corresponds to the deteriorated variables
  calculate the step size t
  set $x_i = x_i + t\nu$
  calculate the error e as in Equation (5.1)
end while
return t

5.1.1 Numerical results for DS in decomposition control problems

The first experiment for the controller algorithm used to test the newly developed algorithm takes the test function used in [Avigad and Eisenstadt, 2010]. This function was selected in order to demonstrate that the DS method can save computational resources in comparison with the method that uses the NSGA-II as its baseline approach. The problem presents a simple bi-objective function described as:

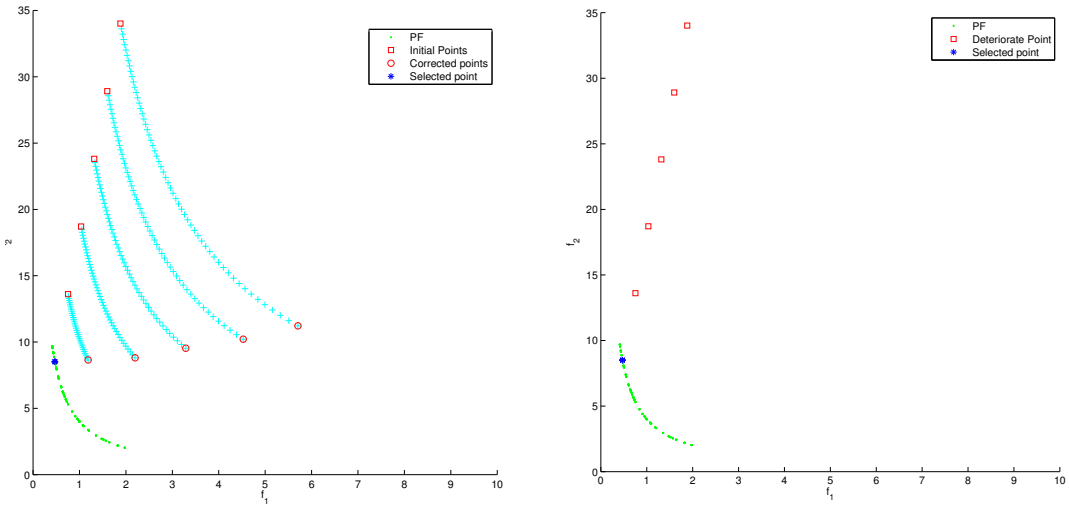
$$F(x) = \begin{pmatrix} \frac{x_1}{x_2} \\ x_1 x_2 \end{pmatrix} \tag{5.5}$$

where $x \in \mathbb{R}^2$, and x_1 is deteriorated in time T by the rate $x_1 = x_1 + 1.2T$.

In Figure 5.2, we show the results obtained by running the algorithm for a time $T = [0, 5]$. In Figure 5.2b, the PF is represented by * symbols. A random point in the PF is selected, and the squares represent how the selected points get deteriorated through the time T (it is represented by a square). Since in this part of the experiment there does not exist a controller the deterioration significantly affects the initial solution.

Figure 5.2b presents the behavior of the DS method. Here, the DS method can perform a correction step over the points that were previously deteriorated, leading to the solutions that are supposed to be closest to the original solution (represented by circles). It is important to mention that it is not possible to get closer to the original solution because only one of the two parameters was varying with the DDS,

while the other parameter was constant.



(a) Results using the controller. (b) Results without the controller.

Figure 5.2: Graphical results from the deterioration on Avigad's function.

For the next experiment a model with more complexity is introduced. Taking the bi-objective function proposed in [Köppen and Yoshida, 2007]:

$$\begin{aligned} F &: \mathbb{R}^{15} \to \mathbb{R}^2 \\ f_i(x) &= \|x - a_i\|_2^2, \quad i = 1, 2, \end{aligned} \tag{5.6}$$

where $a_1 = (1, \ldots, 1)^T, a_2 = (-1, \ldots, -1)^T \in \mathbb{R}^{15}$. Next, $x_j, j = 1, \cdots, 3$ are deteriorated by the coefficient $x = x - 0.3T$.

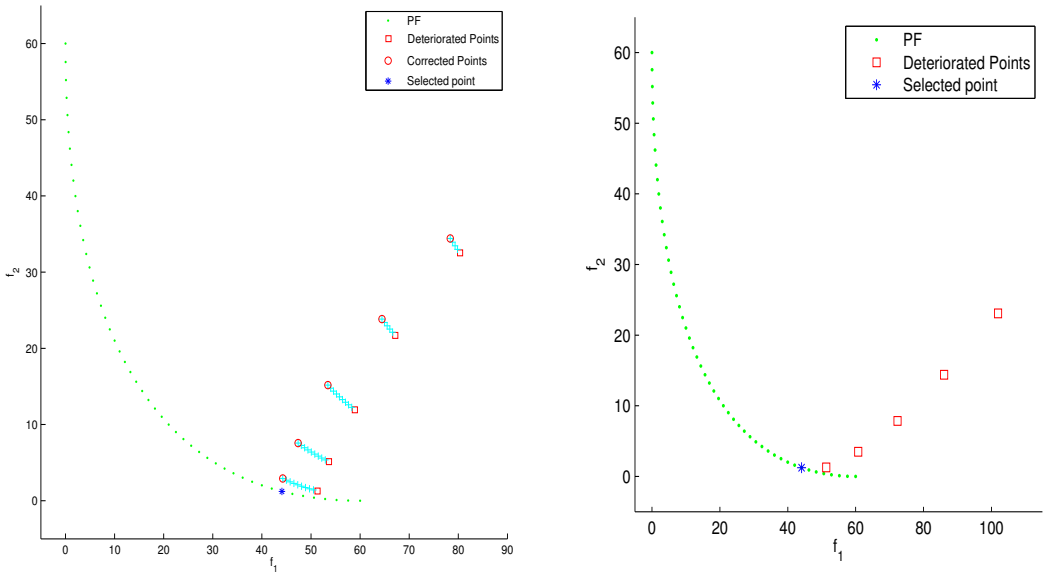
We strat by taking the Pareto optimal solutions of the problem. As in the previous experiment, one solution is picked at random in order to demonstrate the impact of the controller based on the DS.

Figure 5.3a presents the correction steps performed by the DS in order to return the deteriorated solution to the value before deterioration. On the other hand, Figure 5.3b illustrates the deteriorated solution after five periods of time ($T = 5$), in this case there do not exist a correction step.

Finally, it is possible to observe that even in a problem with many parameters deteriorated at the same time, the DS method can return closest to the original optimal value in a considerably low number of function evaluations. Maybe this could lead to a faster response when it is compared with MOEAs. Table 5.1 presents the number of function evaluations along with the number of Jacobian matrices used at each time T;

T	Number of function evaluations	Number of Jacobians
1	10	61
2	19	70
3	12	63
4	4	55
5	2	52

Table 5.1: Statistics of the function calls for the quadratic deterioration problem.



(a) Results using the controller.

(b) Results without the controller.

Figure 5.3: Graphical results from the deterioration on the modified quadratic function.

5.2 The DS method in constrained MOCP

In some of the MOCP sometimes it is desirable to move from one point to another point in objective space, but it is possible that between points there exists an infeasible region that must be avoided.

Let's assume that the initial and final points $A, B \in \mathbb{R}^k$ are given. In general, it is an "easy" task for the DS to perform such a movement for an unconstrained MOP, but when a constraint is added then the problem can not be treated in the same way. In this kind of problems the active constraints do not allow to perform a straight movement along the objective space in order to reach the point B.

When a problem contains one or more constraints, the algorithm must be changed in order to adapt to the new system. First, it is desirable that the DS performs a straight movement until one or more constraints become active. Next, in order to avoid such constraints, a side step along the constraints could be performed in order to get closer to the point B. Finally when the constraints are avoided, the DS again must perform a straight movement in order to reach the final point.

Having a constrained MOP that only has inequality constraints:

$$\begin{aligned} &\min\ (f_1, f_2, ..., f_k). \\ &\quad g_j(x) \leq 0,\ 0 \leq j \leq m \end{aligned} \tag{5.7}$$

As described above, at the beginning, it is desirable to try to perform a straight like movement in order to reach the point B. In order to do this, it is possible to use the direction $d = B - A$ until one or more constraints can be considered as active. Since it is not easy to determinate when one or more constraints become active, a threshold ϵ is introduced as an indicator. When the ϵ value is overpassed for a constraint, such a constraint is then considered as active.

Once one or more constraint become active, the next step is to perform a side step along the constraint in order to avoid them. This can be done by adding the constraints considered active to the system formulated in the Equation (5.7) such that the system can be rewritten as:

$$\min\ (f_1, f_2, ..., f_k, g_1(x), g_2(x), ..., g_i(x)). \tag{5.8}$$

Having the new system of Equation (5.8), it is possible to generate a predictor step by adding a value of -1 to the original direction d for each of the active constraints. The main idea behind this is that the predictor step tries to move away from the constraint by minimizing it. Hence, it is possible to use a direction, such that the active constraints are minimized. Thus, the new direction d_m can be formulated as:

$$d_m = (d^T, -1, -1, ..., -1)^T. \tag{5.9}$$

Figure 5.4 presents the desirable behavior of the direction d_m. By using the DS and the direction d_m is possible to find a new candidate solution that is considered "far" enough for the constraints.

After the predictor step was obtained, the corrector step must perform a line search such that one or more constraints becomes active again. Having the point obtained in the predictor step x_m and the previous candidate solution x_i, it is possible to define a direction d_d:

$$d_d = F(x_m) - F(x_i). \tag{5.10}$$

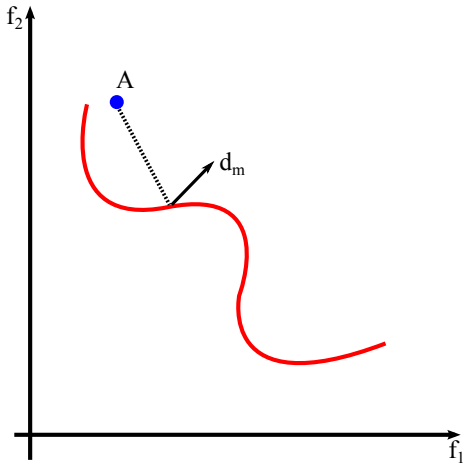


Figure 5.4: Predictor step in direction d_m.

Then it is necessary to take a corrector step such that this step must be orthogonal to the direction d_d. In order to generate the corrector step, a QR-decomposition of the direction d_d is taken. Next, taking the q_2 vector (where q_2 is the 2nd entry of the Q matrix) and the r_1 (the first entry of the R vector), the direction of the correct step can be defined as:

$$d_c = -q_2 r_1. \tag{5.11}$$

Using the direction of Equation (5.11), it is possible to find the next candidate solution by performing a movement in such a direction with the DS method. In Figure 5.5, it is possible to observe that the new predictor-corrector method can perform a side step along the active constraints.

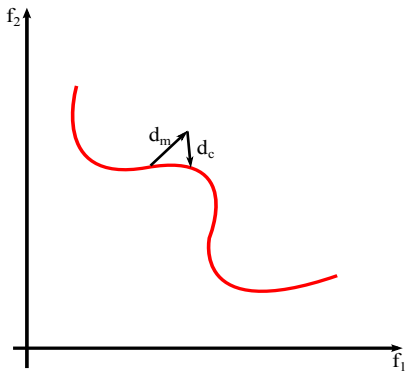


Figure 5.5: Predictor corrector step over the constraint.

Next, the algorithm must continue the procedure described above until the point B is "near" enough. Here, in order to determine if the point B is close, we used the area of the triangle formed with the point in objective space given by the current solution $F(x_i)$, the previous solution $F(x_{i-1})$ and the point B. Hence, we assumed that this triangle's area must decrease at each iteration, so when the area of the triangle of this iteration δ_i becomes bigger than the one from the previous iteration δ_{i-1}, it indicates that the current iteration is starting to "move away"" from B. Then the algorithm stops when the triangle's area of this iteration δ_i is bigger than the one from the previous iteration. Figure 5.6 illustrates the triangles used for the stoping criterion.

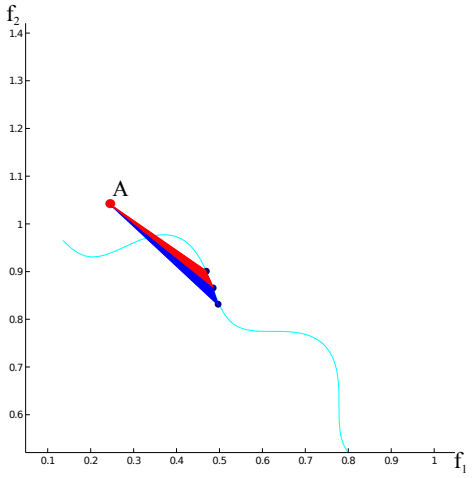


Figure 5.6: Comparison between δ_i and δ_{i-1}.

After the stopping criterion explained above is reached, the final step of the algorithm performs a straight movement in order to reach the final point B. Having the current point x_i, the new direction for the DS can be defined as:

$$d_d = B - F(x_i),$$

Finally, the algorithm must stop when $\|d_d\|_2$ is less than a defined threshold ϵ_c.

An example

In order to test the new approach of the DS for constrained problems, the algorithm was tested in Tanaka's constrained function [Coello et al., 2007]. Here, the function is defined as:

$$F = \begin{pmatrix} x \\ y \end{pmatrix}, \tag{5.12}$$

with the constraints:

$$(x^2)(y^2) + 1 + (a\ \cos(b\ \arctan(x/y))) \leq 0 \tag{5.13}$$

where $a = 0.1$ and $b = 16$.

In the particular case of this experiment a value of $\epsilon = 10^{-6}$ was set. Figure 5.7 presents the final results of the algorithm. Here, the green point represents the path that the DS performs in order to move along the objective space from point A point to point B.

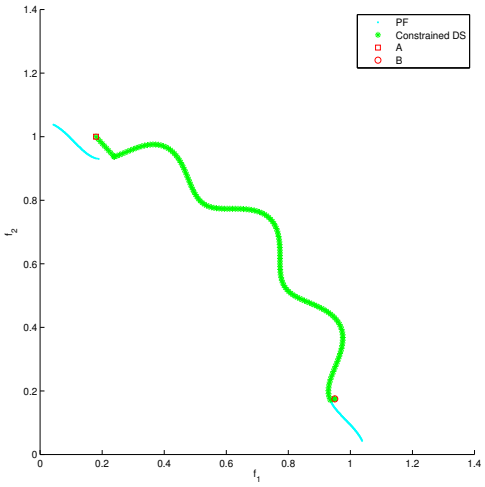


Figure 5.7: Results for the constrained DS into Tanaka's function.

5.2.1 A new predictor step

The main problem from the method discussed above is that it requires the user to define a parameter ϵ used to distinguish an active constraint. As an alternative approaching is using Equation (5.7), since it is possible to know when a point is "near" to a constraint using the condition number κ on the gradient of the function.

In this section a new way to calculate a predictor step in a system that has only one active constraint is discussed.

Given $x_0 \in \mathbb{R}^n$, $d \in \mathbb{R}^k$ and one equality constraint $g(x_0) = 0$, one can try to solve the system:

$$J(x_0)\nu = d, \tag{5.14}$$

where it is intended to calculated an orthogonal projection ν_{new} to the hyperplane defined as:

$$H := \{y \in \mathbb{R}^n | \langle \nabla g(x_0), y \rangle = 0\} \tag{5.15}$$

Figure 5.8 shows how the gradient of the function and the projection over the constraint are related. It is visible that the desirable predictor step must be the projection of ν over the constraint.

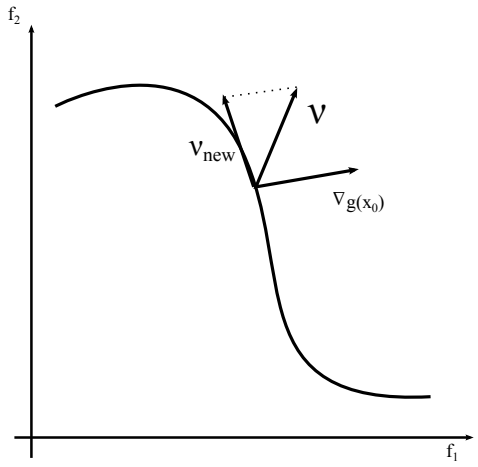


Figure 5.8: Description of the new predictor step.

From Figure 5.8, it can be stated that:

$$\nu_{\text{new}} = \nu + t\nabla g(x_0), \tag{5.16}$$

such that:

$$\begin{aligned} 0 &= \langle \nu_{\text{new}}, \nabla g(x_0) \rangle \\ &= \langle \nu + t\nabla(x_0), \nabla g(x_0) \rangle \\ &= \langle \nu, \nabla g(x_0) \rangle + t \langle \nabla g(x_0), \nabla g(x_0) \rangle \end{aligned} \tag{5.17}$$

then, the optimal step t can be defined as:

$$t = -\frac{\langle \nu, \nabla g(x_0) \rangle}{\langle \nabla g(x_0), \nabla g(x_0) \rangle} \tag{5.18}$$

Finally the direction for the predictor step can be obtained by substituting the t value from Equation (5.18) into Equation (5.16):

$$\nu_{\text{new}} = \nu - \frac{\langle \nu, \nabla g(x_0) \rangle}{\langle \nabla g(x_0), \nabla g(x_0) \rangle} \nabla g(x_0) \tag{5.19}$$

Numerical example

In order to test the new predictor step, the same problem proposed in (5.12) was used. Table 5.2 presents the comparison of the number of function calls required for the

two algorithms of this section. It is remarkable that the new predictor approach helps the algorithm to decrease the number of function evaluations and also the number of Jacobians required.

Algorithm	**Function evaluations**	**Jacobians**
DS with ϵ value	7144	732
DS with orthogonal predictor	3142	61

Table 5.2: Comparison of constrained DS algorithms.

Figure 5.9 illustrate the final result of the algorithm proposed in this section. Here, it is visible that the DS could follow the constraint and reach the B point by following the constraint.

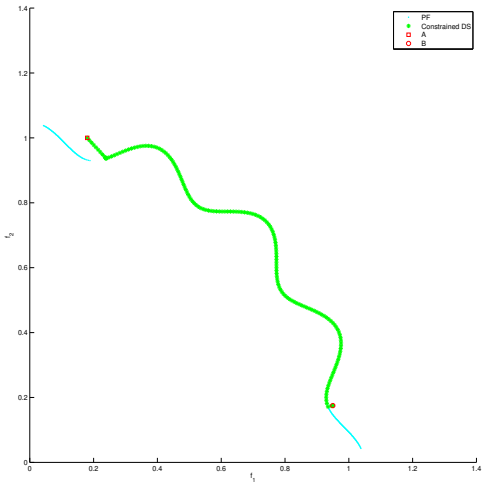


Figure 5.9: Results for the new predictor into Tanaka's function.

Chapter 6

Conclusions and Future Work

The DDS is a new method that is presented as an alternative to the DS in order to eliminate the requirement of gradient information. It uses a novel idea to approximate the gradient information that requires less function calls when it is compared with the finite differences approximation method.

As was shown in this thesis, the DDS method has demonstrated that it can use the same stopping criterion than the one used in the DS. We have also provided discussion of the requirement that the solutions used in the approximation must satisfy certain conditions with respect to linear independence of the solutions.

We also investigated the influence of the parameter r of the new method, and we concluded that when the value increases, the algorithm can perform "larger" steps, but to obtain such a number of individuals can generate a problem with the linear independence of the solutions.

First, the DDS was tested as a standalone algorithm and we showed that it could have a similar behavior than the version that uses gradient information. The algorithm can reach the PF in a particular problem with less than two thousand iterations. However, but it must be noticed that the number of function calls also depends in the size of r. Hence, the most important feature presented by the DDS method is that it only requires at least k function evaluations in order to generate a new candidate solution.

Then, we proposed that DDS was coupled to a MOEA, in order to test its behavior as a local search operator. The first algorithm that was coupled with was the NSGA-II. As a local operator the DDS improved the convergence rate on the average of the test cases for the test problem ZDT1. Here, we took advantage of the fact that the NSGA-II separates the individuals in fronts using the rank value defined inside the algorithm, so the local search technique was only applied to the individuals that were in "nearest" front to the PF.

For the NSGA-II/DDS the individuals were taken from the neighborhood of the x_0 solution, so that, an important number of function calls were saved. Because of this, it was not necessary any extra function call in order to run the algorithm. Hence, the individuals were selected by using a circle-like neighborhood, which helps to obtain the closest solutions according to x_0.

Also, we implemented a memetic algorithm that uses the MOEA/D as its baseline approach. When tested in the CEC'09 functions, the MOEA/D/DDS showed a good performance in four of the seven unconstrained bi-objective functions. The results were compared using the average values obtained in the Δ_p indicator, which measures the convergence rate and the spread of the approximation obtained by the an algorithm.

One of the critical parameters that affects the DDS method, is the selection of the direction d, e.g., in the UF4 function, the direction leads the individuals to a good convergence rate, but lose their spread along the PF.

One of the main problems found in the MOEA/D/DDS occurs when the algorithm was tested in problems having three objectives. Here, in the first case the DDS presented worse results than those obtained by the standalone MOEA/D. So, it became necessary to implement a new model in order to try to find a local search operator that has better results. In order to do that, the decomposition of the direction of the DDS was proposed.

In the decomposition of the direction of the DDS, the components of the original direction d were separated in three sets. Then, with each set, a line search was performed leading to three different solutions in the neighborhood of x_0. Then, using the Differential Evolution mutation operator, the three solutions were coupled into a single solution.

The experiment over the UF8-UF10 test function using the decomposition DDS algorithm presents better results than those obtained with the standard version of the DDS. The new algorithm outperformed the MOEA/D in two of the three test functions, and in the UF8, there does not exist a clear winner between the algorithms.

As with any other stochastic algorithm, the fine tuning of the parameters of the application of the local search operator played an important rule in the performance of DDS. Hence, as many other local search operators, it is necessary perform more experiments in order to study the rate of application of the local operator inside the MOEA. Also, it could be important to study the influence of the reference point z.

In the last part of this work the DS was applied choice to a specific MOCP in order to illustrate one of the biggest advantages of this method: The possibility to

move in any direction in objective space.

The first part of the experiments introduced the DS, as a method to control a process where the deterioration of some variables of the system exists. Here, the DS must control the rest of the variables in order to try to return to the original optimal point. This can be applied in order to avoid to change some of the parts of the system, so that some resources can be saved, e.g. money and time.

The controller method shows that it could handle the deterioration rate, even when there exist more than one variable that is affected by such rate. Also, the algorithm performs the correction step with a low number of function evaluations, this can lead to a faster reaction in comparison to a MOEA.

In the last section of the thesis, the DS was presented as a method that can go from a point in objective space to another one even if there exist some constraints that block the direct path between both points. It was proposed that the DS performs a straight path, and if some constraint is detected as active, the DS changes its movement to a predictor corrector method in order to follow such constraints until reaching the final point that was fixed at the beginning of the algorithm.

The last algorithm was tested over Tanaka's constraint function, and it was shown that indeed the algorithm can perform the movement as it was initially explained.

The first method to follow the constraints uses a parameter ϵ in order to detect the constraints and describes the path above explained. But, the addition of such parameters leads to a new problem, i.e., how to choose a good value for ϵ.

In order to generalize the algorithm for constrained problems, an alternative idea to transform the original predictor-corrector into another method that do not requires an extra parameter in order to work.

6.1 Future work

It can be observed that many of the algorithms that were presented in the discussion above have some critical points that were only was introduced. Therefore, new possible future works can be obtained from the recent work. Some of the points that can be considered as future work are:

- Investigate the influence of the r parameter in the performance of the DDS.
- Study the influence of the neighbors in the behavior of the DDS.
- Study the fine tunning of the DDS when it is coupled to some MOEA, e.g. the selection of the direction d, to which individuals and how often the local search

operator must be applied in order to obtain good results.

- Extend the work by consideing deterioration control problems.
- Apply the deterioration algorithm in some real-world MOCP.
- Extend the research in the treatment of constrained problems using the DS.

Appendix A

CEC'09 test functions.

UF1

$$
\begin{aligned}
f_1 &= x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2 \\
f_2 &= 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2,
\end{aligned}
\tag{1}
$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$ and $n = 30$. The search space is $[0,1] \times [-1,1]^{n-1}$.

PS is defined as:

$$
x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2, \cdots, n, \quad 0 \leq x_1 \leq 1 \tag{2}
$$

PF is defined as:

$$
f_2 = 1 - \sqrt{f_1},\ 0 \leq f_1 \leq 1 \tag{3}
$$

Figure 1 illustrates the PF.

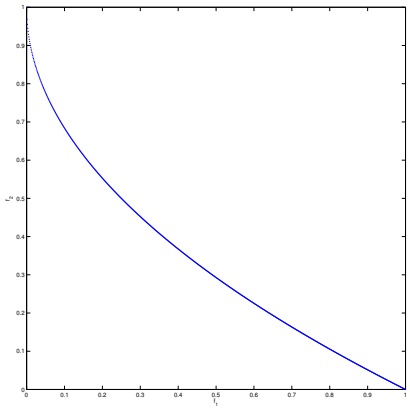


Figure 1: PF of function UF1.

UF2

$$\begin{aligned} f_1 &= x_1 + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2 \\ f_2 &= 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2 \end{aligned}, \tag{4}$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$, and y_j is defined as:

$$y_j = \begin{cases} x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_1 \\ x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_2 \end{cases} \tag{5}$$

The search space is $[0, 1] \times [-1, 1]^{n-1}$.

PS is defined as:

$$x_j = \begin{cases} \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_1 \\ \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_2 \end{cases} \tag{6}$$

PF is defined as:

$$f_2 = 1 - \sqrt{f_1},\ 0 \leq f_1 \leq 1 \tag{7}$$

Figure 2 illustrates the PF.

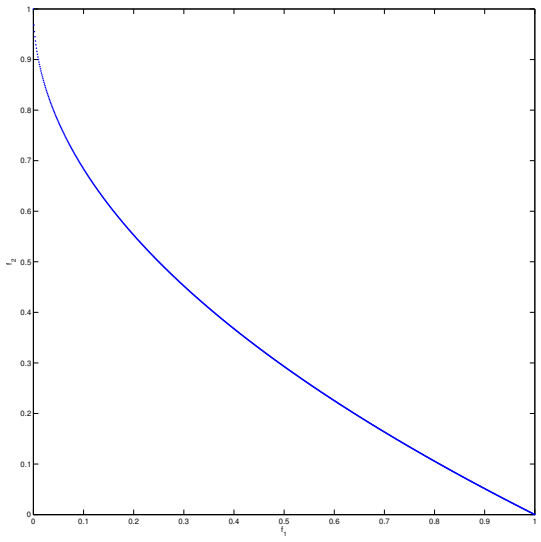


Figure 2: PF of function UF2.

UF3

$$
\begin{aligned}
f_1 &= x_1 + \frac{2}{|J_1|}\left[4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right] \\
f_2 &= 1 - \sqrt{x_1} + \frac{2}{|J_2|}\left[4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right]
\end{aligned}, \tag{8}
$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$, and y_j is defined as:

$$
y_j = x_j - x_i^{0.5\left(1+\frac{2(j-2)}{n-2}\right)}, j = 2, \cdots, n, \tag{9}
$$

The search space is $[0, 1]^n$.

PS is defined as:

$$
x_j = x_1^{0.5\left(1+\frac{3(j-2)}{n-2}\right)}, j = 2, \cdots, n, \quad 0 \leq x_1 \leq 1 \tag{10}
$$

PF is defined as:

$$
f_2 = 1 - \sqrt{f_1},\ 0 \leq f_1 \leq 1 \tag{11}
$$

Figure 3 illustrates the PF.

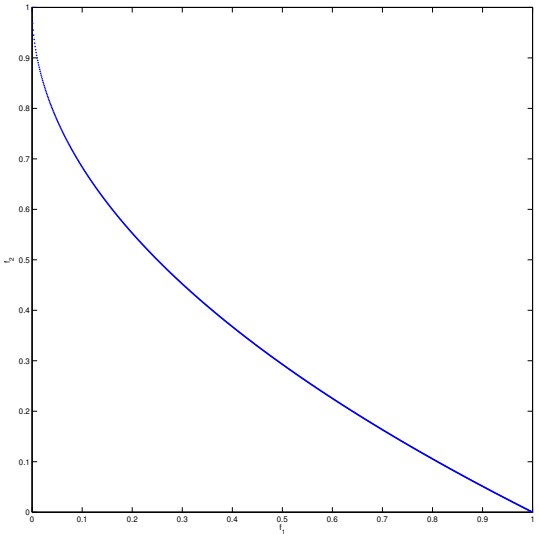


Figure 3: PF of function UF3.

UF4

$$\begin{aligned} f_1 &= x_1 + \frac{2}{|J_1|}\sum_{j\in J_1} h\left(y_j\right) \\ f_2 &= 1 - x_1^2 + \frac{2}{|J_2|}\sum_{j\in J_2} h\left(y_j\right) \end{aligned}, \tag{12}$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$, y_j is defined as:

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, \tag{13}$$

and $h(t)$ is:

$$h(t) = \frac{|t|}{1 + e^{2|t|}}. \tag{14}$$

The search space is $[0, 1] \times [-2, 2]^{n-1}$.

PS is defined as:

$$x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, \quad 0 \leq x_1 \leq 1. \tag{15}$$

PF is defined as:

$$f_2 = 1 - f_1^2,\ 0 \leq f_1 \leq 1 \tag{16}$$

Figure 4 illustrates the PF.

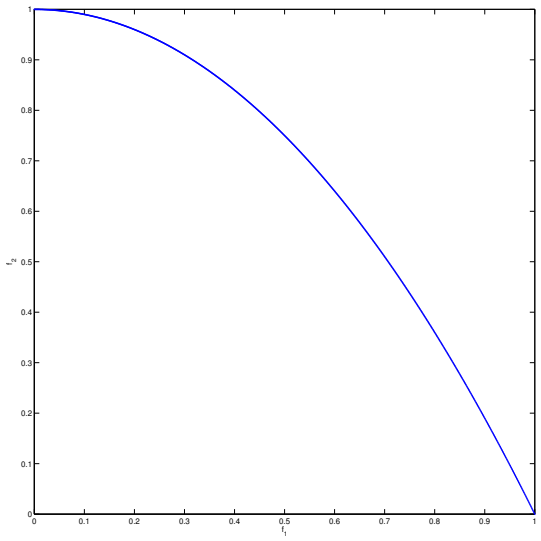


Figure 4: PF of function UF4.

UF5

$$\begin{aligned} f_1 &= x_1 + \left(\frac{1}{2N} + \epsilon\right) |\sin(2N\pi x_1)| + \frac{2}{|J_1|} \sum_{j \in J_1} h(y_j) \\ f_2 &= 1 - x_1 + \left(\frac{1}{2N} + \epsilon\right) |\sin(2N\pi x_1)| + \frac{2}{|J_2|} \sum_{j \in J_2} h(y_j) \end{aligned}, \tag{17}$$

where $J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$, $N = 10$, $\epsilon = 0.1$, y_j is defined as:

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, \tag{18}$$

and $h(t)$ is:

$$h(t) = 2t^2 - \cos(4\pi t) + 1. \tag{19}$$

The search space is $[0, 1] \times [-1, 1]^{n-1}$.

PF consist of $2N + 1$ optimal solutions defined as:

$$p_i = \left(\frac{i}{2N}, 1 - \frac{i}{2N}\right), i = 0, \cdots, 2N. \tag{20}$$

Figure 5 illustrates the PF.

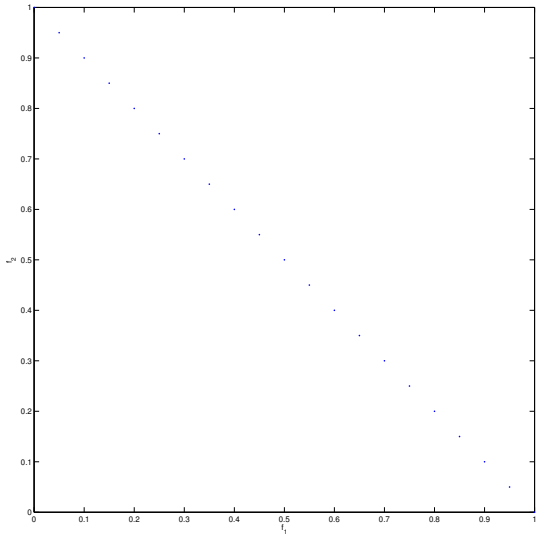


Figure 5: PF of function UF5.

UF6

$$f_1 = x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \epsilon\right)(2N\pi x_1)\right\} + \frac{2}{|J_1|}\left[4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right]$$
$$f_2 = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \epsilon\right)(2N\pi x_1)\right\} + \frac{2}{|J_2|}\left[4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right], \quad (21)$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$, $N = 2$, $\epsilon = 0.1$ and y_j is defined as:

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n. \quad (22)$$

The search space is $[0, 1] \times [-1, 1]^{n-1}$.

PF consist of:

- One insolatd point $(0, 1)$ N disconnected parts defined as:

$$f_2 = 1 - f_1, \quad f_1 \in \bigcup_{i=1}^{N} \frac{2i-1}{2N}, \frac{2i}{2N} \quad (23)$$

Figure 6 illustrates the PF.

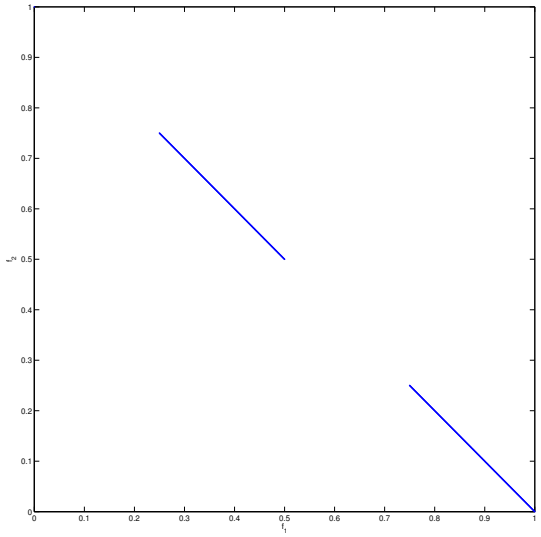


Figure 6: PF of function UF6.

UF7

$$\begin{aligned} f_1 &= \sqrt[5]{x_1} + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2 \\ f_2 &= 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2 \end{aligned}, \tag{24}$$

where $J_1 = \{j|j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j|j \text{ is even and } 2 \leq j \leq n\}$, $n = 30$ and y_j is defined as:

$$y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n. \tag{25}$$

The search space is $[0, 1] \times [-1, 1]^{n-1}$.

PS is defined as:

$$x_j = \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \cdots, n, \quad 0 \leq x_1 \leq 1. \tag{26}$$

PF is defined as:

$$f_2 = 1 - f_1,\ 0 \leq f_1 \leq 1 \tag{27}$$

Figure 7 illustrates the PF.

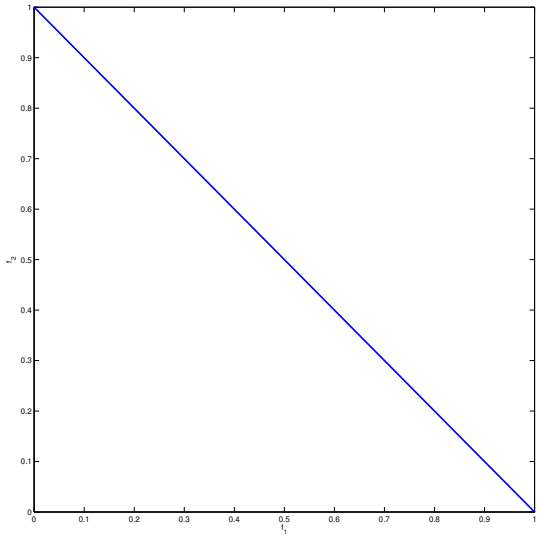


Figure 7: PF of function UF7.

UF8

$$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}\left[x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right]$$
$$f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}\left[x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right], \quad (28)$$
$$f_3 = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}\left[x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)^2\right]$$

where:

$$J_1 = \{j|3 \le j \le n \text{ and } j-1\text{is a multiplication of } 3\}, \quad (29)$$

$$J_2 = \{j|3 \le j \le n \text{ and } j-2\text{is a multiplication of } 3\}, \quad (30)$$

$$J_3 = \{j|3 \le j \le n \text{ and } j\text{is a multiplication of } 3\}, \quad (31)$$

and $n = 30$.

The search space is $[0,1]^2 \times [-2,2]^{n-2}$.

PS is defined as:

$$x_j = 2x_2\sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 3, \cdots, n. \quad (32)$$

PF is defined as:

$$f_1^2 + f_2^2 + f_3^2 = 1, \quad 0 \le f_1, f_2, f_3 \le 1 \quad (33)$$

Figure 8 illustrates the PF.

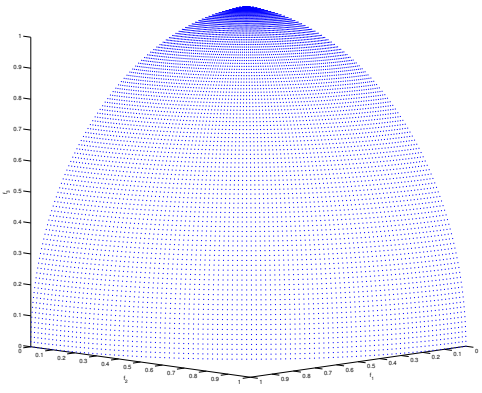


Figure 8: PF of function UF8.

UF9

$$
\begin{aligned}
f_1 &= 0.5\left[\max\left\{0,(1+\epsilon)\left(1-4\left(2x_1-1\right)^2\right)+2x_1\right\}\right]+\frac{2}{|J_1|}\sum_{j\in J_1}\left[x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right)^2\right]\\
f_2 &= 0.5\left[\max\left\{0,(1+\epsilon)\left(1-4\left(2x_1-1\right)^2\right)-2x_1+2\right\}\right]+\frac{2}{|J_2|}\sum_{j\in J_2}\left[x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right)^2\right],\\
f_3 &= 1-x_2\sum_{j\in J_3}\left[x_j-2x_2\sin\left(2\pi x_1+\frac{j\pi}{n}\right)^2\right]
\end{aligned}
\tag{34}
$$

where:

$$J_1=\left\{j|3\le j\le n \text{ and } j-1\text{is a multiplication of } 3\right\},\tag{35}$$

$$J_2=\left\{j|3\le j\le n \text{ and } j-2\text{is a multiplication of } 3\right\},\tag{36}$$

$$J_3=\left\{j|3\le j\le n \text{ and } j\text{is a multiplication of } 3\right\},\tag{37}$$

$\epsilon=0.1$ and $n=30$.

The search space is $[0,1]^2\times[-2,2]^{n-2}$.

PS is defined in two disconnected parts:

$$x_1\in[0,0.25]\cup[0.75,1],\quad 0\le x_2\le x_2,\tag{38}$$

and:

$$x_j=2x_2\sin\left(6\pi x_1+\frac{j\pi}{n}\right),j=3,\cdots,n.\tag{39}$$

PF is also defined in two parts. The first part is:

$$f2=1-f_1-f_3,\quad 0\le f_3\le 1,\ 0\le f_1\le\frac{1-f_3}{4},\tag{40}$$

and the second parts is defined as:

$$f2=1-f_1-f_3,\quad 0\le f_3\le 1,\ \frac{3\left(1-f_3\right)}{4}\le f_1\le 1,\tag{41}$$

Figure 9 illustrates the PF.

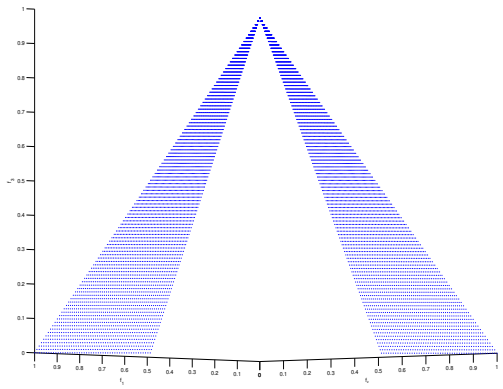


Figure 9: PF of function UF9.

UF10

$$
\begin{aligned}
f_1 &= \cos\left(0.5x_1\pi\right)\cos\left(0.5x_2\pi\right) + \frac{2}{|J_1|}\sum_{j\in J_1}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right] \\
f_2 &= \cos\left(0.5x_1\pi\right)\sin\left(0.5x_2\pi\right) + \frac{2}{|J_2|}\sum_{j\in J_2}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right], \\
f_3 &= \sin\left(0.5x_1\pi\right) + \frac{2}{|J_3|}\sum_{j\in J_3}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]
\end{aligned}
\tag{42}
$$

where:

$$
J_1 = \{j|3 \le j \le n \text{ and } j-1\text{is a multiplication of } 3\}, \tag{43}
$$

$$
J_2 = \{j|3 \le j \le n \text{ and } j-2\text{is a multiplication of } 3\}, \tag{44}
$$

$$
J_3 = \{j|3 \le j \le n \text{ and } j\text{is a multiplication of } 3\}, \tag{45}
$$

$n = 30$ and y_j is defined as:

$$
y_j = x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right), \quad j = 3, \cdots, n \tag{46}
$$

The search space is $[0,1]^2 \times [-2,2]^{n-2}$.

PS is defined as:

$$
x_j = 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right), j = 3, \cdots, n. \tag{47}
$$

PF is defined as:

$$
f_1^2 + f_2^2 + f_3^2 = 1, \quad 0 \le f_1, f_2, f_3 \le 1 \tag{48}
$$

Figure 10 illustrates the PF.

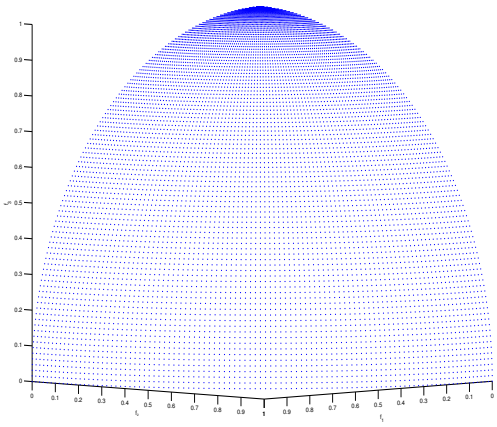


Figure 10: PF of function UF10.

Bibliography

[Avigad and Eisenstadt, 2010] Avigad, G. and Eisenstadt, E., *Robustness of multi-objective optimal solutions to physical deterioration through active control*, in SEAL, pp. 394–403, 2010.

[Brown and Smith, 2005] Brown, M. and Smith, R.E., *Directed multi-objective optimization*, Int. J. Comput. Syst. Signal, vol. 6, no. 1, pp. 3–17, 2005.

[Coello et al., 2007] Coello, C.A., Lamont, G.B. and Veldhuizen, D.A.V., *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed., Springer, USA, 2007.

[Das and Dennis, 1998] Das, I. and Dennis, J.E., *Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems*, SIAM Journal on Optimization, vol. 8, no. 3, pp. 631–657, August 1998.

[Dawkins, 1989] Dawkins, R., *The Selfish Gene*, Popular Science, Oxford University Press, 1989, ISBN 9780192860927.

[Deb, 2001] Deb, K., *Multi-Objective Optimization using Evolutionary Algorithms*, 1st ed., John Wiley and Sons, USA, 2001.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., *A fast and elitist multiobjective optimization genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, April 2002.

[Dennis and Schnabel, 1987] Dennis, J. and Schnabel, R., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1987.

[Edgeworth, 1881] Edgeworth, F.Y., *Mathematical Psychics; An Essay on the Application of Mathematics to the Moral Sciences*, P. Keagan, England, 1881.

[Ehrgott, 2005] Ehrgott, M., *Multicriteria Optimization*, 2nd ed., Springer, USA, 2005.

[Fliege and Svaiter, 2000] Fliege, J. and Svaiter, B.F., *Steepest descent methods for multicriteria optimization*, Mathematical Methods of Operations Research, vol. 51, no. 1, pp. 479–494, Springer-Verlat, 2000.

[Köppen and Yoshida, 2007] Köppen, M. and Yoshida, K., *Many-objective particle swarm optimization by gradual leader selection*, in Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms (ICANNGA 2007), Springer-Verlag, pp. 323–331, Berlin, Heidelberg, 2007, ISBN 978-3-540-71589-4, doi:http://dx.doi.org/10.1007/978-3-540-71618-1_36.

[Kuhn and Tucker, 1951] Kuhn, H.W. and Tucker, A.W., *Nonlinear programming*, in Proceedings of the 2nd Berkley Symposium on Mathematical Statics and Probability, University of California Press, 1951.

[Lara et al., 2010] Lara, A., Sanchez, G., Coello, C.A. and Schütze, O., *Hcs: A new local search strategy for strategy for memetic multiobjective evolutionary algorithms*, IEEE Transactions on Evolutionary Computation, vol. 14, no. 1, pp. 112–132, February 2010.

[Mejia and Schütze, 2010] Mejia, E. and Schütze, O., *A predictor corrector method for the computation of boundary points of a multi-objective optimization problem*, in Proceedings of IEEE 7th International Conference on Electrical Engineering Computing Science and Automatic Control, September 2010.

[Miettinen, 1999] Miettinen, K., *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, USA, 1999.

[Moscato, 1989] Moscato, P., *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*, 1989.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S.J., *Numerical Optimization*, 2nd ed., Springer, USA, 2006.

[Pareto, 1896] Pareto, V., *Cours DEconomie Politique*, F. Rouge, Switzerland, 1896.

[Rall, 1981] Rall, L., *Automatic Differentiation - Techniques and Applications*, vol. 120 of *Lecture Notes in Computer Science*, Springer, 1981.

[Schütze et al., 2010] Schütze, O., Lara, A. and Coello, C.A., *The directed search method for unconstrained multi-objective optimization problems*, Tech. Rep. TR-OS-2010-01, CINVESTAV, January 2010.

[Schütze et al., 2011] Schütze, O., Lara, A. and Coello, C.A., *On the influence of the Number of objectives on the hardness of a multiobjective optimization problem*, IEEE Transactions on Evolutionary Computation, vol. 15, no. 4, pp. 444–455, August 2011.

[Storn and Price, 1997] Storn, R. and Price, K., *Differential evolution a simple and efficient heuristic for global optimization over continuous spaces*, J. of Global Optimization, vol. 11, no. 4, pp. 341–359, Kluwer Academic Publishers, Hingham, MA, USA, Dec. 1997, ISSN 0925-5001.

[Vasile and Zuiani, 2011] Vasile, M. and Zuiani, F., *Macs: An agent-based memetic multiobjective optimization algorithm applied to space trajectory design*, Institution of Mechanical Engineers, Part G, Journal of Aerospace Engineering, August 2011.

[Zhang et al., 2009a] Zhang, Q., Liu, W. and Li., H., *The performance of a new version of moea/d on cec09 uncon- strained mop test instances*, in Proceedings of IEEE Congress In Evolutionary Computation, pp. 203–208, May 2009a.

[Zhang and Li, 2006] Zhang, Q. and Li, H., *A multi-objective evolutionary algorithm based on decomposition*, Tech. rep., University of Essex, May 2006.

[Zhang et al., 2009b] Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W. and Tiwari, S., *Multiobjective optimization test instances for the cec 2009 special session and competition*, Tech. Rep. CES-487, University of Essex and Nanyang Technological University, April 2009b.

[Zitzler et al., 2000] Zitzler, E., Deb, K. and Thiele, L., *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation, vol. 8, no. 2, pp. 173–195, 2000.