Universidade Federal de Minas Gerais

Dissertação de Mestrado

# Preference-guided Evolutionary Algorithms for Optimization with Many Objectives

*Autor:*
Fillipe Goulart

*Orientador:*
Dr. Felipe Campelo

*Dissertação submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia*

Julho 2014

# Declaration of Authorship

I, Fillipe Goulart, declare that this thesis, titled "Preference-guided Evolutionary Algorithms for Optimization with Many Objectives", and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

"Preference-guided Evolutionary Algorithms For Optimization With Many Objectives"

Fillipe Goulart Silva Mendes

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 04 de julho de 2014.

Por:

Prof. Dr. Felipe Campelo França Pinto
DEE (UFMG) - Orientador

Prof. Dr. Lucas de Souza Batista
DEE (UFMG)

Prof. Dr. Ricardo Hiroshi Caldeira Takahashi
DMAT (UFMG)

DISSERTAÇÃO DE MESTRADO Nº 833

# PREFERENCE-GUIDED EVOLUTIONARY ALGORITHMS FOR OPTIMIZATION WITH MANY OBJECTIVES

**Fillipe Goulart Silva Mendes**

DATA DA DEFESA: 04/07/2014

**Universidade Federal de Minas Gerais**

**Escola de Engenharia**

**Programa de Pós-Graduação em Engenharia Elétrica**

# PREFERENCE-GUIDED EVOLUTIONARY ALGORITHMS FOR OPTIMIZATION WITH MANY OBJECTIVES

Fillipe Goulart Silva Mendes

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Felipe Campelo França Pinto

Belo Horizonte - MG

Julho de 2014

*The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!", but "That's funny"...*

Isaac Asimov

FEDERAL UNIVERSITY OF MINAS GERAIS

# *Abstract*

Escola de Engenharia

Master of Engineering

**Preference-guided Evolutionary Algorithms for Optimization with Many Objectives**

by Fillipe GOULART

Evolutionary Algorithms became very famous in solving multi-objective problems in the last two decades. They were mainly used to approximate the whole extension of the efficient front so a decision maker could choose a preferred solution later. However, this *a posteriori* way of thinking is not well suited for problems with many objectives, mainly because the number of solutions to approximate the whole front usually increases exponentially, and the decision process can get really hard. Therefore, this work proposes the inclusion of preferences during the optimization process, such that, instead of focusing on the whole Pareto front, a smaller region is considered, so the problem of choosing among many alternatives is alleviated. Two different evolutionary methods - one with the usual non-dominated sorting with Pareto-dominance and another based on indicators - are considered together with their counterparts that take preferences into account. Along with them, a new approach is also proposed here. These algorithms are compared in a benchmark of problems with many objectives, and their outcomes are measured according to convergence and the ability to find the most preferred solutions of the decision maker. The results show that the inclusion of preferences generates significant improvements in the algorithms, indicating that they should deserve more attention in this field.

UNIVERSIDADE FEDERAL DE MINAS GERAIS

# *Resumo*

Algoritmos evolutivos tornaram-se muito famosos na resolução de problemas multiobjetivos nas duas últimas décadas. Suas aplicações consistiam praticamente na aproximação de toda a fronteira Pareto-ótima de modo que um Tomador de Decisões pudesse escolher a sua solução preferida depois. Contudo, essa filosofia *a posteriori* não é muito apropriada para problemas com muitos objetivos, sobretudo devido ao número de soluções necessárias para aproximar a fronteira normalmente cresce exponencialmente, e o processo de decisão pode tornar-se extremamente complicado. Portanto, o presente trabalho propõe a inclusão de preferências durante a etapa de otimização de maneira que, ao invés de o foco estar em toda a fronteira eficiente, uma região menor é considerada, e o contratempo de escolher dentre uma miríade de soluções é amenizado. Dois métodos evolutivos diferentes - um adotando o conhecido *non-dominated sorting* com Pareto-dominância e outro baseado em indicadores - são considerados, bem como suas adaptações que adotam preferências. Aliado a eles, um novo método é também proposto aqui. Os algoritmos são comparados em um banco de problemas de teste com muitos objetivos, e a qualidade de suas populações finais é avaliada segundo indicadores de convergência e sua habilidade de aproximar a solução preferida do decisor. Os resultados mostram que a inclusão de preferências gera melhorias significativas nos algoritmos, indicando que este método deve receber mais atenção neste campo.

# *Acknowledgements*

I would like to thank my advisor for the patience and the guts for helping me with all the tough moments of this process of becoming a master, and for showing that there are still people in the world that use science for innovation, and not just for personal achievements.

I also want to thank Master Splinter, Old Master, Master System, Master Yoda, Master Roshi and all other masters that make this title worth all the hard work.

Last but not least, I want to thank my mother and my father for the support. I know that, even without understanding a single sentence of this text, they would still read it from cover to cover.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Life is all about decisions. We are always faced with new challenges that demand a choice from us, and each different choice produces a respective outcome. Decision theory [1] is a science that deals with the evaluation of the alternatives and the selection of the course of action that generates the best outcome. In that way, it is intimately connected with the concept of *optimization*.

Optimization is a field that handles the determination of a solution that minimizes or maximizes some *objectives*, also called *criteria*. When a problem can be described by only one objective, it can be solved by a *single-objective optimization*, and when it requires more than one criterion, a *multi-objective optimization* is applied instead. In this last case, usually there is not a single solution that optimizes all objectives simultaneously, so, instead of an optimum, these problems have a set of *trade-offs*, called *Pareto-optimal* or *efficient* solutions. The solving procedure of a multi-objective problem requires then the selection of a solution that best satisfies the desires of a human decision maker (DM).

There are different methods for computing efficient solutions in order to help in the decision making process. Among them, the Evolutionary Algorithms (EA) are one of the most famous because of their ability to approximate many Pareto-optimal solutions in a single execution, thanks to the interaction among its elements. They popularized the philosophy of "first compute efficient solutions and then let the DM choose his[1] favorite later", an approach named *a posteriori*.

Despite their success in solving a lot of real-world problems, the state of the art of evolutionary computation did not return satisfactory outcomes when these problems had a higher number of objectives. The so-called field of *many-objective optimization*, comprising more than three criteria, presented a challenge for EA enthusiasts, and so

[1] Even tough "decision maker" has no gender predefined, this dissertation uses the masculine pronouns to refer to the DM, with no intention of being sexist.

new techniques started to be developed in order to cope with these high dimensional problems.

## 1.1 Scope of the problem

In summary, many-objective problems have the following difficulties:

- It is harder to visualize the solutions in order to make a final choice;
- The number of points required to approximate the efficient solutions usually grows exponentially with the number of objectives;
- The state of the art of Evolutionary Algorithms has trouble in getting points satisfactory close to the Pareto-optima.

A number of the methods presented to enable EAs to cope with many objectives focus on solving only the last issue, so they can still keep the *a posteriori* idea. However, this way of thinking is often inadequate for this class of problems, because the DM will have to make a final choice among a huge number of options. Therefore, instead of solving his problem, a new one is created.

In this work, I believe that a small adaptation of the *a posteriori* philosophy should be made in order to not approximate all of the efficient solutions, but a *smaller region* of them. Therefore, the DM should express his desires *during* the optimization process, and in the end he will have a smaller subset of preferred alternatives such that the decision procedure will be easier.

## 1.2 Goals of This Dissertation

The aim of this work is to verify if the inclusion of preferences in an Evolutionary Algorithm allows it to produce solutions with better convergence and better satisfaction of the Decision Maker's interests. For that, I test the performance of different non-preference based algorithms and their adaptations that take the DM's desires into account. Also, a new method is proposed with the goal of facilitating the interactive process of expressing his preferences and to control the size of his region of interest.

## 1.3 Structure of the Work

Chapters 2 through 4 are dedicated to the mathematical background of the work. Specifically, Chapter 2 presents the fundamentals of single and multi-objective optimization, as well as some methods to compute efficient solutions. Chapter 3 goes further in the solution process of a multi-objective problem, showing methods to select a final solution and for the DM to express his desires in order to guide the search for this choice. Chapter 4 describes the field of many objectives, when the usual evolutionary algorithms start to fail, and some techniques to try to cope with this phenomenon.

Chapter 5 presents the proposed method of this dissertation, and Chapter 6 compares the performance of it with other algorithms already present in the literature. Finally, a conclusion is drawn is chapter 7, as well as some ideas for future works.

# Chapter 2

# Multi-objective Optimization

In diverse moments of life, people are faced with the task of taking a decision. Whether they are more complicated like "Which course of action should I take for my company?" or "Which career should I follow?"; more casual like "Which route should I choose to go to my school?"; or maybe simple like "Should I wake up now?" or "Should I keep reading this text?"; the need for evaluating the possibilities and choosing among one or some of them is present in all of these cases and in a lot more.

This task is in fact so important that there is a study field called *Decision Theory* [1] dedicated (but not limited) to it. This is a truly interdisciplinary area, comprising fields like psychology, mathematics, biology, engineering etc. Whenever someone (or a group of people) takes a decision, he usually has the goal of getting the most satisfaction possible with the outcomes of this decision. For example, the director of a company takes decisions with the purpose of getting the highest profits and smallest cost; a person can opt for a career based on how much money he will earn, or maybe at how happy he will feel with it (or maybe both); a driver looks for a route that consumes the least fuel possible; a student may intent to get the most hours of sleep possible, and the best grades as well, so the decision of "wake up now" influences these two goals; etc. This satisfaction can be thought as *objectives* (or *criteria*), and the affirmative "getting the most satisfaction possible" can be translated as *optimizing* these objectives.

Therefore, the concept of *optimization* is intimately connected to the concept of *decision making*. This chapter is then devoted to the analysis of this field.

## 2.1 Decision Making and Optimization

A lot of problems of real life can take the most benefit possible when the best solution is chosen to solve them. The optimization is the area of knowledge that deals with the computation of the best solution of a problem, that is, with its optimal solution. One of its sub-fields, called *mathematical optimization* [2], assumes that there is a function $f(\cdot) : \mathbb{X} \mapsto \mathbb{Z}$ (or more) that associates to each alternative $\mathbf{x} \in \mathbb{X}$ a number $f(\mathbf{x}) \in \mathbb{Z}$ representing its objective value. The set $\mathbb{X}$ is called the *search* or *variable space*, which comprises all of the alternatives, and $\mathbb{Z}$ is the *criteria* or *objective space*, containing their outcomes. The goal of the optimization is to find the alternative (or variable) $\mathbf{x}^*$ that yield the minimum or maximum value of $f(\cdot)$.

If that sounded too abstract, one of the examples cited before can help. For instance, if $\mathbf{x}$ represents a route from the student's house to his school, then $f(\mathbf{x})$ gives the amount of fuel consumed in this trip. The driver wants then to minimize this consumption by finding the best route $\mathbf{x}^*$.

The alternatives $\mathbf{x}$ can be represented by anything, like a sequence of streets, names of people, answers like "yes" or "no"... but in this text they will symbolize vectors of real numbers, that is[1], $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$, with $x_i \in \mathbb{R}$, $i = 1, 2, \ldots, n$, and likewise for the objective values. Therefore, except for purposes of examples, it can be assumed that $\mathbb{X} = \mathbb{R}^n$ and $\mathbb{Z} = \mathbb{R}$.

The complete process of optimization usually involves:

1. Understanding the current problem and making a mathematical formulation of it;
2. Finding the best solution for this formulation;
3. Performing some post-analysis, like validity and possibly repeating the step 1 if required.

Many textbooks are concerned only with the step 2, assuming the mathematical formulation was done and the objective function $f(\cdot)$ is given as granted. Nevertheless, the step 1 is very important, since, if the formulation is not adequate, your "optimal solution" will probably be not even reasonable to the problem. Step 3 is also relevant, specially when there are more than one objective to be optimized. Unfortunately, thanks to the present goals, this text will follow this habit and assume that someone has already accomplished the task of providing a good model for the problem. Similarly, the

[1] The notation $\mathbf{x} \in \mathbb{R}^n$ will be used as well.

step 3 won't be treated directly here, but there will be some mention about it in later chapters[2].

The method for actually finding the optimum value $\mathbf{x}^*$ depends on a lot of factors, like the nature of the problem, like if is continuous or discrete; on the characteristics of the objective functions, like if they are continuous, differentiable, convex, discrete etc.; and on the number of objectives in question. There is a huge amount of books, papers, reports, blogs and related with the subject of optimization, and even if a small percent of it was discussed here this text would be five times its actual size. Because of this, I will focus on the cases when you have one objective in question - called *single-objective* - and when there are more than one - called *multi-objective*. The reader interested in further topics is referred to references like [4], [2] and [3].

## 2.2 Single-objective Optimization: When the Decision is Simple

When your problem involves the satisfaction of only one criterion, the decision problem can be solved by a single-objective optimization task. The driver who looks for the route that consumes the least fuel and the student who wishes to find the study plan that provides him with the highest scores are examples of single-objective problems. Using the notation introduced before, there is a function $f(\cdot)$ which needs to be optimized, that is, minimized or maximized. Since maximization can be achieved by minimizing the negative of the function, here I will use the convention of assuming that "optimization" is a synonym to "minimization". Therefore, the formulation of the problem can be written as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \in \mathbb{R} \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \end{aligned} \tag{2.1}$$

wherein

$$\mathcal{X} = \begin{cases} g_i(\mathbf{x}) \leq 0, & i = 1, 2, \ldots, p \\ h_i(\mathbf{x}) = 0, & i = 1, 2, \ldots, q \\ x_{i,lower} \leq x_i \leq x_{i,upper}, & i = 1, 2, \ldots, n \end{cases} \tag{2.2}$$

[2] One reason for not treating them is that they are very problem specific. The interested reader can check, for example, Rao in [3], which provides a lot of examples of formulating and solving optimization problems for mechanical engineering.

Equation (2.1) is to be understood as "find, among all possible variables $\mathbf{x}$, the one which gives the minimum value of $f(\cdot)$". In this notation, $\mathcal{X}$ is called the *feasible set*, which is a subset of the search space. Its task is to limit this space to only alternatives that are suitable for the current problem. For instance, when constructing a box, its dimensions cannot be negative; the routes given to the driver cannot induce him to drive against the traffic in one-way streets; and the study plans cannot exceed a given time so the student's health is not compromised, and they must cover the whole subject of the semester. These restrictions are called *constraints*, and are formalized in equation (2.2) as $p$ *inequality constraints* $g_i(\cdot)$, $i = 1, 2, \ldots, p$; $q$ *equality constraints* $h_i(\cdot)$, $i = 1, 2, \ldots, q$; and possible boundaries of the variables, with $x_{i,lower}$ and $x_{i,upper}$ indicating the lower and upper bounds of the $i$-th coordinate, $i = 1, 2, \ldots, n$. The image of this feasible set in the objective space is called *feasible objective space*, and it is symbolized by $\mathcal{Z} = f(\mathcal{X})$.

In order to see how the single-objective optimization can be used to solve a decision problem, consider the following example which will be continued in the next chapters.

**Example 2.1** (The perfect workout routine)**.** *Maria is a young Brazilian girl who is trying to get into shape. She already consulted a nutritionist for a diet, and now she wants to complement it with a good workout routine. For that end, she hired a personal trainer, Peter, who made a list of some daily workout routines she could practice. He translated the "get into shape" objective to "reduce her body fat percentage".*

*Table 2.1 shows*[3] *the increase of body fat (in percent) Maria will get for each workout routine, after 3 months. The goal is to find the routine that minimizes this increase, that is,*

| Number | Workout routine | Increase of body fat |
|---|---|---|
| 1 | No exercises | 10% |
| 2 | Simple walks | 2% |
| 3 | Running | -3% |
| 4 | Ballroom dancing | -5% |
| 5 | Bicycle | -6% |
| 6 | Yoga | -10% |
| 7 | Weight lifting | -12% |
| 8 | P90X | -15% |

TABLE 2.1: Increase of body fat with each workout routine after 3 months. Positive values indicate an increase of body fat, while negative ones, a decrease. Her objective is to minimize this indicator, so smaller values are preferred.

[3] P90X is a 90-days fitness program created by Tony Horton that focus on variety. In this way, you will be doing weight lifting, plyometrics, cardio routines, yoga, pilates..., that is, exercises that you may be good at and the ones you may be awful at. The creator believes that this variety helps to get better results in less time. Check their website http://www.beachbody.com/product/fitness_programs/p90x.do if you are interested.

$$\begin{aligned} &\textit{minimize} && f(\mathbf{x}) = [\textit{Increase of body fat after 3 months}] \\ &\textit{subject to} && \mathbf{x} \in \mathcal{X} \end{aligned}$$

*wherein $\mathcal{X}$ is the set of "feasible" workout routines, which can be defined as the ones that do not require more than, say, 1 hour and a half a day, and that don't offer any physical risk.*

*According to Maria's goals, she should choose the P90X routine, simply because this is the one that provides the smallest increase of body fat.*

□

This example showed why single-objective problems are considered simple in decision theory: if you have a lot of alternatives, choose the one that satisfies you the most [1], i.e., choose the optimal solution. This solution is defined as, assuming minimization,

*Definition* 1 (Global minimum). A feasible point $\mathbf{x}^*$ is called a (global) *minimum* if

$$\nexists \mathbf{x} \in \mathcal{X} | f(\mathbf{x}) < f(\mathbf{x}^*)$$

that is, there is no other feasible point that has a smaller objective value than its own.

Of course, *how* to find this minimum is another story. If you have a small number of alternatives, like in the previous example, you can just evaluate all of them and pick the best one. But, if this number is too high like in some combinatorial problems, or even infinite, like in continuous optimization, then special techniques or algorithms are required. They are out of topic here, but if you are interested, search some optimization references like [2] and [3].

The decision problem gets more interesting (and harder) when the number of objectives is greater than one. This is discussed in the next section.

## 2.3 Multi-objective Optimization: Complicating the Decision Process

Sometimes only one criterion is not enough to completely describe your problem. For example, a student may wish to achieve the highest scores and also to get the most hours

of sleep[4]. Now there are two objectives to be optimized. Moreover, they are normally not independent. In order to get high scores, he needs to spend a great time studying, and this precludes spending time in bed. Similarly, if he sleeps too much, there will be not much time left to study, and then his grades will probably drop.

Problems that require more than one objective to be formulated are called *multi-objective problems*, but also the notation *vector optimization* is present sometimes in the literature[5]. The mathematical formulation of such a problem can, at least in principle, be written as an extension of equation (2.1). Instead of just one function $f(\cdot)$, there are now $m$ functions $f_1(\cdot)$, $f_2(\cdot)$, ..., $f_m(\cdot)$, with $m > 1$, such that $f_i(\cdot) : \mathbb{R}^m \mapsto \mathbb{R}$, $i = 1, 2, \ldots, m$. A more convenient way of writing these function is by comprising them into a vector $\mathbf{f}(\cdot) = [f_1(\cdot)\ f_2(\cdot)\ \ldots f_m(\cdot)]^T$, such that $\mathbf{f}(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$. The transpose symbol $T$ is used because the vectors are assumed to be column vectors. Therefore, equation (2.1) can be written as

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \end{aligned} \tag{2.3}$$

wherein $\mathcal{X}$ represents the feasible region and $\mathcal{Z} = \mathbf{f}(\mathcal{X})$ the objective feasible region as before, except that $\mathcal{Z} \subseteq \mathbb{R}^m$ now.

Despite being a popular notation in the literature, its interpretation is not so immediate [5]. This is because of a peculiar characteristic multi-objective problems have: it is usually not possible to find an alternative that optimizes all of them simultaneously, because they are *in conflict*. In order to see how this affects the decision process, consider the continuation of the Workout Routine example.

**Example 2.2** (The perfect workout routine - Part 2)**.** *In the first part, Maria wanted a workout routine that provided her with the smallest increase of body fat. However, working out is something that requires determination and discipline, that is, it demands* effort*. And since she tends to be a bit lazy, it would be a good idea to include as another objective the "minimization of effort".*

[4]Notice that this is not the same as, e.g., optimizing the grades and sleep for, say, 8 hours a day; that would be a constraint. We don't know the preferences of the student: he may be completely dedicated to his studies and not care about sleeping just a little; or he may accept regular grades if he can rest for a greater time; it is even possible that his purpose of life be just to sleep, so he does not care about school at all. Therefore, in the general case, the two criteria need to be taken into account in the optimization procedure.

[5]To be rigorous, "multi-objective optimization" has normally the Pareto-dominance in mind, while "vector optimization" treats partial orders generated by any convex cone, which is a more general approach than the first. Of course, these terms will be defined later in this text.

*The effort is such that, when Maria does no exercise at all, or when it is a lighter training, it is small; and when the workout is heavy or she simply does not enjoy it, it receives a big value. Table 2.2 complements Table 2.1 with the effort required for each exercise, normalized between 0 and 1.*

| Number | Workout routine | Increase of body fat | Effort required |
|---|---|---|---|
| 1 | No exercises | 10% | 0 |
| 2 | Simple walks | 2% | 0.2 |
| 3 | Running | -3% | 0.6 |
| 4 | Ballroom dancing | -5% | 0.3 |
| 5 | Bicycle | -6% | 0.4 |
| 6 | Yoga | -10% | 0.7 |
| 7 | Weight lifting | -12% | 1 |
| 8 | P90X | -15% | 0.8 |

TABLE 2.2: Increase of body fat and effort required with each workout routine. The aim is to minimize the body fat gained and the effort necessary (normalized between 0 [easiest] and 1 [hardest]).

*In this case, it may be easier to evaluate these alternatives in a graphical form, as shown in Figure 2.1. For each workout, the horizontal axis gives its effort required, and the vertical axis, its increase of body fat.*

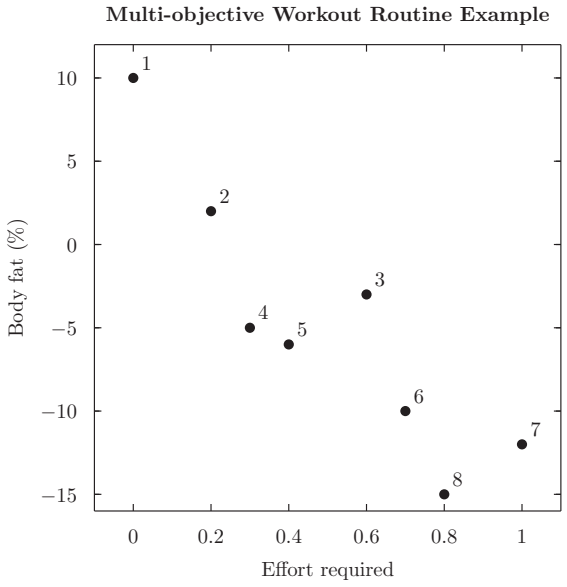


FIGURE 2.1: Workout example. For each exercise routine (numbered according to Table 2.2), the figure shows the effort required (horizontal axis) and the increase of body fat (vertical axis).

*From the figure, it can be seen that 7 (weight lifting) and 3 (running) are not good options. For the first one, the option 8 (P90X) provides a smaller increase of body fat* and *less effort. Similarly, for 3, the options 4 (ballroom dancing) and 5 (bicycle) are better in both objectives.*

*So we know that the trainings 3 and 7 are not optimal solutions. But what about the remaining alternatives, how to compare them? Exercise 8, for example, has the best decrease of body fat, but it demands more effort. Also, doing no exercises (routine 1) leaves you with the biggest body fat percentage, but it is the easiest "training" to do. If Maria has no preferences in any objective, there is no way of telling which exercise is the best.*

□

In the single-objective case, we usually don't get stuck like this because there is a natural ordering in the objectives, so the best point is well defined (when it exists, of course). In the vector case, there is no such ordering. For example, it is possible to say that 1 is smaller than 3, but what should we infer about $[0\ 1]^T$ and $[1\ 0]^T$? The first component is smaller in the first vector, but the second is bigger in it, so, there is not, in principle, a "best" among these two. Therefore, the concept of "optimum" needs to be redefined in the multi-objective case, and this will be done by introducing *dominance* and *efficiency*.

### 2.3.1 Concept of efficiency, or Pareto-optimality

Since there are no direct "less than (or equal to)" comparisons in vector-valued optimization, the concept of *dominance*, more precisely, *Pareto-dominance* is introduced.

*Definition* 2 (Pareto-dominance). Given two solutions, $\mathbf{x}_1$ and $\mathbf{x}_2$, one says that $\mathbf{x}_1$ Pareto-dominates, or simply dominates $\mathbf{x}_2$ if

- $\forall i \in \{1, 2, \ldots, m\}, f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)$, that is, $\mathbf{x}_1$ is not worse than $\mathbf{x}_2$ in any objective; *and*
- $\exists i \in \{1, 2, \ldots, m\} | f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$, that is, in at least one objective $\mathbf{x}_1$ is better than $\mathbf{x}_2$.

When both conditions are satisfied, one writes $\mathbf{x}_1 \prec \mathbf{x}_2$ or $\mathbf{f}(\mathbf{x}_1) \prec \mathbf{f}(\mathbf{x}_2)$. If $\mathbf{x}_1$ does not dominate $\mathbf{x}_2$ and neither the contrary, one says that they are *incomparable* or *non-dominated*.

The name *Pareto-dominance* was given after Vilfredo Pareto, an Italian economist who used this concept in his studies [6]. A graphical representation for two objectives is shown in Figure 2.2. For the point 1, any other point inside the filled cone will satisfy the conditions above, and, therefore, will be dominated by it. On the other hand, the points outside are not dominated by 1. Notice that, in order to check incomparability,

you have to verify if a first solution dominates the second, *and* if the second dominates the first. In Figure 2.2, 1 dominates 5, is incomparable with 3 and 4, and is dominated by 2.

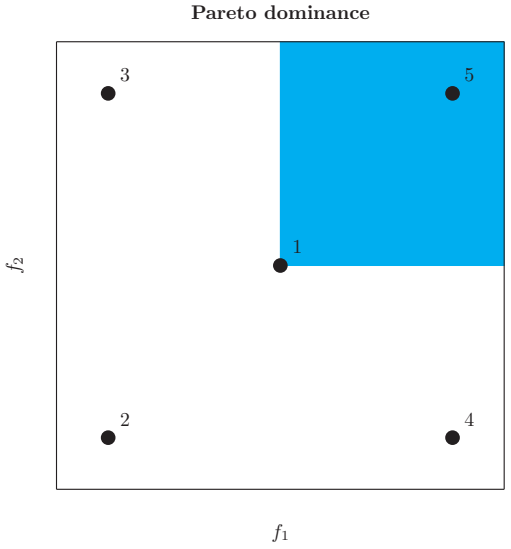


FIGURE 2.2: Graphical visualization of Pareto-dominance. For the point 1, all of the vectors inside the blue region are dominated by it. In this case, 1 dominates 5, is incomparable with regard to 3 and 4, and is dominated by 2.

The Pareto-dominance induces a *partial ordering* in the objective space [7]. This means that some pairs of different vectors can be compared (like $[1\ 2]^T \prec [3\ 5]^T$), but others cannot (like $[1\ 2]^T \not\prec [3\ 1.5]^T$ nor $[3\ 1.5]^T \not\prec [1\ 2]^T$). This contrasts with the single-objective case, when a *total ordering* is present, i.e., for any pair of elements $a$ and $b$, with $a \neq b$, it is always possible to say that either $a < b$ or $b < a$. Because of this, each point will dominate or be dominated by others, or maybe it will be incomparable with them, and some "special solutions" won't be dominated by any other alternative. These alternatives are called *Pareto-optimal*, or simply *optimal* solutions, and are rigorously defined as

*Definition* 3 (Pareto-optimal solution). A solution $\mathbf{x}^*$ is called *Pareto-optimal*, or simply *optimal*, if

$$\nexists \mathbf{x} \in \mathcal{X} \ \mid \ \mathbf{x} \prec \mathbf{x}^*$$

that is, if there is no feasible point that dominates it.

Put into words, a Pareto optimal point is such that, in order to improve one objective value, at least one of the others needs to be deteriorated.

Notice two things: an optimal solution is not necessarily the one that dominates every other solution; and there may exist more than one optimal point, sometimes infinite. The set of all Pareto-optimal solutions in the variable space is called *Pareto-optimal set*, or simply *Pareto-set*, indicated by

$$\mathcal{X}^* = \{\mathbf{x}^* \in \mathcal{X} \mid \mathbf{x}^* \text{ is Pareto optimal}\} \tag{2.4}$$

and its image in the objective space is named *Pareto-optimal front*, shortly *Pareto front*, represented by

$$\mathcal{Z}^* = \{\mathbf{y}^* \in \mathcal{Z} \mid \mathbf{y}^* = \mathbf{f}(\mathbf{x}^*), \text{ and } \mathbf{x}^* \text{ is Pareto-optimal}\} \tag{2.5}$$

Figure 2.3 shows an example of a two-objective problem. The filled region represents the feasible objective space, and the bold line indicates the optimal front. Note that, since both objectives should be minimized, the solutions are in the west-south direction. In this figure, there are infinite Pareto-optimal solutions. In the workout routine example, in Figure 2.1, the Pareto front is composed of the routines 1, 2, 4, 5, 6 and 8, because they have no other solution that dominates them, while exercises 6 and 7 are dominated and, thus, do not belong to the optimal front.

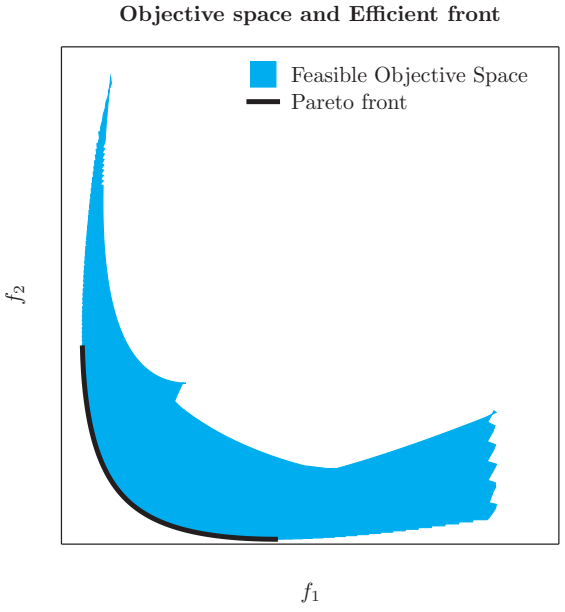


FIGURE 2.3: The filled region indicates the feasible objective space. Inside it, some points dominate, are dominated by other solutions, or are incomparable. The points belonging to the thick curve have no alternative that dominates them, and so are called optimal points. This set of these solutions is named optimal or efficient front.

Sometimes the Pareto-optimality concept is analyzed in terms of dominance cones, like shown in [8] and [7]. In this case, more general definitions of dominance can be achieved, from which the Pareto is a special case. In those situations, the optimal solutions are also called *efficient* or *minimal*. For the purposes of this text, the definition of Pareto-dominance presented suffices, but the other terms will also be used as synonyms.

#### 2.3.1.1 Weak Efficiency

Together with the concept of efficiency, there is the *weak efficiency* [8]. First, it is instructive to define the concept of *strict dominance*:

*Definition* 4 (Strict dominance). A solution $\mathbf{x}_1$ is said to *strictly dominate* another solution $\mathbf{x}_2$ if

$$f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2),\ \forall i = 1, 2, \dots m$$

that is, if $\mathbf{x}_1$ is smaller than $\mathbf{x}_2$ in all objectives. When that happens, one writes $\mathbf{x}_1 < \mathbf{x}_2$ and, similarly, $\mathbf{f}(\mathbf{x}_1) < \mathbf{f}(\mathbf{x}_2)$.

This dominance can be understood as the component-wise "smaller than", but *only in the objective space*. The other similar definitions of optimal solutions can also be made for this case.

*Definition* 5 (Weak efficient solution). A solution $\mathbf{x}^{*w}$ is called *weak efficient solution*, or *weakly efficient*, if

$$\nexists \mathbf{x} \in \mathcal{X} \ \mid\ \mathbf{x} < \mathbf{x}^{*w}$$

that is, if there exists no other solution that has all objectives better that its own, or, even simpler, if there is no other point that strictly dominates it.

*Definition* 6 (Weak efficient set/front). The set of all weakly minimal solutions is called *weak efficient set*, $\mathcal{X}^{*w}$, in the variable space, and *weak efficient front*, $\mathcal{Z}^{*w}$, in the objective space.

Note that, since a Pareto-optimal solution doesn't have all objectives better than another optimal solution, they are weakly minimal as well. Therefore, $\mathcal{X}^* \subseteq \mathcal{X}^{*w}$. Figure 2.4 shows, in a two-objective space, the efficient (lighter and thicker line) and weakly efficient (darker line) fronts of a feasible space. One simple example of the difference is to consider the solutions $\mathbf{z}_1 = [1\ 0]^T$ and $\mathbf{z}_2 = [1\ 0.5]^T$. Clearly the first dominates the second in the Pareto sense, but since $\mathbf{z}_1$ is not smaller than $\mathbf{z}_2$ in both components, they are incomparable according to the weak dominance.

Differently from Pareto-optimality, the concept of weak efficiency was never proposed in a decision context, as it is not very useful for practical applications [9]. For example, suppose you wanna buy a new suit with the smallest price and greatest elegance. If you work with the weak efficiency in mind, once the most attractive price is found, it doesn't matter if suit number 1 looks like a footy pajama and suit number 2 glows in the dark; they are both weakly efficient, so should satisfy your needs equally. Clearly, this does not help in making good decisions.

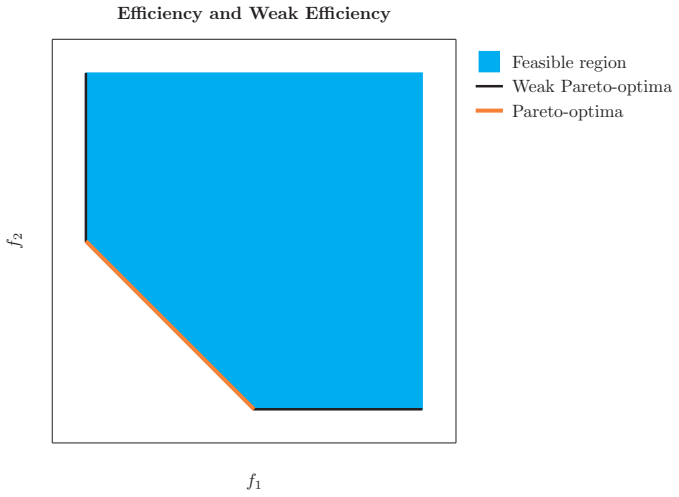


FIGURE 2.4: Weak efficient solutions in a two-objective problem. The weakly minimal front includes the minimal front and the solutions in the "south" and "west" boundaries.

So, why bother with this dominance anyway? The reason is that some methods for solving multi-objective problems (as will be shown later) have only guarantee to find weak optimal solutions. This is important, because, as mentioned before, they should be avoided if possible. Hence, it is good to see the limitations of these methods before just using them, and this requires the knowledge of the weak efficiency.

#### 2.3.1.2 Proper efficiency

I have mentioned that the weak optimality is generally not useful to take decisions. Well, sometimes, even the Pareto-dominance is too weak for applications. For example, in some cases, it is possible to improve significantly an objective in the expense of an infinitesimal decrease of other. The (limits of) ratios of improvement and deteriorations of these objectives are called *trade-off coefficients* [6]. Solutions that are efficient and also have a bounded trade-off are called *properly efficient*. Then, a last concept of optimality that is both useful and complicated is the *proper efficiency*. One reason for this complication is that there are different definitions for it, and they are not always necessarily equivalent [10]. For that, I will present the definition in the Geoffrion's sense [8]. See [10] and [7] for some different definitions.

Consider Figure 2.5 and the Pareto front shown there. From the definition of efficiency, if you are at a minimal point, say, $\mathbf{z}'$ in the figure, there is no way of reducing one objective without increasing another, that is, there is a trade-off. So, assume you start from $\mathbf{z}'$ and begin to move to the left in the Pareto front. When you do that, for each decrease you get for the first objective, the second has to increase a given amount. Because of the slope of the curve, as you approach the solution $[0\ 1]^T$, equal steps to the left results in even bigger steps up. In the limit of solution $[0\ 1]^T$, an infinitesimal move to the left

will result in an infinite step up. A corresponding effect happens in the other extreme, $[1\ 0]^T$. In these solutions, one says that the trade-off is unbounded.

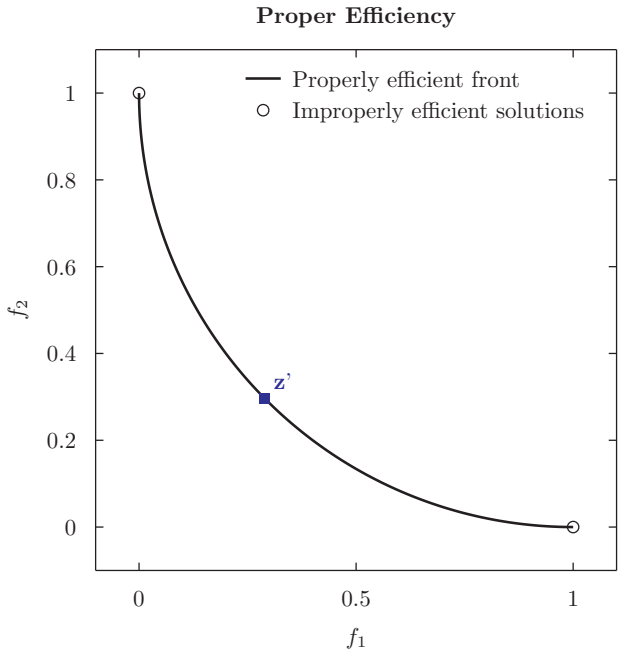


FIGURE 2.5: Example of properly efficient solutions. Each solution somewhere in the middle of the front has a finite trade-off when moving to a neighbor. But, when this neighbor approaches the limits, this trade-off becomes infinity. The solutions that provide finite trade-offs are properly efficient, and the extreme ones, improperly efficient.

Now it is possible to define proper efficiency [10].

*Definition* 7 (Proper efficiency (Geoffrion, 1968)). A feasible solution $\mathbf{x}^{*p}$ is called *properly efficient* in the Geoffrion sense if:

- It is efficient; and
- There is a real number $M > 0$ such that for all $i$ and $\mathbf{x} \in \mathcal{X}$ satisfying $f_i(\mathbf{x}) < f_i(\mathbf{x}^{*p})$ there exists an index $j$ such that $f_j(\mathbf{x}) > f_j(\mathbf{x}^{*p})$ and

$$\frac{f_i(\mathbf{x}^{*p}) - f_i(\mathbf{x})}{f_j(\mathbf{x}) - f_j(\mathbf{x}^{*p})} \leq M$$

If both conditions are satisfied, then $\mathbf{x}^{*p}$ [and its corresponding image $\mathbf{f}(\mathbf{x}^{*p})$] is called *properly Pareto optimal*, or *properly efficient*. If a point satisfies only the first condition, then it is efficient but also called *improperly efficient*.

In words, a solution is properly efficient if it is minimal and it has bounded trade-offs between its objectives. The value of $M$ is usually set by the user, and for different values a solution may be properly efficient or not. In section 3.2.2.1 it is shown a (indirect) way of setting it.

The set of all properly minimal solutions is the *properly efficient set* $\mathcal{X}^{*p}$ in the search space, and *properly efficient front* $\mathcal{Z}^{*p}$ in the objective one. Also, according to its definition, since a properly minimal solution is also efficient, it can be written $\mathcal{X}^{*p} \subseteq \mathcal{X}^{*} \subseteq \mathcal{X}^{*w}$.

These types of solutions are more useful for the decision process, both because they are already efficient and because sometimes the limited trade-off property can be useful. Also, some methods, under some circumstances, can find weakly optimal points (normally undesirable), but, in different occasions, some properly minimal solutions can be computed (see section 2.4.1). Therefore, it is good to have an idea of this kind of dominance as well.

### 2.3.2 Boundaries of the Efficient front

Given a feasible objective space, there are some special points that deserve attention.

The first point is called *ideal solution*, also named *shadow minimum* in some studies (like [11]), indicated by $\mathbf{z}^{\star}$. Its components are the individual minima of each objective, that is,

$$\mathbf{z}^{\star} = \left[\min_{\mathbf{x}\in\mathcal{X}} f_1(\mathbf{x}) \ \min_{\mathbf{x}\in\mathcal{X}} f_2(\mathbf{x}) \ \ldots \ \min_{\mathbf{x}\in\mathcal{X}} f_m(\mathbf{x})\right]^T$$

If there was a feasible point that minimized at the same time all of the objectives, it would be elected the final solution and there would be no problem in the decision making, that is, it would be an *ideal* situation. However, this is usually not the case in the *real* world, so this point is normally attainable only in dreams. See Figure 2.6 for a two-dimensional illustration of it.

Opposed to the ideal point, there is the *maximal point*, defined as

$$\mathbf{z}^{max} = \left[\max_{\mathbf{x}\in\mathcal{X}} f_1(\mathbf{x}) \ \max_{\mathbf{x}\in\mathcal{X}} f_2(\mathbf{x}) \ \ldots \ \max_{\mathbf{x}\in\mathcal{X}} f_m(\mathbf{x})\right]^T$$

that is, it is composed of the individual maxima of each function.

Another important solution is the *Nadir point*, defined as

$$\mathbf{z}^{Nadir} = \left[\max_{\mathbf{x}\in\mathcal{X}^*} f_1(\mathbf{x}) \ \max_{\mathbf{x}\in\mathcal{X}^*} f_2(\mathbf{x}) \ \ldots \ \max_{\mathbf{x}\in\mathcal{X}^*} f_m(\mathbf{x})\right]^T$$

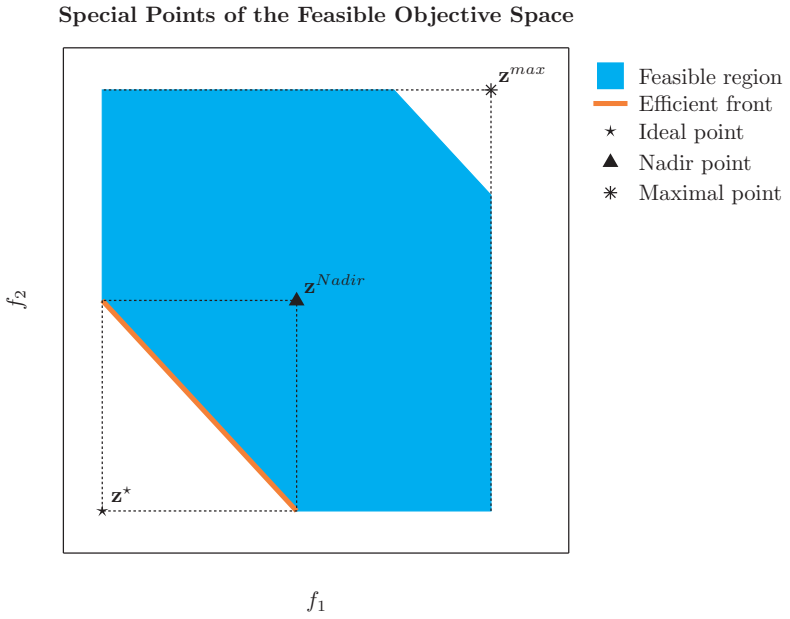


FIGURE 2.6: This figure shows the ideal, Nadir and maximal points for a given feasible objective space. Notice that the Nadir is comprised to the efficient set, while the maximal one is computed for all of the feasible space.

The difference between this and the maximal point is that the Nadir solution is limited only to the efficient front (notice that $\mathbf{x} \in \mathcal{X}^*$ here), as shown in Figure 2.6. While the first two points are easier to compute because the whole feasible objective set is considered, the computation of the Nadir can get really hard because only the portion of this set that comprises the efficient solutions is taken into account. For two objectives, one can use a *pay-off table* [8] to calculate it exactly. However, for three or more, this same technique is usually unable to correctly determine its components. Because of this, sometimes just a good approximation for this point can be used. See [9] for more details about this topic. Notice that none of these points needs to be attainable, especially the ideal one.

What is so special about these points? Sometimes, the objectives in a problem represent different things, with distinct measurement units and diverse ranges. Some methods for solving multi-objective problems require the aggregation of all criteria into one, like a weighted sum of the objectives, and it makes no sense to add centimeters with bananas. Moreover, when you have a set of non-dominated points, you may wish to perform some trade-off analysis on them. If one objective ranges from 0 to 1 and the other from 0 to 10,000, any small perturbation in the first (like 0.1) can yield a giant variation in the other (like 100) that is not actually significant for its range.

Therefore, for these operations to work properly one needs to reduce the objectives to dimension-free scales. For that, the lower and upper limits of the feasible objective set must be used. At first, the ideal and the maximal points can be adopted. Nevertheless,

there are cases when the feasible objective set is unbounded from above, so the maximal point goes to infinite. Also, most of the attention is reserved to the efficient front, so the ideal and Nadir points are most often preferred to standardize the objectives. The $i$-th objective can be converted into dimensionless with

$$\tilde{f}_i(\cdot) = \frac{f_i(\cdot) - z_i^\star}{z_i^{Nadir} - z_i^\star},\ i = 1, 2, \ldots, m$$

wherein the operation $f_i(\cdot) - z_i^\star$ means to subtract all the $i$-th component of the ideal solution from all possible points of the $i$-th objective. If these limits are not known exactly, some good approximations can be adopted instead. "Good" here means "suitable for your problem", so it will probably vary a lot from case to case. Only when the objectives are standardized, a discussion about relative importance of criteria, their weights and an interpretation of trade-off coefficients become meaningful.

### 2.3.3 Shapes of the Efficient front

In this subsection I will consider briefly some characteristics of the efficient front. First, as all of the figures shown so far depict two-objective problems (two dimensions), the Pareto front was always a curve (one dimension). In general, if a problem has $m$ objectives, its minimal front, assuming it exists, will be a $(m-1)$-dimensional hyper-surface [8]. This is not, however, a rule: there may be some degenerate cases, like a curve in a three-objectives problem, or a single point in a two-criteria one. So, the best that can be said is that the dimension of the efficient front of a $m$-objectives problem will be less than or equal to $m-1$.

The other consideration is about the shapes the fronts can present. Figure 2.7 shows the most common encountered in continuous problems. I will describe them in a simple and intuitive way. If you are interested, the reference [8] contains more technical information about this topic.

The top left shows a *convex front*. It is such that, for any pair of optimal points $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$, the line going from the first to the second remains entirely in the image of the objective functions. The top right illustrates a *concave front*. It is the opposite of the convex case: the line between two optimal points lies outside the feasible space. The bottom left indicates a more general front, with a convex portion and another concave. It can be referred as a *mixed front*. Finally, the bottom right is an example of a *disconnected front*. Notice that only some portions of the "south-west" boundary of this feasible region are efficient.

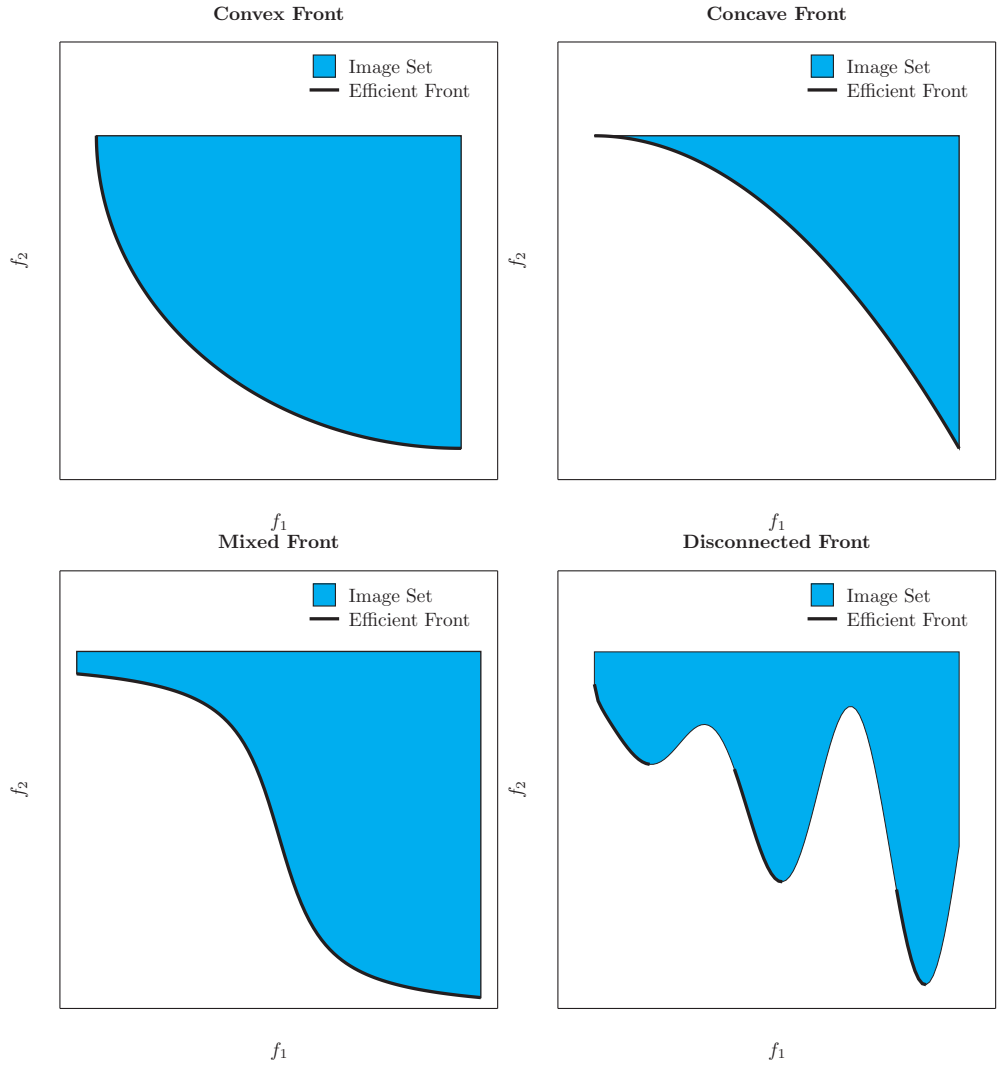


FIGURE 2.7: Some common shapes of efficient fronts. Top left: convex; top right: concave; bottom left: mixed (convex and concave); bottom right: disconnected.

The convex front is possibly the most important, since a lot of real-life problems present as efficient front a convex one. Also, many known methods for computing efficient points can only approximate the solutions in the convex portions. This will be shown in the next section.

### 2.3.4 Solving a Multi-objective Problem

Let's summarize by making a small comparison about optimality in single and multi-objective optimization. In the first case, there is one minimal solution (or maybe more, but with the same objective value), which makes the decision process easy: if you have only one criterion, choose the solution that optimizes it. In the second case, instead of a minimum solution, there are efficient points, which could be many, or even infinite.

Without further information, they are equally good, so, in principle, all of them should be considered as "the optimal solution".

However, in practical problems, only one or a few alternatives are effectively implemented. Consider the last part of the Perfect Workout example to see that:

**Example 2.3** (The perfect workout routine: Part 3)**.** *In part 2, Peter formulated Maria's problem into finding the workout routine that yields the minimum increase in fat body percentage after 3 months and the smaller effort required. From all eight exercises, 1, 2, 4, 5, 6 and 8 are efficient, and they are shown in Figure 2.8. Even after excluding the non-efficient alternatives, she still needs to choose only one for the next three months. How should she proceed?*

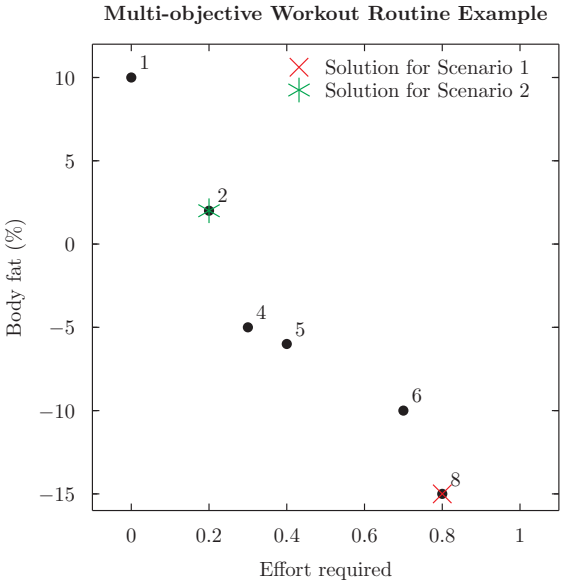


FIGURE 2.8: Efficient workout routines. Among these alternatives, routine 8 (P90X) can satisfy Maria's preferences in Scenario 1, while routine 2 is more preferable in Scenario 2. See the text.

*The concept of efficiency says that, assuming no further information is available, the efficient alternatives are all equally preferable. However, Maria is a human being, so she has different needs and opinions, i.e., she has preferences. She should then pick the solution that pleases her most for a given situation.*

*Consider the following scenarios:*

- ***Scenario 1:*** *Maria decided to make some money by working as a model in a bikini's company, and the pictures start to be shot in 3 months. In that case, she could give more importance to her increase in body fat at the sacrifice of her laziness, so a candidate like 8 (P90X, shown by an '×' in Figure 2.8), which gives the smaller body fat increase, would be the final solution in this case.*

- ***Scenario 2:*** *Maria will visit her grandmother in Canada for three months, during the winter. She (the grandmother) is always complaining how her granddaughter looks so skinny. Under these circumstances, Maria could give herself a break and prefer a more effortless activity, like exercise 2 (Simple walks, indicated by the asterisk in Figure 2.8). Maybe doing nothing (number 1) could also be an option, but she might think the increase of body fat is too big compared to the walks, so this last one would be the final solution of this scenario.*

□

Just like Maria can only follow one workout routine per time, an employer usually hires a few people for the job, even tough hundreds of the candidates can be "efficient" according to his criteria; a student cannot sleep and study at the same time; etc. So, finding Pareto-optimal solutions is not enough to finish the decision procedure.

When solving a multi-objective problem, there is a person that [12] is assumed to know the problem in hand and who is able to provide preference information related to the different solutions. This person is called *decision maker* (DM)[6]. Besides the DM, there is also an *analyst*, which may be another person, or a group, or even a computer program responsible for the mathematical modeling and computing sides of the process. His aims are to help the DM in various stages of the optimization. In the workout example, one may elicit Maria as a decision maker, and Peter as the analyst.

Therefore, in this point of view, finding all of the Pareto-optimal solutions is not "solving a multi-objective problem". Rather, it can be understood as "helping a human decision maker in considering the multiple objectives simultaneously and finding a Pareto-optimal solution that pleases him the most" [12]. It is assumed that the DM prefers efficient solutions, or maybe properly efficient, and the weakly minimal ones should be avoided. But, even so, the Pareto set doesn't tell which solutions to choose, but which points to avoid. The final solution must have some insight from the DM.

## 2.4 Methods for Finding Efficient Points

Apart from the discussion of the last section, let us forget about the DM for the rest of this chapter. I will now present here some popular methods for computing optimal solutions for a given multi-objective problem. They are classified as *Scalarizing Methods*, *Deterministic Methods without Scalarization* and *Evolutionary Algorithms*. Some of

[6]The case of multiple DMs is not considered here, since it falls into other problem involving harder process like negotiation, and it is out of topic for this work.

them, like the first, are able to compute sometimes efficient points, but weakly and properly ones can be expected; and others, like the last, provide in the best case a good approximation of the Pareto front, but with no guarantee of optimality. For each one, I will present some characteristics, weaknesses and strengths.

### 2.4.1 Scalarizing methods

In the scalarizing methods, the functions are combined in order to convert the original multi-objective problem into a single-objective one. In this way, instead of dealing with vectors, one handles scalars. This is a clever approach, because it is possible to use single-objective techniques, which are very well developed.

When the problem is scalarized, the solution of it is a single point. It is then necessary to discuss how this point relates with the solutions of the original problem. More precisely, the following questions can be made [8]:

1. Does the optimization of the scalarized problem always result in an efficient solution of the original one?
2. Converting a vector into a scalar requires the use of some parameters (e.g., aggregating its components using a weighted sum demands a vector of weights). Different parameters may produce distinct solutions. The question is: is it possible to generate all of the minimal solutions by varying these parameters?
3. This question is related to the previous one: assuming the optimization of the scalarized problem yields an efficient point, how does the choice of the parameters control the position of the solution in the Pareto front?

As you may think, each scalarizing method may have a different answer to these questions. I will respond them for each of the techniques presented here.

One last note worth mentioning: in the second question, just because I want to know if a method can produce all of the minimal points, it doesn't mean that I *want* them all. The idea is more like "If I want a specific region of the front, will this method be able to find a point belonging to it?". It is similar to a fire alarm. You probably don't want to see it working, because it usually means a fire is going on. However, in the eventual case something is actually burning, it better be working properly!

#### 2.4.1.1 Weighting Method

Also known as *linear aggregation*, in this method the original vector function is converted into a scalar by aggregating its components into a weighted sum, which should be minimized. The original problem then becomes

$$\begin{aligned} \text{minimize} \quad & f_w(\mathbf{x}) = \sum_{i=1}^{m} w_i f_i(\mathbf{x}) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{m} w_i = 1 \\ & w_i \geq 0, \quad \forall i = 1, 2, \ldots, m \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.6}$$

wherein $w_i$ is the weight for the $i$-th objective, considered non-negative, and whose sum over all objectives is assumed[7] to be 1. These weights can be viewed as a "relative importance" of the respective objective. For instance, in the workout example, if Maria thinks that reducing her body fat percentage is more important than decreasing the effort, she can assign $w_1 = 0.8$ for the first and $w_2 = 0.2$ for the second, meaning roughly "losing body fat deserves 80% of her attention".

Each vector of weights represents a different problem, so a diverse solution can be expected when you vary them. Figure 2.9 shows a two-objective example. By changing the relative values of the weights, different problems with distinct solutions are obtained.

Now, let's get down to business and answer the questions of the introduction. First, are the solutions of the weighted problem always Pareto-optimal? As can be seen in [6, 8], the solutions are weakly Pareto-optimal, but if all of the weights are *positive*, not just non-negative as is written in (2.6), the solutions are properly Pareto-optimal. If you think a little about that, it does not make much sense to let a weight assume the value zero, because it means that the corresponding objective has no significance at all, so why include it in the first place? Nevertheless, zero weights are still included to make the formulation more general.

Second, can all of the efficient points be found by this method? The answer is a big "yes" followed by "provided the Pareto front is convex". If the front is convex, for every optimal solution $\mathbf{x}^*$ there exists a weight $\mathbf{w}$ such that the weighted problem (2.6) yields $\mathbf{x}^*$ as a solution [6]. Conversely, if the optimal curve is non-convex, there does not exist any $\mathbf{w}$ such that the solution of (2.6) lies in this non-convex portion [13]. Das and Dennis in [13] show why this sad effect happens.

[7] A simple normalization takes care of this condition.

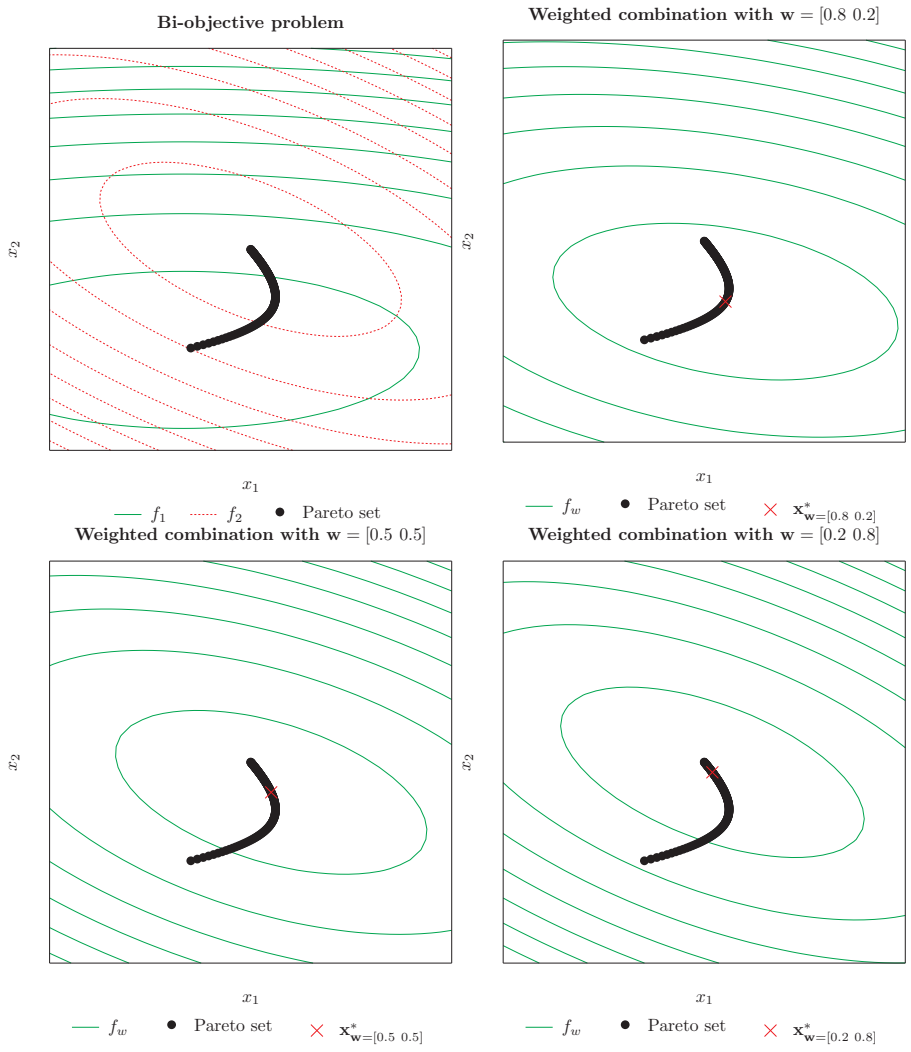


FIGURE 2.9: Solving a two-objective problem by the weighting method. Top left: the level curves of two quadratic functions and the respective Pareto set. Top right: level curves of the weighted problem with $\mathbf{w} = [0.8\ 0.2]^T$, favoring the first objective; the 'x' indicates the solution with this aggregation. Bottom left: same thing, but with $\mathbf{w} = [0.5\ 0.5]^T$, giving the same importance to both objectives. Bottom right: similar, with $\mathbf{w} = [0.2\ 0.8]^T$, preferring more the second objective.

There are some more considerations to be made in order to answer to the last question. One important thing is that the idea of "relative importance" mentioned before is only valid *when the objectives are dimensionless* [9]. So, even for convex problems, when the functions have different ranges, do not expect to have a good control of the position of the solutions by just changing the weights. This is also illustrated in Figure 2.9. The case of equal weights, for instance, does not result in a solution in the middle of the Pareto set because the functions have distinct ranges. This is not stressed enough in many texts, but it is important. As stated before, the standardization usually means computing the ideal $\mathbf{z}^\star$ and the Nadir (or at least a good approximation of it) $\mathbf{z}^{Nadir}$ points, but choosing suitable ranges (which are problem specific) can also do the trick.

For the other consideration, suppose you want to approximate the whole Pareto front. For that, you must solve the problem (2.6) a lot of times with different options of weights. And there lies the question of how to set their values in order to cover the efficient front. In two dimensions, one can set $w_1 = w$, $w_2 = 1 - w$ and vary $w$ up in the interval $[0, 1]$. But for three or more this is not so obvious, because one would need to wander around the whole hyper-surface $\sum_{i=1}^{m} w_i = 1$. There is, however, a related study in [11].

Finally, even if the Pareto front is convex, the functions are normalized and you managed to get a uniform distribution of weights for any number of objectives, there is no guarantee that this will yield uniform points in the efficient front [13]. The problem is not just that the final result will not look so nice, but, more important, it is hard to control the location of the solutions (see Figure 2.10). It may be possible that a small perturbation in a weight results in a very far point from the previous one, or even the opposite[8].

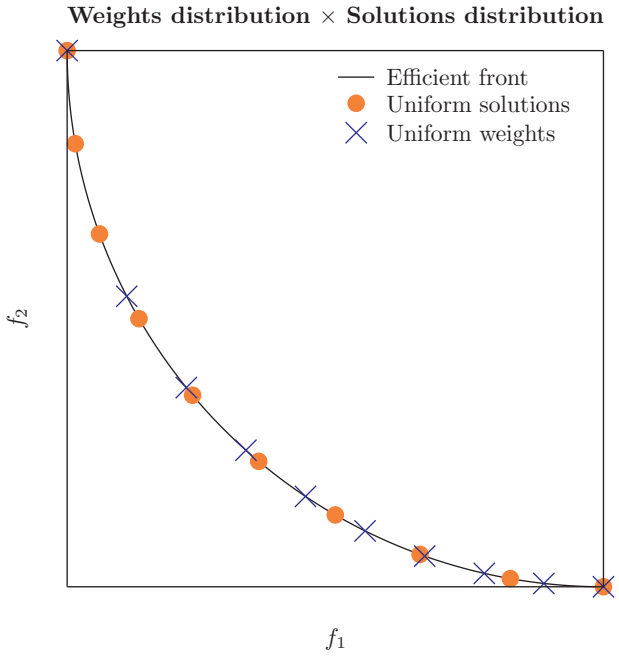


FIGURE 2.10: Generating 10 efficient solutions with uniform weights. The '×' represents solutions obtained by setting $w_1 = w$ and $w_2 = 1 - w$ and varying $w$ from 0 to 1 in equal increments. For comparison, the circles indicate what is a true equally spaced set of solutions.

In summary, the weighting method is one of the simplest scalarization methods to solve multi-objective problems. Its solutions are weakly Pareto-optimal, and if *all* of the weights are positive, they are properly optimal. However, only convex portions of the front are reachable, and, even when that is the case, it is hard to control the location of the solutions.

[8] In [13], Das and Dennis show some bi-objective examples of weights distributions that generate uniform solutions. These distributions are, in general, not uniform.

A lot of details were provided here about this method. The goal is not defame it, but more to warn analysts of its limitations before, for example, blindly summing all objectives and hoping to get a solution in the middle of the front.

#### 2.4.1.2 $\epsilon$-constraint Method

In the $\epsilon$-constraint method, also called *Compromise Programming* [8], one function, say, $f_k$, is chosen to be minimized, while the other $m-1$ receive upper bounds $\epsilon_i$, wherein $i = 1, 2, \ldots, m$ and $i \neq k$. So, $m-1$ new constraints are introduced in the problem, and the original multi-objective one becomes

$$\begin{aligned} \text{minimize} \quad & f_k(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq \epsilon_i, \quad i = 1, 2, \ldots, m, i \neq k \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.7}$$

To get an idea of how this works, check the Figure 2.11. If $f_2$ receives the upper limit $\epsilon_1$, the minimization of $f_1$ generates the point $\mathbf{x}^*_{\epsilon_1}$, whose image is shown as a square in the figure. If $f_2$ gets a different bound $\epsilon_2$, another solution $\mathbf{x}^*_{\epsilon_2}$ is obtained. By varying the bounds of the $m-1$ remaining objectives, different points are obtained.

In principle, any function can be chosen to be minimized, and the solutions of (2.7) are *weakly efficient* with regard to the original problem [6]. In order to answer the first question, there are, nevertheless, ways of guaranteeing efficiency instead of just weak efficiency [6]. If a point $\mathbf{x}^*$ is the solution of the $\epsilon$-constraint problem solved varying $k$ from 1 to $m$, that is, each time minimizing one different objective and with the same vector $[\epsilon_1\ \epsilon_2\ \ldots\ \epsilon_m]^T$ of upper limits, than it is Pareto optimal. Alternatively, if it can be shown that $\mathbf{x}^*$ is a unique solution of (2.7) with a given value of $k$, then it is Pareto optimal.

The verification that a point is a unique solution of a problem is hard (unless the front is convex), and solving $m$ different problems in order to get just one efficient solution may be a big price to pay, principally when there is a great number of objectives. The advantage of this method is that, theoretically, any minimal solution can be found, even the ones in non-convex portions. This answers the second question made before.

If someone desires to approximate the whole front using this method, it can be accomplished by setting different upper bounds $\epsilon_i$. Of course, there is a chance of adjusting a too restrictive limit (then the problem has no solution), or bounds that are too loose

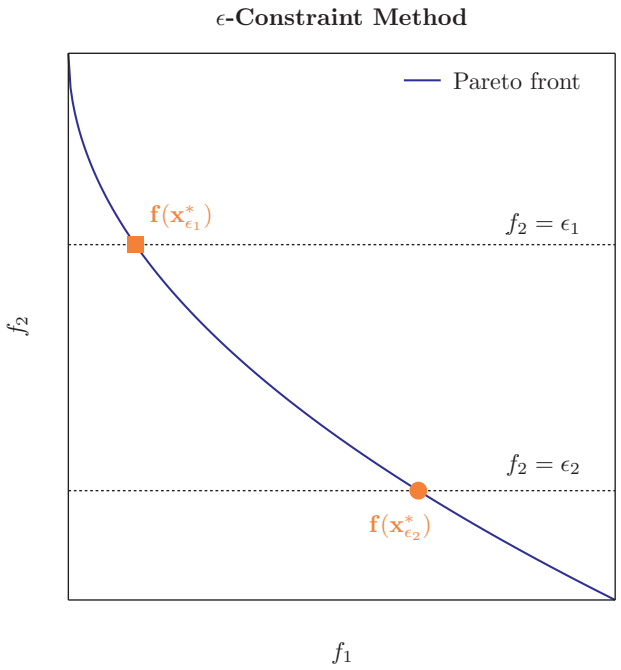


FIGURE 2.11: Two-objective example solved by the $\epsilon$-Constraint method. Here, $f_1$ is minimized and $f_2$ receives upper limits. When $f_2$ needs to be smaller or inferior than $\epsilon_1$, minimizing $f_1$ generates the solution $\mathbf{x}^*_{\epsilon_1}$. A similar reasoning applies to when $f_2 \leq \epsilon_2$.

(so the solutions are always the same). In order to not waste time with these degenerate cases [8], the ideal and the Nadir points are required again. So, these two points create a hyper-box limiting the Pareto front, and a hyper-grid can be created by varying the vector of constraints inside it. This technique provides a better control of points than the weights do, but, depending on the front's shape, even a uniform grid cannot guarantee uniform solutions.

The $\epsilon$-constraint method beats the weighting method with its ability of approximating any efficient solution, instead of just the ones in convex portions. However, the assurance of efficiency is harder to verify (unless the front is convex, situation where the weighting method can succeed as well) and the complexity of the problem is increased by augmenting the constraints. Also, in order to avoid degenerate problems, the knowledge of the ideal and Nadir points is necessary. Typically, problems with a small number of objectives can be solved without much complications by this method.

#### 2.4.1.3 Weighted Metrics

In the Weighted Metrics method[9] a different approach is pursued. In the weighting method, one deals with preferences in the objectives, and in the $\epsilon$-constraint, upper bounds are adopted. Here, the concept of *aspiration levels* is introduced.

These levels are simply the desires one may have for each objective. For example, a student may wish to score 95% of his efficiency index and sleep during 8 hours per night. Also, Maria can decide to lose 20% of body fat and do only 0.2 of effort. More generally, one may opt for the level $z_1^r$ in objective $f_1$, $z_2^r$ in objective $f_2$, and so on, and then he has a *reference point* $\mathbf{z}^r = [z_1^r \ z_2^r \ \dots \ z_m^r]^T$. The idea is then to get a solution that is the *closest possible* to this point. This is a more natural way of thinking, and should be a good reason to popularize this method. However, it hasn't received the importance it apparently deserves. I will point some reasons for it in the end of this section.

There are some definitions of "getting close" to a reference point [9]. In this method, this is translated as "finding the point with the smallest distance to $\mathbf{z}^r$" or, more generally, with the smallest value of a metric. The $L_k$ metrics, or Minkowski distance, or even norms, adopted here, are defined as

$$L_k(\mathbf{f}(\mathbf{x}), \mathbf{z}^r) = \|\mathbf{f}(\mathbf{x}) - \mathbf{z}^r\|_k, \quad k = 1, 2, \dots, \infty \tag{2.8}$$

in which, if $1 \leq k < \infty$, equation (2.8) takes the form

$$\|\mathbf{f}(\mathbf{x}) - \mathbf{z}^r\|_k = \left[\sum_{i=1}^{m} (\mathbf{f}(\mathbf{x}) - \mathbf{z}^r)^k\right]^{1/k}, \ 1 \leq k < \infty$$

and, if $k = \infty$, the distance is called Tchebycheff norm, and assumes the form

$$\|\mathbf{f}(\mathbf{x}) - \mathbf{z}^r\|_\infty = \max_{i \in \{1,2,\dots,m\}} |f_i(\mathbf{x}) - z_i^r|$$

Therefore, the original multi-objective problem is converted in a scalar one by minimizing a weighted metric in the following way:

[9] It is also called "compromise programming" in [6] and "distance to a reference point" in [8]. The first one is the same as the one attributed to the $\epsilon$-constraint method (this reflects how the literature is still not unified), and the second will be reserved by another method presented in a later chapter. Therefore, I will stick with "Weighted Metrics".

$$\begin{aligned} \text{minimize} \quad & L_k(\mathbf{f}(\mathbf{x}), \mathbf{z}^r, \mathbf{w}) = \|\mathbf{w}^T(\mathbf{f}(\mathbf{x}) - \mathbf{z}^r)\|_k \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{m} w_i = 1 \\ & w_i \geq 0 \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.9}$$

for a given reference point $\mathbf{z}^r$, a vector of weights $\mathbf{w} = [w_1 \; w_2 \; \ldots \; w_m]^T$ and a given metric value $k$. These weights are, in principle, used to normalize functions with different ranges, but, if this was already done[10], they can be used to generate different solutions.

The ways one can create different solutions are by: (i) changing the reference point; (ii) setting a new metric value $k$; and (iii) varying the weights. Some examples are shown in Figure 2.12. For a given reference point, this method tries to get close to it according to different metrics and sets of weights. This technique is very simple, and the figures show how it can be versatile. However, the answers to those three questions will probably dry out the fantasy.

First of all, the solution to the weighted metric (2.9) is not always Pareto optimal. Sometimes even dominated solutions are obtained. For example, if you choose a reference point that is feasible and not optimal, then the distances in (2.9) will have their minimum value (equal to zero) in exactly the (not efficient) reference point. There is a lemma [8] that says that, in order to avoid dominated solutions, the reference point should be the ideal point $\mathbf{z}^\star$ or any other such that $\mathbf{z}^r \prec \mathbf{z}^\star$. Please note that this is not a necessary condition, since sometimes minimal points can be obtained even if $\mathbf{z}^r$ is dominated by the ideal point, as the examples of Figure 2.12 show. Nevertheless, most books already use $\mathbf{z}^\star$ in place to $\mathbf{z}^r$ in the formulation (2.9).

So, from now on, assume the reference point satisfies the condition above. Now, are the solutions of (2.9) efficient in the original problem? The answer is [6]:

- If the metric used is finite (that is, $1 \leq k < \infty$), then, if a solution $\mathbf{x}^*$ is unique (a hard thing to test) or the weights are all positive, then it is efficient;
- If the Tchebycheff metric is in use, then if a solution $\mathbf{x}^*$ is unique, it is efficient. And if the weights are all positive, then the solution is *weakly* efficient.

It seems here that the Tchebycheff has a disadvantage with regard to the other metrics. But, what about the possibility of generating any solution in the front? This is where

[10] The standardization of the functions is important for post-analysis. That is why this is mentioned here. See [9] for details.

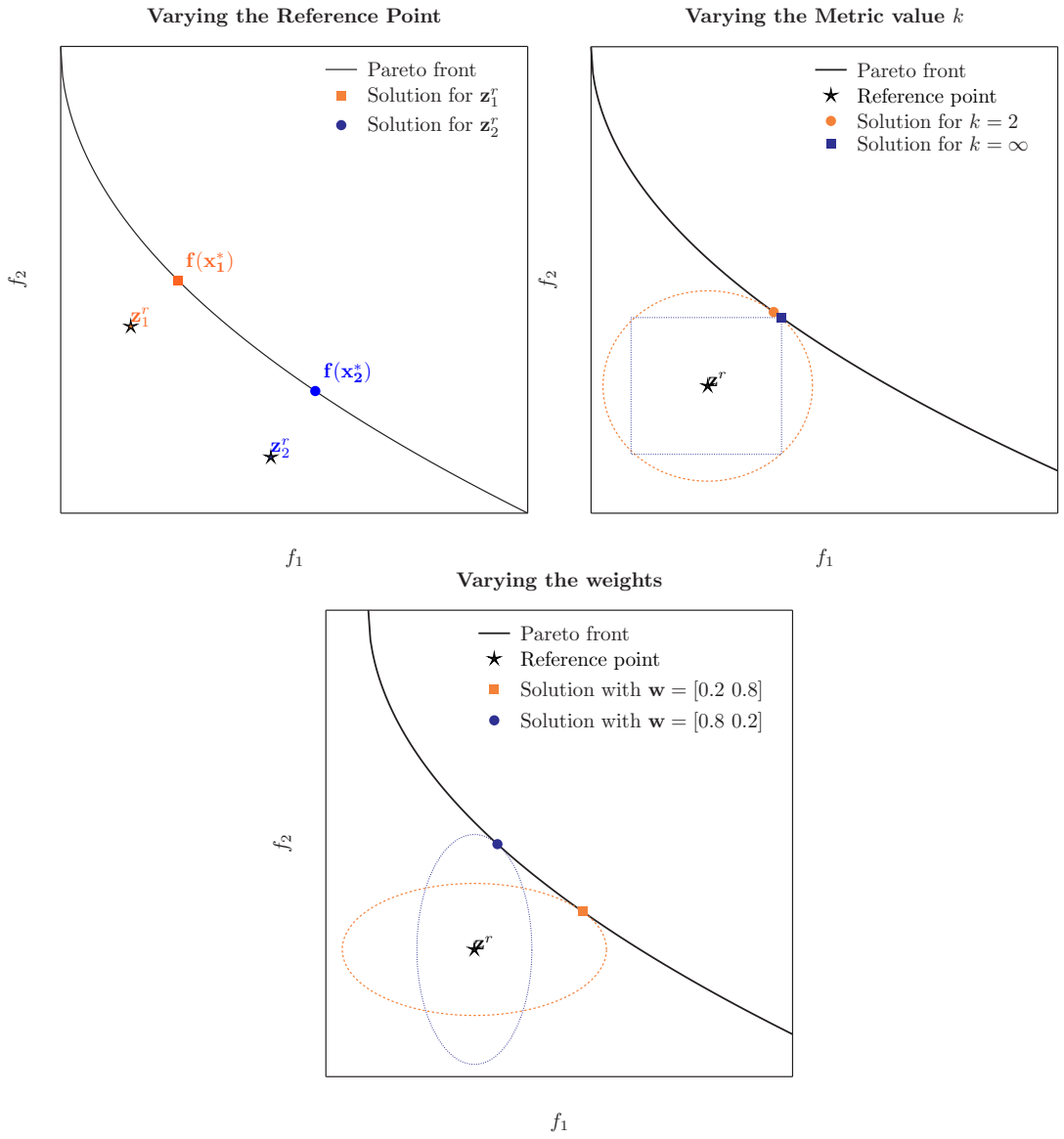


FIGURE 2.12: Generating different solutions in the Weighted Metrics method. Top left: changing the reference point and keeping the rest constant. Top right: using a different metric. Bottom: varying the weights.

this infinite norm excels: if $1 \leq k < \infty$, then solutions in the non-convex region of the front are not obtained, *even if an ideal reference point is used.* And if $k = \infty$, any solution can be generated in any part of the front - but be prepared to get some weak optimal ones.

Finally, if $\mathbf{z}^r = \mathbf{z}^\star$ or better and the Tchebycheff metric is used (or any other and the front is convex), all minimal points can be found by varying the weights. This simply means that the same issues of controlling the location and uniformity of the solutions of the weighting method apply here as well.

In summary, this is a quite simple method with a very attractive way of thinking - by setting targets. It is a pity that you cannot choose any aspiration levels desired because they may generate dominated points, and even with a suitable reference point you have to choose between getting efficient points but missing non-convex parts of the front, or generating solutions in any portion of this region but receiving some weakly efficient as a bonus. These were the main reasons that prevented this method from becoming a legend. But since this philosophy of choosing a reference point is very useful in the Decision Making literature, an improvement of the weighted metrics will be provided in section 3.2.2.1.

Another method with a similar idea is *goal programming* [6]. Here, the aspiration levels are called *goals*, but the same concept of getting close to these goals is used, and, unfortunately, the same drawbacks discussed before are also present. There is, however, a workaround in [14].

#### 2.4.1.4 Normal-boundary Intersection Method

If you read the previous methods, you saw that scalarizing methods are not very good in controlling uniformity of solutions in the optimal front. That is, even equally spaced weights (or grid, in the $\epsilon$-constraint method) are not enough to assure a good spacing of the points. The present method was proposed to overcome exactly this issue.

The method is called Normal-Boundary Intersection (NBI) [11], and, according to its authors, it is capable of generating, from a set of uniform weights, a uniform set of points in the Pareto front *independently of the relative ranges of the functions*. Since the mathematical formulation is a bit harder than the previous techniques, I will give an intuitive idea first.

This method assumes you have the knowledge of the ideal solution $\mathbf{z}^\star$, but, fortunately the Nadir point is not required. Assume the first objective $f_1(\cdot)$ achieves its minimum in $\mathbf{x}_1^*$, $f_2(\cdot)$ in $\mathbf{x}_2^*$, and so on. Then, draw a hyper-plane with vertices in the points $\mathbf{f}(\mathbf{x}_1^*)$, $\mathbf{f}(\mathbf{x}_2^*)$, $\ldots$, $\mathbf{f}(\mathbf{x}_m^*)$. In two dimensions, this is a line like the one shown in the left in Figure 2.13.

Now the idea is to generate, inside this hyper-plane, "auxiliary points" and start to look for optimal solutions with a given direction (see Figure 2.13). The problem is converted into an optimization constrained in a line (in the objective space). For each auxiliary point, a different sub-problem is solved and a distinct solution is obtained. If these references are equally spaced , the solutions will be as well, according to the authors [11].

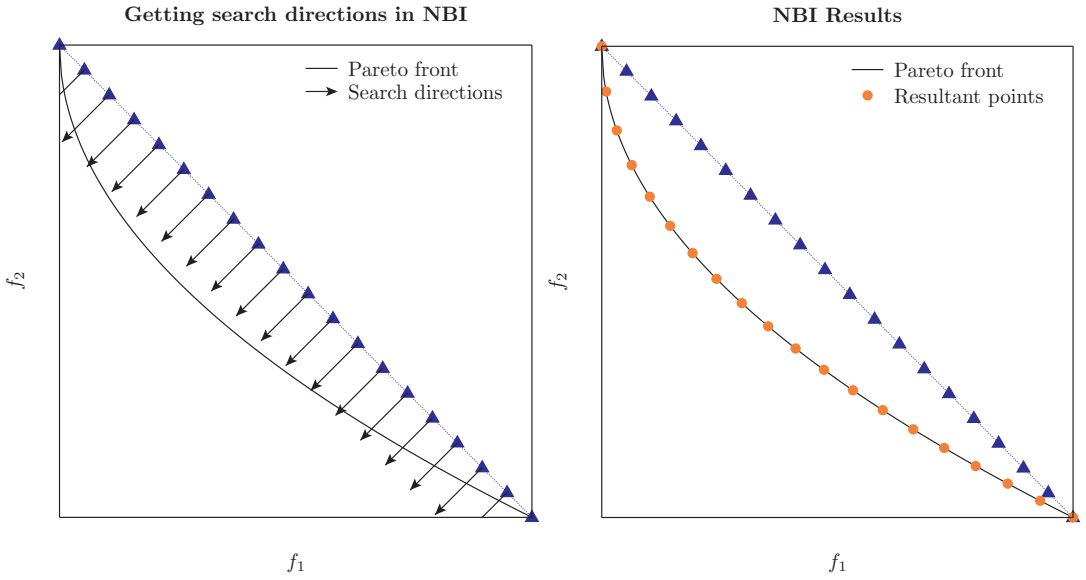


FIGURE 2.13: Solving a two-objective problem with NBI. On the left, the hyper-plane, which is simply a line here, is shown with its "auxiliary points". Also, the arrows indicate the direction of search. On the right, the resultant points. Notice how well spaced these solutions are.

Let us get to the mathematical formulation. If $\mathbf{z}^\star$ is the ideal solution, the *convex hull* of the individual minimum is written as $\mathbf{\Phi w}$, wherein $\mathbf{\Phi}$ is the $m \times m$ matrix

$$\mathbf{\Phi} = [\mathbf{f}(\mathbf{x}_1^*) - \mathbf{z}^\star, \mathbf{f}(\mathbf{x}_2^*) - \mathbf{z}^\star, \ldots, \mathbf{f}(\mathbf{x}_m^*) - \mathbf{z}^\star]$$

where, remember, $\mathbf{x}_i^*$, $i = 1, 2, \ldots, m$ is the point where $f_i(\cdot)$ achieves its minimum. The subtraction with $\mathbf{z}^\star$ is just to shift the origin to this point. The (column) vector $\mathbf{w} = [w_1 \ w_2 \ \ldots \ w_m]^T$, such that $\sum_{i=1}^m w_i = 1$, is a vector of weights. In simple words, for each set of weights, the product $\mathbf{\Phi w}$ generates a point inside the hyper-plane described before. For two objectives, we have seen that the small trick of setting $w_1 = w$, $w_2 = 1 - w$ and letting $w$ vary from 0 to 1 yields uniform weights. For more than three dimensions, the authors also present a technique to generate a diverse set of points inside the hyper-plane, so check [11] if you desire.

Then, for each auxiliary point, the scalarized problem searches for the feasible point with the maximum distance from it along the normal pointing towards the origin, like shown in Figure 2.13. The formulation of the method is

$$\begin{aligned} \text{maximize} \quad & t \\ \text{subject to} \quad & \mathbf{\Phi w} + \hat{\mathbf{n}} t = \mathbf{f}(\mathbf{x}) \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.10}$$

wherein $\hat{\mathbf{n}}$ is the vector (usually unitary) with the direction of search, and the additional constraint is just to assure the search to be in this given direction. The solution to this problem yields the maximum $t$ and the optimum $\mathbf{x}^*$. In the original formulation, $\hat{\mathbf{n}}$ should point from the auxiliary point $\mathbf{\Phi w}$ to the origin. Since this is usually difficult to achieve, the authors propose the use of the *quasi-normal* vector $\hat{\mathbf{n}} = -\mathbf{\Phi 1}$, where $\mathbf{1} = [1\ 1\ \ldots\ 1]^T$ is a $m$-vector of ones (it can be made unitary by dividing it by $\sqrt{m}$). This quasi-normal vector has the good property of not being dependent on the relative scales of the functions.

Now it is time to answer those three questions of the beginning. The third question about the control of the location of the solutions is already answered: uniform weights yield uniform solutions, so the problem of getting unpredictable points for small perturbations is alleviated. Also, all of the efficient points of the front can be found. The main problem is: there is no way of telling if the solution obtained with problem (2.10) is efficient, unless the front is convex. In the more general case, you may get the efficient points, but also the weakly efficient, and even some dominated ones. Take a look at Figure 2.14 for an example. Of course, if the method is used to map the whole efficient front, the dominated solutions can be filtered by comparing them with the points found previously.

If you expected this last method to be the one which beats all of the previous ones, you are probably disappointed knowing that it does not guarantee optimality. For that, I have two things to say. First, if this were true, there would be no reason to present any other method, or even to write this work, because the best method was already available. Second, there is a work in [15] which tries to improve the NBI in order to prevent the generation of dominated points, so check it if you desire.

#### 2.4.1.5 Review of Scalarizing Methods

This section presented methods that convert the original multi-objective function into a scalar one, so the usual tools for single-objective optimization can be employed. When setting new parameters for the scalarization, a possibly distinct solution is found each time.

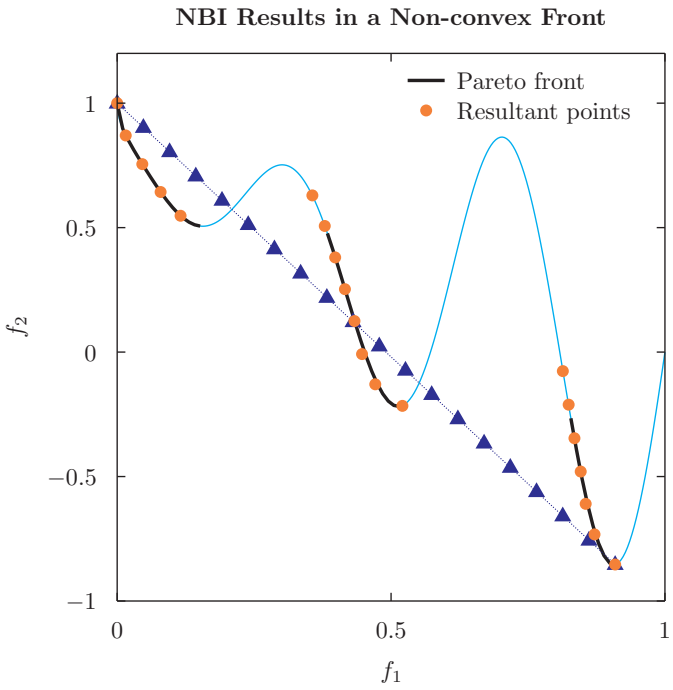


FIGURE 2.14: NBI solving a problem with a non-convex front. The points inside the black lines are efficient, and the rest is dominated. Some auxiliary points of the line generate minimal solutions, but others yield dominated ones.

Depending on the method, the solution for the converted problem is normally weakly efficient, but, if some conditions are satisfied or more tests are performed, this solution can be minimal or even properly minimal.

The biggest advantage of these techniques is the possibility to use scalar-based algorithms. The main drawback is the necessity of setting parameters for the conversion, and the difficulty of controlling the location of the solutions in the Pareto front (a possible exception is the NBI method, but it can find dominated points in compensation).

### 2.4.2 Deterministic Methods with no Scalarization

To be honest, these methods have no classification name, so I decided to use this rather long title to indicate with no doubt what they do. As the name implies, they use deterministic methods and don't require the vector function to be converted into a scalar. Because of that, the usual assumptions that the objective functions are continuously differentiable are required, but in compensation the convergence to a point satisfying first-order necessary conditions for efficiency can be guaranteed.

Recall the steps presented in a typical deterministic method for single-objective optimization [2]. Given an actual point $\mathbf{x}^k$ in the $k$-th iteration, the point $\mathbf{x}^{k+1}$ of the next iteration is found by:

1. From $\mathbf{x}^k$, compute a descent direction $\mathbf{s}^k$, that is, one that reduces the value of the objective function;
2. Calculate a step size $\alpha^k$ that minimizes $f(\cdot)$ in this direction, that is, find $\alpha^k$ such that $f(\mathbf{x}^k + \alpha^k \mathbf{s}^k)$ is minimum;
3. Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{s}^k$, augment the iteration counter and repeat the process.

These steps are performed until a stopping criterion is satisfied, which can be the satisfaction of the Karush-Kuhn-Tucker conditions, the current value of the objective is reasonable, the number of iterations have achieved a maximum etc.

In order to extend these ideas to the vector case, it is necessary to specify how to set a descent direction (step 1) and to perform the line search (step 2) for more than one function. And, just like in the scalar case, different methods choose distinct directions. For example, in the single-objective case, the steepest descent uses $\mathbf{s}$ as the negative of the gradient such that $\mathbf{s} = -\nabla f$; Newton's Method goes further with the second order using also the inverse of the Hessian, so $\mathbf{s} = -\mathbf{H}^{-1}\nabla f$; some derivative-free methods introduce a perturbation in the current point and select as a direction the one that gives the greatest improvement [16]; and a lot of other possibilities. For the multi-objective problem, there is not yet a great deal of techniques, and the only ones I am aware are the extensions of the *Steepest Descent* [17] and of *Newton's method* [18]. The first one will be briefly described here in order to illustrate the process.

#### 2.4.2.1 Computing a Descent Direction

Take a look at the left of Figure 2.15, where it is shown the level curves of a quadratic function, its minimum $\mathbf{x}^*$ and the current point $\mathbf{x}^k$. From this point, all of the directions of search $\mathbf{s}^k$ that result in a point inside the region **I** will increase the function, while any direction inside the region **II** will decrease it[11]. Therefore, any direction inside **II** can be considered a descent one, and each method has its own way of choosing it. Of course some directions are better than others - in fact, if the objective function has a minimum value, there is a direction $\mathbf{s}^\star$ and a step size $\alpha^\star$ that goes directly from $\mathbf{x}^k$ to $\mathbf{x}^*$, but we usually don't know which are they. The best we can do is find a good $\mathbf{s}^k$ and a $\alpha^k$, and if the new point is not the minimum, repeat the process.

Now consider the right picture for the multi-objective case. The regions **I** and **II** have similar interpretation as before, but now in the first region *both* objectives get worse,

[11] Of course, this reasoning is valid if a suitable step size is used. If, for instance, you choose a direction almost parallel to the boundary of the regions, you will end up increasing the objective value. Even if you choose a direction straight from $\mathbf{x}^k$ to $\mathbf{x}^*$, too big values of $\alpha^k$ will make you surpass the optimum and the objective value will end up increasing again.

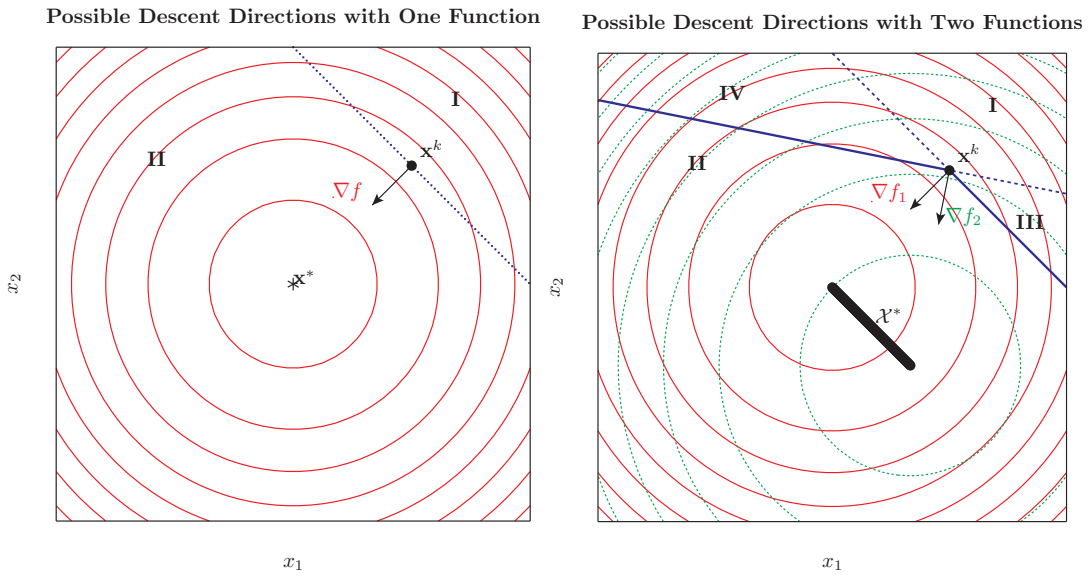


FIGURE 2.15: Possible descent directions in a single and in a two-objective problem. In the left figure, all of the directions that point inside the region **II** result in a descent movement. In the right figure, the same reasoning applies, and both objectives are increased or decreased. The regions **III** and **IV** represent directions of diversity, where one objective improves but the other gets worse.

and in **II**, *both* improve. As the point $\mathbf{x}^k$ progresses in direction to the efficient solutions, this region shrinks, reaching zero when $\mathbf{x}^k$ is Pareto optimal. This is expected, since, by definition, in an efficient solution it is not possible to improve all of the objectives at once.

In the vector case there is the emergence of two new regions, indicated by **III** and **IV** in the right of Figure 2.15. Directions inside them improve some objectives but decrease others; in region **III** $f_1$ is decreased but $f_2$ increased, and vice-versa in region **IV**. These zones are called *diversity regions* [19], and are better explored by Evolutionary Algorithms (section 2.4.3) in order to provide more diverse solutions. They can also be useful when you already reached the efficient front and want to look for alternatives.

Even though this was explained for two objectives, the same applies to more objectives. There are still regions **I** and **II** where all objectives are deteriorated or improved, respectively, but the rest of the space is composed of diversity regions. When the number of objectives grows, region **II** gets smaller quickly. That is one of the reasons why some algorithms that do not use the first order information (gradient) have trouble in finding the efficient set in higher dimensional problems.

The steepest descent of Fliege [17] aims at finding a direction $\mathbf{s}^k$ inside **II**. Assume

$$\nabla \mathbf{f}(\mathbf{x}) = [\nabla f_1(\mathbf{x}) \ \nabla f_2(\mathbf{x}) \ \ldots \ \nabla f_m(\mathbf{x})] \in \mathbb{R}^{n \times m}$$

is the gradient of the vector function $\mathbf{f}(\cdot)$ computed at the point $\mathbf{x}$, obtained by stacking each individual gradient column-wise. The gradient of the $i$-th function in the point $\mathbf{x}$ is given by the $n$-length vector

$$\nabla f_i(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_i}{\partial x_1} \\ \frac{\partial f_i}{\partial x_2} \\ \vdots \\ \frac{\partial f_i}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n$$

wherein $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T$ and $n$ is the number of variables, or dimension of the problem.

Now, the direction $\mathbf{s}^k$ will be the value of $\mathbf{s}$ that solves the following single-objective optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_x(\mathbf{s}) + \tfrac{1}{2}\|\mathbf{s}\|^2 \\ \text{subject to} \quad & \mathbf{s} \in \mathbb{R}^n \end{aligned} \tag{2.11}$$

where the function $f_x(\mathbf{s})$ is defined as[12]

$$f_x(\mathbf{s}) = \max_{i \in \{1,2,\dots,m\}} \left\{ \left( \nabla \mathbf{f}(\mathbf{x_k})^T \mathbf{s} \right)_i \right\}$$

In order to understand equation (2.11), assume a descent direction $\mathbf{s}$. Hence, $\nabla f_i^T \mathbf{s}$ gives the decrease we would get in the $i$-th function if this direction was chosen, so it is usually a negative number. The function $f_x(\mathbf{s})$ then represents the value of the worst reduction. What the equation (2.11) attempts to do is to find a direction that minimizes this worst decrease. Notice that this reduction is only local, just as the steepest method for single-objective optimization. The solution to this problem gives the descent direction $\mathbf{s}^k$ of the current iteration.

There is a reformulation of (2.11) to remove the non-differentiability:

[12] In some studies the Jacobian is used in place to the gradient, and it is given by $\mathbf{J}(\mathbf{x}) = \nabla \mathbf{f}(\mathbf{x})^T$.

$$\begin{aligned} &\text{minimize} && \gamma + \tfrac{1}{2}\|\mathbf{s}\|^2 \\ &\text{subject to} && \left(\nabla \mathbf{f}(\mathbf{x}^k)^T \mathbf{s}\right)_i \leq \gamma, \quad i = 1, 2, \ldots, m \end{aligned} \tag{2.12}$$

which is now a convex quadratic problem with linear inequality constraints.

It can be shown [17] that when $\mathbf{x}^k$ is efficient, then the solution to (2.11) is $\mathbf{s}_k = \mathbf{0}$, and $\gamma = 0$ if (2.12) is used instead. This is expected with the intuition, because there is no direction to improve all objectives if you are already in the Pareto front. Also, it can be used as a stopping condition of this method. Finally, the same paper [17] discusses some other methods to find a descent direction.

#### 2.4.2.2 Computing the step size

So you have computed a direction $\mathbf{s}^k$ capable of improving all of the objectives. The next phase is to find an optimal step size $\alpha^k$. The problem with this part is that, because of the nature of vector-valued functions, there will not be usually a single minimum in this direction like there is in the scalar case. An example is shown in Figure 2.16, where two quadratic functions are being optimized. The left panel shows the objectives plotted as a function of the step size $\alpha$, and the right one illustrates one objective against the other. The dark line indicates the optimal points in this direction. In the right figure, from the starting point $\alpha_0$, any step inside the interval $[\rho_a, \rho_b]$ will yield an optimal point[13], but any step in $(\alpha_0, \alpha_d)$ will dominate the initial point; the values in $[\alpha_d, \rho_b]$, even if optimal, will not dominate the initial, then they may not be viewed as an improvement.

So, the optimal $\alpha^*$ must be the one such that which condition is satisfied? One of the objectives is minimized (in this direction)? Or any objective attains its minimum? The new point dominates the initial? There is still some room to answer that, and I will mention two possibilities from the literature.

In the same paper that proposed this algorithm [17], an Armijo-like rule is adopted. Starting from $\mathbf{x}^k$ and $\alpha = 1$, compute the objective value in $\mathbf{x}^k + \alpha \mathbf{s}^k$ and check if the following condition is satisfied:

$$\mathbf{f}(\mathbf{x}^k + \alpha \mathbf{s}^k) \leq \mathbf{f}(\mathbf{x}^k) + \beta \alpha \nabla \mathbf{f}(\mathbf{x}^k)^T \mathbf{s}^k \tag{2.13}$$

[13] Notice that this "optimal set" is only optimal in a given direction $\mathbf{s}^k$. They are not necessarily efficient for the original problem.

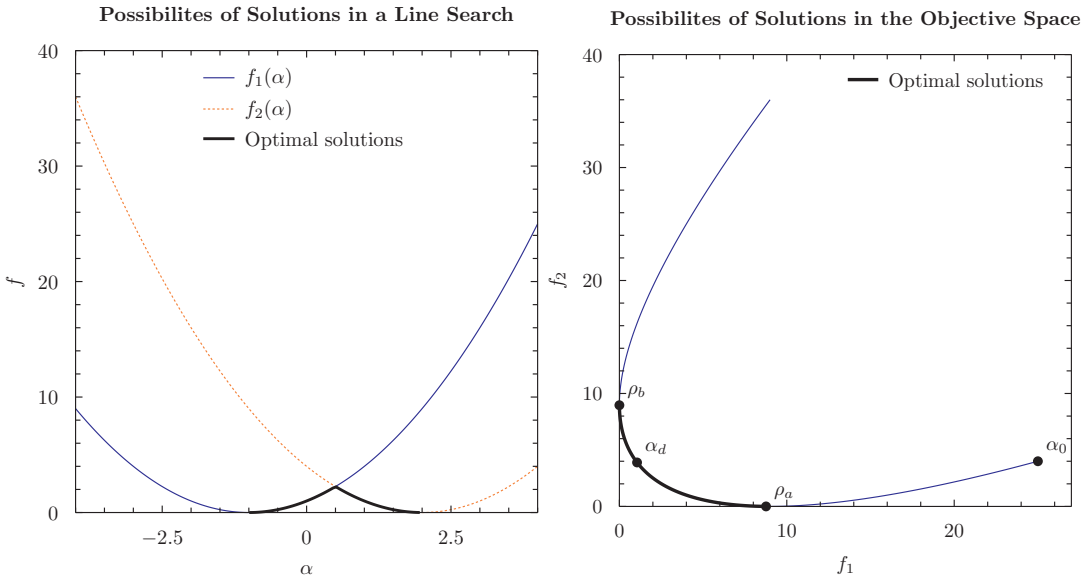


FIGURE 2.16: Possible optima in a multi-objective line search. On the left, two functions with their efficient set (in the given direction $\mathbf{s}^k$), and, on the right, one function plotted against the other.

wherein $\beta \in (0, 1)$ is a pre-specified constant and the inequality is taken to be componentwise. If this condition is not satisfied, set $\alpha \leftarrow \alpha/2$ and try again until a positive result arises. In [17] it is shown that this procedure will stop after a finite number of tries. This method aims at giving a point inside the interval $(\alpha_0, \alpha_d)$, so that the new point dominates the old one.

Another possibility of line search is the *multi-objective golden section*, presented in [20]. In the scalar case, given an interval $[a, b]$, $a < b$ such that $f(\mathbf{x}^k + \alpha^k \mathbf{s}^k)$ is uni-modal, two new points inside it are generated, $\alpha_1 = a + \Phi^2(b - a)$ and $\alpha_2 = a + \Phi(b - a)$, wherein $\Phi = (\sqrt{5} - 1)/2 \approx 0.6180$ is called *golden ratio*. Then, according to the objective values in each of these four points, the interval where the optimal is not present is removed (see [2] for more details), a new smaller interval $[a', b']$ is generated, and the process starts again (see Figure 2.17). This goes on until the interval is smaller than a given tolerance, and the optimal step is assumed to be its mean point.

The multi-objective golden section is similar in the idea of interval reduction, but instead of comparing the objective values, a dominance relation is used instead. The technical details can be seen in the original paper [20]. When compared to the Armijo-like rule, it is shown that the golden section can yield solutions inside the interval $[\rho_a, \alpha_0)$ (see Figure 2.16), so the convergence is usually faster.

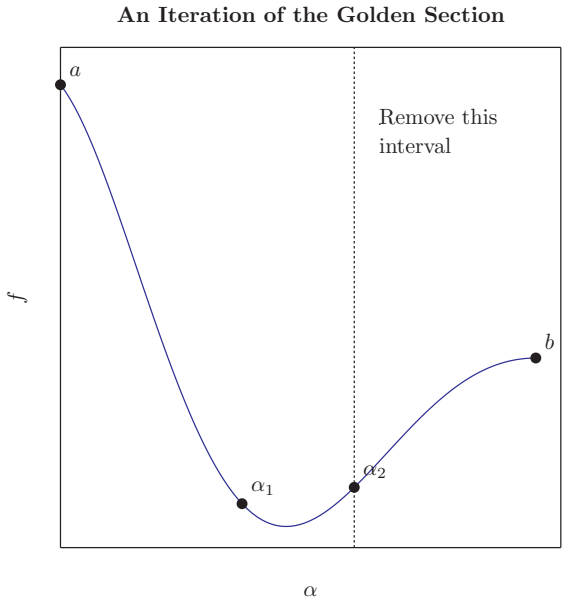


FIGURE 2.17: An example of an iteration of the scalar golden section. Given the interval $[a, b]$ where the function is uni-modal, two points $\alpha_1$ and $\alpha_2$ are computed. In this case, $f(\alpha_1) < f(\alpha_2) < f(b)$, so the optimum must be inside $[a, \alpha_2]$. The interval $[\alpha_2, b]$ is removed, and, in the next iteration, $b$ becomes $\alpha_2$, $a$ keeps its old value, two new points $\alpha_1$ and $\alpha_2$ are computed and the process is repeated.

#### 2.4.2.3 The final algorithm

The complete steepest descent can be described as:

1. Set the iteration counter $k = 0$, and choose an initial point $\mathbf{x}^0$;
2. Compute a descent direction $\mathbf{s}^k$ using (2.11), (2.12) or any other similar approach;
3. Check if the current point is efficient. Normally, you can check if $\mathbf{s}^k$ or $\gamma$ is smaller than a given tolerance. If it is, stop the algorithm. Else, go to step 4;
4. Compute a step size $\alpha^k$ according to the Armijo-like rule, the Multi-objective Golden Section, or any method of your taste;
5. Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{s}^k$;
6. Augment the iteration counter $k \leftarrow k + 1$ and go back to step 2.

This algorithm works directly in the original problem, with no need to transform it like the scalarization procedures do. Because of that, if a point $\mathbf{x}^*$ is proved to be optimal, then it is indeed an efficient point. Keep in mind that, since it is deterministic, the objectives must satisfy some conditions like differentiability and unimodality in order to yield (global) minimal solutions. Notice that depending on the algorithm used, the

scalar methods of the previous section require these same conditions. If you use a scalar Quasi-Newton to solve, say, a weighted multi-objective problem, if the functions are not uni-modal and continuous, the solution will possibly not be efficient.

Another characteristic of the multi-objective steepest descent is that there is no need to set any parameter like scalarization methods need. This can be viewed as an advantage, but also as a drawback. On the plus side, you simply run the algorithm and you get an efficient point (provided the functions satisfy the conditions already mentioned). On the other side, you have absolutely no control of where this point is located. The only way of getting different solutions is by modifying the initial point, and even so its final location is a mystery. Therefore, this algorithm may be well suited in applications where a single minimal solution is required, or maybe as a starting point of other methods, but it is probably not useful if you desire a specific part of the Pareto front, or if you trying to approximate the whole region.

Finally, there is a similar extension of the Newton's method to the multi-objective case in [18]. This field is still open to new methods and solutions for the aforementioned issues.

### 2.4.3 Evolutionary Algorithms

Evolutionary Algorithms (EA) are a subfield of *evolutionary computation* [21], and they became very famous in the 90s for solving a lot of difficult real-world problems [22]. They are methods inspired in biology. Because of that, many different studies describe them using some biological terms. In my opinion, for some algorithms, these terms can facilitate the understanding, but sometimes they complicate, or even move the attention away from the subject that is the optimization. Because of that, I will try to simplify matters in the following discussion, and use only the intuitive terms. Interested readers can find a good introduction about EAs in [21], [23] and [24].

Contrasting with the deterministic methods, in which one point is used and it is "evolved" until hopefully the optimum is found, EAs employ a set of points, which collectively form a *population*. Assume $\mathcal{P}(t)$ represents the population in iteration $t$, which comprises $\mu$ candidate points, also called *individuals*, $\mathbf{x}_1$, $\mathbf{x}_2$, $\ldots$, $\mathbf{x}_\mu$. This set is also called *population of parents*. From these, another set $\tilde{\mathcal{P}}(t)$ of $\lambda$ tentative points, called *offspring*, $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$, $\ldots$, $\tilde{\mathbf{x}}_\lambda$, is created by some kind of perturbation. Then, the individuals of both populations are combined somehow into another set of candidates, $\mathcal{R}(t)$ and, from this

set, $\mu$ individuals are selected to compose the population of the next iteration[14], $\mathcal{P}(t+1)$, and the process is repeated.

The whole procedure described in the last paragraph was very vague, mostly because there are endless possibilities of generating the offspring, combining them with the parents and selecting the new points for the next iteration. In the literature, these steps are accomplished by the so-called *operators*. The most used are:

- **Variation operators:** they are responsible for generating the offspring of iteration $t$, $\tilde{\mathcal{P}}(t)$. When a new point $\tilde{\mathbf{x}}_i \in \tilde{\mathcal{P}}(t)$ is created with the assistance of parents [i.e., individuals from the current population $\mathcal{P}(t)$], like a weighted sum, the operator is named *reproduction* or *crossover*. If $\tilde{\mathbf{x}}_i$ is constructed using a perturbation, like adding a noise from a specific probability distribution to its coordinates, the operator is called *mutation*. Sometimes both operators can be applied in sequence.

  The reproduction operator usually creates new individuals that are close to its parents, so they can "fine-tune" the search within a given neighborhood. Mutation, on the other side, is normally responsible for creating larger random changes in the candidates, so basically any point of the search space can be generated. Together, they provide the exploration of the space in the quest for the optimal point of efficient set.

- **Selection operators:** they decide which are the most promising individuals generated in a given iteration. So, their job is to guide the population in the direction to the minimal solutions. In order to accomplish that, each individual receives a value indicating its fitness, which may be its objective value or something related to that, and the candidates with better fitness have greater chances to proceed to the next iteration.

A pseudo-code for a generic Evolutionary Algorithm is shown in Algorithm 1. The initial population is usually created at random in the search space, but if some knowledge about promising regions is available before-hand, then it could be used to improve the process.

The control parameters depend on each algorithm, and the most common are the probability of an individual to suffer reproduction, the distribution of the perturbation used in the mutation etc. The performance of an algorithm can vary for different values of these parameters, so it is important to set them to suitable values.

[14] This assumes that the population size is to be kept constant for the whole optimization process, which is a common assumption. Anyway, this is not an obligation. The algorithm proposed in [25], for instance, allows the population to grow with basically no limits.

Initialize the iteration counter $t \leftarrow 0$;
Initialize the population $\mathcal{P}(0)$ with $\mu$ points;
Initialize the control parameters;
**Result**: Optimal Value $\mathbf{x}^*$ or Pareto set approximation $\mathcal{P}^*$
**while** *Stopping condition not met* **do**
  Evaluate the fitness $f(\mathbf{x}(t))$ of each point $\mathbf{x}(t) \in \mathcal{P}(t)$;
  Perform the variation operator to create the offspring $\tilde{\mathcal{P}}(t)$ and compute their fitnesses ; `// Variation`
  Combine both sets into another one $\mathcal{R}(t)$;
  Choose the points for the next population, $\mathcal{P}(t+1)$ ; `// Selection`
  $t \leftarrow t + 1$;
**end**
Assign to $\mathbf{x}^*$ the point $\mathbf{x} \in \mathcal{P}(t)$ with the best objective value (single-objective case), or return the whole population as approximation to the efficient front;

**Algorithm 1:** Pseudo-code for a basic Evolutionary Algorithm.

Another relevant topic is the stopping condition [21]. It is what tells that the algorithm has achieved a satisfactory result, or that the resources are over, like if the number of function evaluations reaches its limit. For the scalar case, one can use the gradient of the function, so that if (usually its norm) it is smaller than a given tolerance, the solution is close enough to the minimum. If the derivatives are not available or not used, which is normally the case in EAs, other tests can be checked. For instance, if the difference in the best and worst individuals is smaller than a defined value, like $10^{-5}$, it means that the population has converged to a small enough region. Another option is to check if the best value achieved does not improve after a given number of iterations, case where no further progress should be expected. These pre-defined parameters may also influence the result, so it is common to combine them with a maximum number of iterations or function evaluations.

For the vector case, the only stopping criterion adopted is normally the shortage of resources. The other tests have usually no meaning because, since a set of different efficient points is expected, there is no reason to hope that the population will converge to a smaller region. There is, however, a first attempt to a different stopping condition in [26] using Kalman filters, but this will not be taken further in this work.

While the variation operators can be used in the same way both in the single and in the multi-objective cases, the selection needs to be adapted. In the first situation only one point is usually sought - the minimum. Then, each individual has a better fitness depending on how smaller its objective value is, and the selection can prefer these points. The process can take a form exemplified in Figure 2.18: the points start disperse in the search space, and, as soon as the iterations pass, the selection prefers

the ones closest to the optimal, the population begins to converge and starts to gather around its neighborhood.

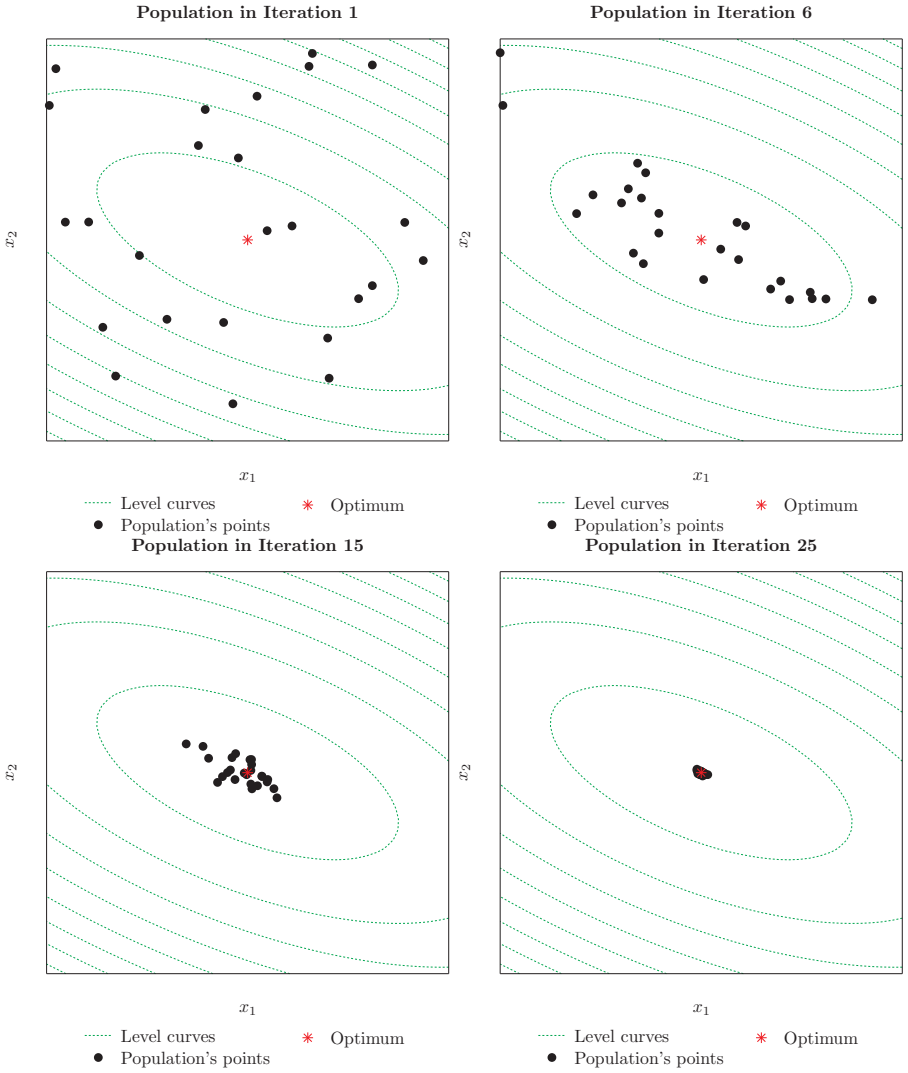


FIGURE 2.18: Typical behavior of an EA solving a single-objective problem. In the beginning, the points are spread around the feasible region, and then, when the iterations pass, they start to gather close to the optimum. In this example, after 50 iterations, the best and the worst individual differed for less than $10^{-5}$ of objective value.

In the vector case, instead of running the algorithm many times to get one different efficient point in each execution, since EAs already work with sets of points, it is more plausible to use them to approximate regions of the Pareto front, or even its whole extension, in one single run, like the one shown in Figure 2.19. But, because of that, the fitness attributed to the points and the comparisons among them must be made such that non-dominated individuals have equal fitnesses [24].

In [24] some different attempts to perform selection are shown in the course of time since the first multi-objective EA. The algorithms of interest here are the ones that

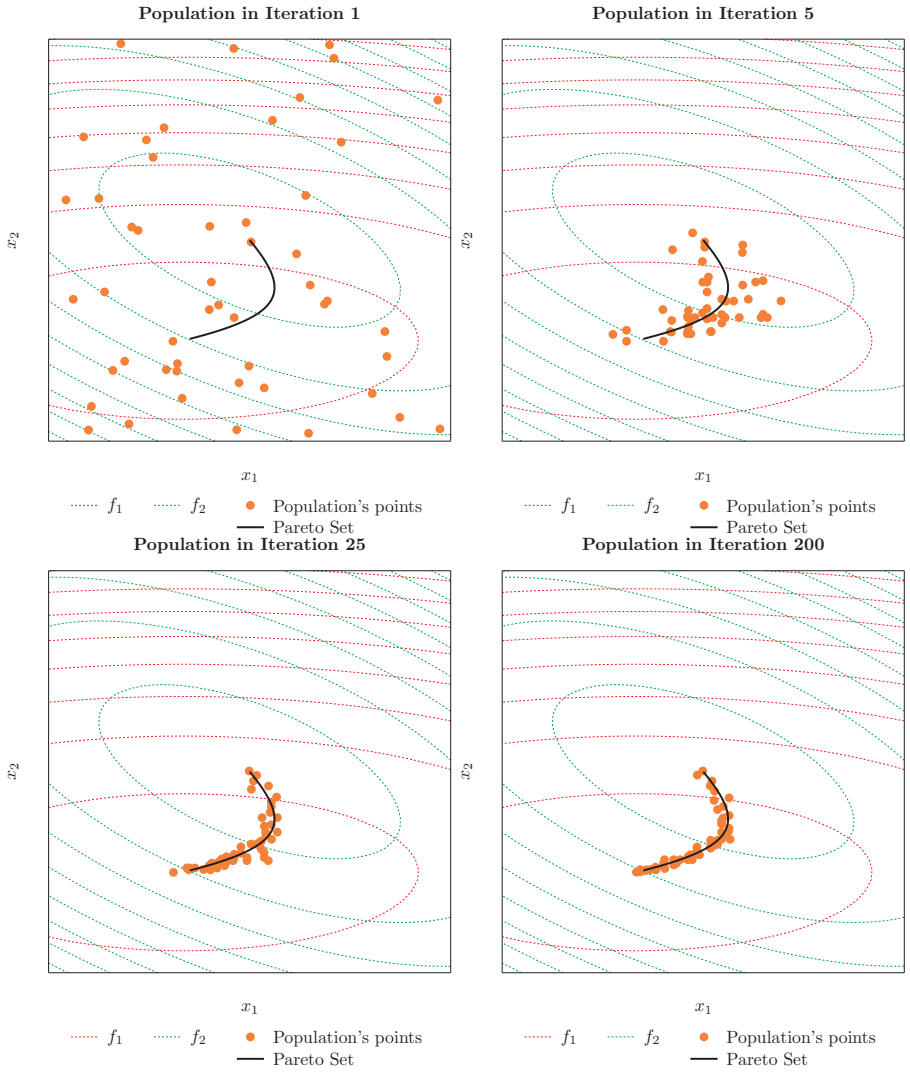


FIGURE 2.19: Typical behavior of an EA solving a two-objective problem. As with the scalar case, the points start spread in the feasible region, and then start to converge not to a single point, but to a set of points, possibly very close to the Pareto set/front.

follow the (Pareto-) dominance principle to assign the fitness. The main representations of this class are the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [27] and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [28]. In the following I will describe two methods for performing selection in multi-objective EAs: the *non-dominated sorting* procedure and the use of *indicators*.

#### 2.4.3.1 Non-dominated Sorting

The non-dominated sorting was proposed by Deb et al. in [27] with the NSGA-II in mind, but it can be applied in a lot of different algorithms. Assume, in the $t$-th iteration, the populations of parents and offspring were combined in a population $\mathcal{R}(t)$, with

cardinality $\mu \leq |\mathcal{R}(t)| \leq 2\mu$, which depends on the algorithm. So, the selection must be such that only $\mu$ individuals are chosen.

The non-dominated sorting consists of clustering the points in ranks of dominance. Given the population $\mathcal{R}(t)$, compute the non-dominated elements and give them the rank 1. Then, remove these points, and find the new non-dominated points, labeling them with the rank 2. This process is repeated until all elements receive a rank. This process is illustrated in Figure 2.20. The points of each rank are sometimes said to belong to a "front of dominance".

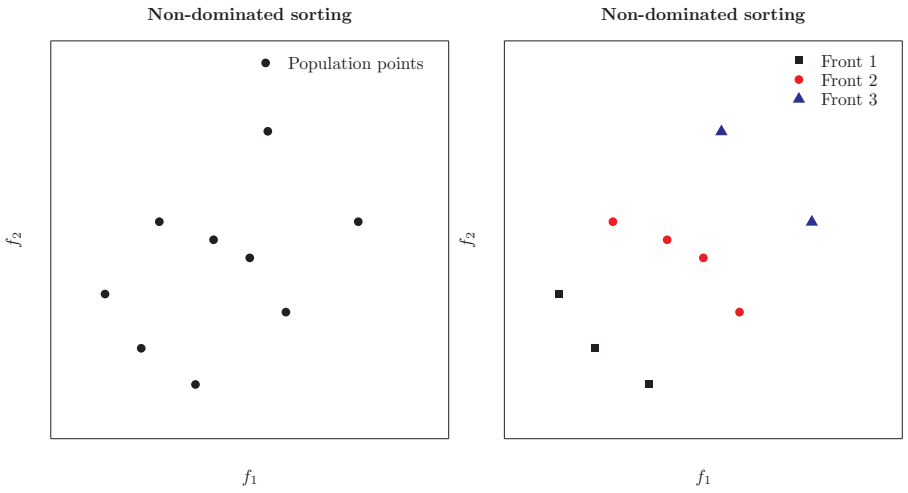


FIGURE 2.20: Example of non-dominated sorting, assuming minimization of both objectives. Given a set of points in the objective space (figure in left), the non-dominated sorting consists of ranking the solutions according to fronts of dominance. In the right panel, the points are classified in three fronts.

Then, start filling the next population $\mathcal{P}(t+1)$ with points from the first rank, then from the second, until, say, in the $k$-th front, there are more elements than required to get $\mu$ individuals. Since they all belong to the same rank, they all deserve the right to go the next population. How to truncate these points in a fair way?

The idea is to use a second criterion in order to distinguish the points of the same front. Given that the goal of the NSGA-II proposed in [27] is to give a good approximation of the whole efficient front, the second criterion can be to prefer individuals that give a more diversified population. This diversity can be promoted in the decision or in the objective space, but the latter is normally the preferred alternative.

The diversity here is computed using a method called *crowding distance*, proposed in the same paper [27]. For the $i$-th point in the objective space, find its closest neighbors and draw a hyper-parallelepiped using them as vertices. Its crowding distance value will be (an approximation of) the perimeter of this hyper-surface (see Figure 2.21). The

extreme points receive a value of infinity. Notice that the crowding distance is valid only for points belonging to the same front.

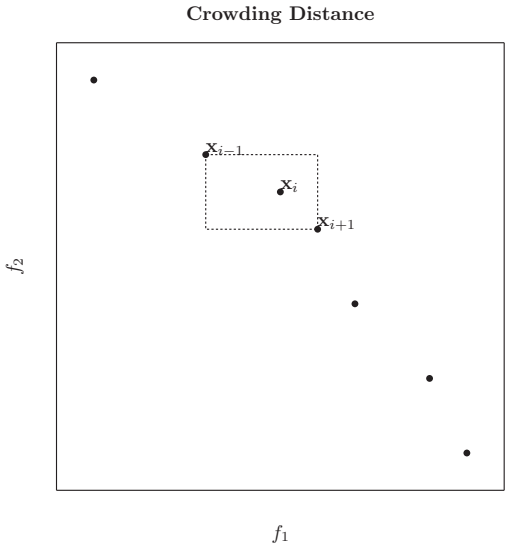


FIGURE 2.21: Computation of the crowding distance in two objectives. Its value for each point is the perimeter of the rectangle formed using its neighbors as vertices in the objective space.

Assuming $\mathcal{R}^{*,k}$ represents the $k$-th front, the crowding distance of the point $\mathbf{x} \in \mathcal{R}^{*,k}$ is given by

$$c(\mathbf{x}, \mathcal{R}^{*,k}) = \sum_{i=1}^{m} \frac{c_i(\mathbf{x}, \mathcal{R}^{*,k})}{z_{i,max} - z_{i,min}} \tag{2.14}$$

wherein $z_{i,max}$ and $z_{i,min}$ are (estimates of) the maximum and minimum values of the $i$-th objective, respectively, and

$$c_i(\mathbf{x}, \mathcal{R}^{*,k}) = \begin{cases} \infty, \text{ if } f_i(\mathbf{x}) = \min\limits_{\mathbf{x}' \in \mathcal{R}^{*,k}} \{f_i(\mathbf{x}')\} \text{ or } \max\limits_{\mathbf{x}' \in \mathcal{R}^{*,k}} \{f_i(\mathbf{x}')\} \\ \min\{f_i(\mathbf{x}'') - f_i(\mathbf{x}') \mid \mathbf{x}', \mathbf{x}'' \in \mathcal{R}^{*,k} \backslash \{\mathbf{x}\} : f_i(\mathbf{x}') \leq f_i(\mathbf{x}) \leq f_i(\mathbf{x}'')\}, \text{ otherwise} \end{cases}$$

The crowding distance is such that points in a more crowded region receive smaller values, and since they contribute less to the overall diversity, they should be pretermited to individuals with higher values. Therefore, if one uses as fitness of a point its dominance rank and its crowding distance value as a tie breaker, a total pre-order is obtained. The complete method to select the new population $\mathcal{P}(t+1)$ becomes:

1. Rank each individual of $\mathcal{R}(t)$ in fronts of dominance;

2. Start to fill $\mathcal{P}(t+1)$ with points from the first front $\mathcal{R}^{*,1}$, then the second, $\mathcal{R}^{*,2}$, until the $k$-th front $\mathcal{R}^{*,k}$ has more elements then necessary;

3. Compute the crowding distance value of each individual in $\mathcal{R}^{*,k}$ and remove the point with the worst value. Repeat the process[15] until the size of $\mathcal{R}^{*,k}$ is equal to the required to complete $\mu$ elements in $\mathcal{P}(t+1)$.

This method is described with the aim of approximating the whole efficient front with a good uniformity. In terms of computational cost, the order of complexity of the non-dominated sorting is $\mathcal{O}(mN^3)$, wherein $m$ is the number of objectives, and $N$ is the number of elements in $\mathcal{R}(t)$, but in [27] a *fast non-dominated sorting* procedure is described with complexity $\mathcal{O}(mN^2)$. The crowding distance has $\mathcal{O}(mN' \log N')$, wherein $N'$ is the number of points in the $k$-th front.

#### 2.4.3.2 Indicator-based Evolutionary Algorithms

The aim of a multi-objective EA is usually to return a set of points that give the best representation possible of the Pareto front. This "best representation" is usually translated as "minimum distance to the optimal front and maximum diversity on it", but it has no good mathematical translation yet. But what if there was a function which could take the whole population and return a number giving its quality? In that way, the optimization problem could be converted to an optimization of this function.

These kinds of functions already exist, and they are called *indicators* [29]. They were first used to measure the quality of the results of different algorithms for comparison purposes, but Zitzler et al. in [24] had the idea of using them to guide the optimization process.

An $N$-ary indicator $I$ is a function $I : \mathcal{X}^N \mapsto \mathbb{R}$, that is, it receives $N$ sets of points (which can be singletons) $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_N$, with $\mathcal{P}_i \subseteq \mathcal{X}$, for $i = 1, 2, \ldots, N$, and maps then into a real number $I(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N)$. The most important indicator used here is the binary $I(\mathcal{P}_1, \mathcal{P}_2)$, which tells how better a set is with regard to the other.

An example of an indicator is the $\epsilon$ additive $I_{\epsilon,+}$. For two points $\mathbf{x}_1$ and $\mathbf{x}_2$, it is defined as

$$I_{\epsilon,+}(\mathbf{x}_1, \mathbf{x}_2) = \min_{\epsilon}\{f_i(\mathbf{x}_1) - \epsilon \leq f_i(\mathbf{x}_2) \text{ for } i = 1, 2, \ldots, m\} \tag{2.15}$$

[15] It is important to point that, in the original NSGA-II proposal, the crowding distance is applied only once and the individuals with worst values are removed. However, the reapplication of this process can yield more diverse solutions, so it is adopted here.

In words, it is the smallest $\epsilon$ such that $\mathbf{f}(\mathbf{x}_1)$ weakly-dominates $\mathbf{f}(\mathbf{x}_2)$. See Figure 2.22 for an example. It also shows that, in general, $I(\mathbf{x}_1, \mathbf{x}_2) \neq I(\mathbf{x}_2, \mathbf{x}_1)$. With the $\epsilon$-indicator, the smaller its value, the best, so, here, $\mathbf{x}_1$ would be considered better than $\mathbf{x}_2$, even though they are non-dominated according to Pareto-dominance. Some other indicators are the multiplicative $\epsilon$-indicator and the binary hyper-volume. See [29] for more details and possibilities.

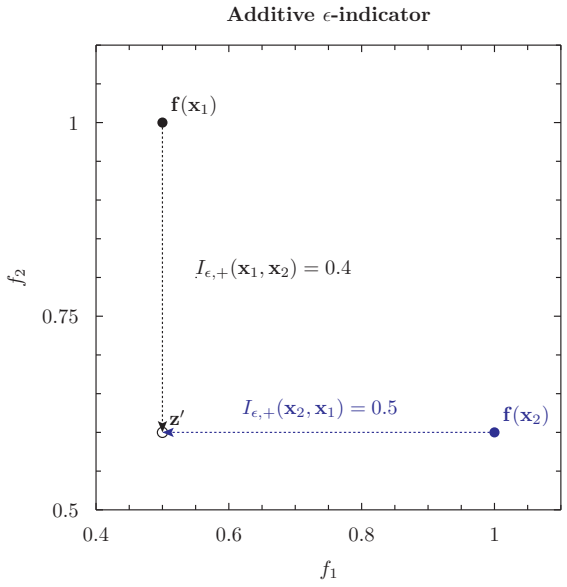


FIGURE 2.22: Example of the additive $\epsilon$-indicator. In order to $\mathbf{x}_1$ weakly dominate $\mathbf{x}_2$, it must "go the point" $\mathbf{z}'$, that is, its second objective value must be decreased by 0.4. This is the indicator value of $I_{\epsilon,+}(\mathbf{x}_1, \mathbf{x}_2)$. Now, for $\mathbf{x}_2$ dominate $\mathbf{x}_1$, it also needs to go to $\mathbf{z}'$, but now, its first objective must be decreased by 0.5, so $I_{\epsilon,+}(\mathbf{x}_2, \mathbf{x}_1) = 0.5$. Notice that these indicators have different values.

So, given a binary indicator $I$, the original problem can be converted into finding the population $\mathcal{P}$ which optimizes the value $I(\mathcal{P}, \mathcal{X}^*)$, wherein $\mathcal{X}^*$ is the efficient set. Sure, the true optimum is not known, so the last affirmative was just for formalization. I will now describe the *Indicator-Based Evolutionary Algorithm*.

### Using the Indicators in an EA

Suppose you are in the selection operator of an EA and have the same combined population $\mathcal{R}(t)$ with $\mu \leq |\mathcal{R}(t)| \leq 2\mu$. In order to choose the best $\mu$ elements to go the next population $\mathcal{P}(t+1)$, there is the need to assign a fitness to the individuals. The previous method used the non-dominated sorting followed by a diversity measure to do the job. Here, a previous selected binary indicator $I$ will be used.

Given an individual $\mathbf{x}_i \in \mathcal{R}(t)$, its fitness $\varphi(\mathbf{x}_i)$ will be combination of its indicator values with all of the other points $\mathbf{x}_j \in \mathcal{R}(t)$, $j = 1, 2, \ldots, |\mathcal{R}(t)|$, and $j \neq i$. The simplest combination is a sum, so

$$\varphi(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{R}(t)|} I(\mathbf{x}_j, \mathbf{x}_i) \tag{2.16}$$

However, in [24], they propose a slight variation of (2.16) that emphasizes the influence of dominating points over the dominated ones:

$$\varphi(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{R}(t)|} -e^{-I(\mathbf{x}_j, \mathbf{x}_i)/\kappa} \tag{2.17}$$

wherein $\kappa > 0$ is a scaling factor. Its value may depend on each problem, but if the indicators are normalized, it can be fixed, for instance, as $\kappa = 0.05$ in many problems with good results [24].

Now that a fitness assignment was described, the selection procedure becomes:

1. Given the population $\mathcal{R}(t)$, compute the fitness $\varphi(\mathbf{x}_i)$ for all $\mathbf{x}_i \in \mathcal{R}(t)$ according to equations (2.16) or (2.17);
2. Find the point $\mathbf{x}_{worst}$ with the worst fitness value:
   - If the indicator used $I$ is to be minimized, then the worst point will have the smallest value of $\varphi$;
   - If the indicator is to maximized, then the worst point will have the greatest $\varphi$.
3. Remove $\mathbf{x}_{worst}$ from $\mathcal{R}(t)$;
4. Repeat the process until $|\mathcal{R}(t)| = \mu$. When that happens, $\mathcal{P}(t+1) = \mathcal{R}(t)$.

Algorithms that follow this scheme are called Indicator-Based Evolutionary Algorithms (IBEAs). There are various indicators that can be used, and the results are usually dependent on them. Sure, since an indicator was just defined here as a "function that maps $N$ sets (or points) into a real number", there must be another condition they must satisfy in order to be useful. For that, an indicator $I$ must be *dominance preserving* [24]. For binary indicators, this is expressed as[16]

[16] These expressions are valid if the indicator should be minimized. In case of maximization, one needs to invert the inequalities.

- If $\mathbf{x}_1 \prec \mathbf{x}_2$, then $I(\mathbf{x}_1, \mathbf{x}_2) < I(\mathbf{x}_2, \mathbf{x}_1)$; and
- If $\mathbf{x}_1 \prec \mathbf{x}_2$, then $I(\mathbf{x}_3, \mathbf{x}_1) \geq I(\mathbf{x}_3, \mathbf{x}_2)$, for all $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{X}$.

Now that the basics of Evolutionary Algorithms for single and multi-objective optimization are explained, I will now describe the method used in this work. First, its original formulation for the scalar case will be presented, and then an extension to vector-valued functions will take place.

#### 2.4.3.3 Single-objective Differential Evolution

The Differential Evolution (DE) is a very simple EA proposed by Storn and Price [30] to solve continuous optimization problems. Following the analogy presented before, in iteration $t$, each parent $\mathbf{x} \in \mathcal{P}(t)$ will produce one offspring $\tilde{\mathbf{x}} \in \tilde{\mathcal{P}}(t)$. They are then compared with each other in order to produce the next population. In what follows, its operators will be described and later everything will be put together.

##### Mutation

For the $i$-th parent $\mathbf{x}_i$, a so called *trial vector* $\mathbf{u}_i$ will be produced. For that, choose randomly two individuals from the population $\mathbf{x}_{i2}$ and $\mathbf{x}_{i3}$, and form a difference vector $\mathbf{x}_{i2} - \mathbf{x}_{i3}$. Then, pick a third point (also randomly) among the parents, $\mathbf{x}_{i1}$, called *base vector*, and add to it a scaled version of the difference vector:

$$\mathbf{u}_i = \mathbf{x}_{i1} + F(\mathbf{x}_{i2} - \mathbf{x}_{i3}) \tag{2.18}$$

wherein $F \in \mathbb{R}$ is one of the control parameters called *scale factor*. Even if it can assume any value, $F \in [0, 1]$ is suggested. Also, in principle, the three random individuals can be any, but it is advisable [21] that they are different among themselves, *and* different from the current parent $\mathbf{x}_i$.

##### Recombination

Given the $i$-th parent, $\mathbf{x}_i$, and the $i$-th trial vector, $\mathbf{u}_i$, the $i$-th offspring $\tilde{\mathbf{x}}_i$ is created variable by variable by choosing a coordinate sometimes from the parent, sometimes from the trial vector. The $j$-th component of the offspring will be

$$\tilde{x}_{i,j} = \begin{cases} u_{i,j} & \text{if} \quad U_j \leq CR \\ x_{i,j} & \text{otherwise} \end{cases}, \quad j = 1, 2, \ldots, n \tag{2.19}$$

wherein $U_j$ is a Random Variable with uniform distribution over the interval $[0, 1]$, and $CR \in [0, 1]$ is another control parameter called *crossover factor*. Bigger values of $CR$ implicate offspring with more components of the trial vector, and vice-versa for smaller values. It is recommended that at least one coordinate of $\tilde{\mathbf{x}}_i$ is due to $\mathbf{u}_i$ to prevent mere copies, and this approach is followed here.

This recombination is called *binomial*. There is another one, called *exponential*, which will not be of interest here. More details about it can be found in [21].

#### Selection

The selection is simple: the $i$-th parent $\mathbf{x}_i$ is compared to the $i$-th offspring, created using the two aforementioned operators, and the best one goes to the next population. Assuming a minimization problem, the next $i$-th parent of the $(t+1)$-th iteration will be

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{x}_i(t) & \text{if} \quad f(\mathbf{x}_i(t)) < f(\tilde{\mathbf{x}}_i(t)) \\ \tilde{\mathbf{x}}_i(t) & \text{otherwise} \end{cases} \tag{2.20}$$

Note that, if they have the same objective value, the offspring is chosen to promote diversity.

The complete algorithm is given in Algorithm 2. It is referred here as "basic Differential Evolution", because there are lots of variations that can be adopted in the above operators. But since this "basic" version is already very good for various applications [21] and for the purposes of this work, it will be the one used.

### 2.4.3.4 Multi-objective Differential Evolution

Thanks to the popularity of the scalar Differential Evolution, this algorithm received some attention in order to extend it to the multi-objective case. A survey of some of these instances can be found in [31]. I am going to present very briefly here the Differential Evolution for Multi-objective Optimization (DEMO), first introduced in [32].

```
Initialize the iteration counter t ← 0;
Initialize the population P(0) with μ points;
Initialize the control parameters F and CR;
Result: Optimal Value x*
while Stopping condition not met do
    P(t+1) ← ∅;
    foreach x_i(t) ∈ P(t) do
        Evaluate the fitness f(x_i(t)) of each point;
        Create the trial vector u_i(t);                          // Mutation
        Create the offspring x̃_i(t);                          // Recombination
        if f(x_i(t)) < f(x̃_i(t)) then                          // Selection
            x_i(t+1) ← x_i(t);
        else
            x_i(t+1) ← x̃_i(t);
        end
        Include x_i(t+1) in P(t+1);
    end
    t ← t+1 ;
end
Assign to x* the point x ∈ P(t) with the best objective value
```

**Algorithm 2:** Pseudo-code for the basic Differential Evolution.

Once the basic single-objective version is understood, this instance is very easy to grasp. The variation operators of mutation and crossover are identical to the ones used in the scalar case. The selection, however, needs to be changed.

The parent/offspring comparison is still adopted. But here, the Pareto-dominance is used to indicate if an improvement, a deterioration or neither has happened. So, for the $i$-th parent $\mathbf{x}_i$ and its offspring $\tilde{\mathbf{x}}_i$ in the iteration $t$, the selection starts like:

- If $\mathbf{x}_i \prec \tilde{\mathbf{x}}_i$, include $\mathbf{x}_i$ in $\mathcal{R}(t)$;
- If $\tilde{\mathbf{x}}_i \prec \mathbf{x}_i$, include $\tilde{\mathbf{x}}_i$ in $\mathcal{R}(t)$;
- If they are incomparable, include *both* in $\mathcal{R}(t)$.

It can be easily seen that the size of $\mathcal{R}(t)$ will be $\mu$ if there are no dominance ties, $2\mu$ if every parent is incomparable to its offspring, and something in between in the general case. Excluding the first case, one has to truncate $\mathcal{R}(t)$ in order to keep the optimization process. I have described two possibilities: using the non-dominated sorting with crowding distance and employing indicators. The original paper [32] proposes the first method, but the second can be used as well. In this work, I will use these two instances.

#### 2.4.3.5 Review of Evolutionary Algorithms

The Evolutionary Algorithms (EA) are very attractive meta-heuristics for solving optimization problems when they don't satisfy the conditions for deterministic methods. Some of the advantages of EAs are [23]:

- They don't require continuity, differentiability or smoothness of the functions;
- Convexity or connectedness of the efficient front are not necessary for their application;
- EAs can be applied in multi-modal functions, with less chances of getting stuck in local minima like the deterministic algorithms;
- They can deal efficiently with constraints;
- If desired, they are able to approximate a region of the Pareto front (or even its whole extension) in only one execution.

Some of its drawbacks are:

- Because they are meta-heuristics, there is no guarantee that the final solution will be optimal or efficient;
- EAs require the setting of control parameters, and their performance can be very dependent on them. They are usually very problem oriented, and even for a given problem it is not easy to find the best adjustment.

The EAs are suitable for approximating portions of the efficient front with one execution, but what is commonly done in practice is to try to approximate the whole optimal set. Because of their set-based property, they can provide better control of the diversity of the solutions, and thus became popular alternatives for many problems.

As a final note, it is important to realize that when the objective functions do satisfy the deterministic methods' conditions, they are more indicated to do the job [2, 3], because of their guarantee of optimality and efficiency. EAs are an awesome tool for solving optimization problems, because they exchange the impossibility of finding the best solution with the big chances of getting something good enough, possibly very close to the optimal. But, if with some more research or analysis the problem can satisfy the deterministic techniques' needs, then they are clearly better options.

#### 2.4.3.6 Interlude

Multi-objective Evolutionary Algorithms, mainly the Pareto-based ones (like NSGA-II and SPEA2), were very successful when solving a lot of real world problems (see [22] for some examples), such that in some of these works they were called *state of the art*.

This reign of happiness was due mainly because they were living in a world where everything could be seen: a world where the problems had two or maybe three objectives only. In theory, these algorithms are prepared to handle an arbitrary number of objectives. However, in practice, what happened was that the usual EAs were not able to get close to the optimal solutions (of course, this observation was made when the true front was known, like in benchmark problems). Some other problems became evident, like the impossibility of *seeing* the solutions in order to check if the outcome is nice enough, or even to choose a final solution. Moreover, some fronts grow (in size) when a new objective is added, and one would need greater population sizes in order to approximate the whole region, increasing the computational cost.

All of these problems created a new sub-field of multi-objective optimization, called *many-objective optimization*. Chapter 4 elaborates better what kinds of problems arise when the number of objectives $m$ grows.

## 2.5 Summary

Decision theory deals with the choice of a solution among different alternatives, such that a human decision maker (DM) gets the most satisfaction possible from it. When the problem has only one criterion $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ to be satisfied, the decision making process can be solved by a *single-objective optimization*, which aims at finding the alternative $\mathbf{x}^* \in \mathcal{X} \subseteq \mathbb{R}$ with the optimum (minimum) objective value.In this case, the DM does not need to get much involved in the process, unless he wants to perform some post-analysis on the final alternatives.

If the problem requires more than one criterion to be described, it can be solved by means of a *multi-objective optimization*, where now a vector function $\mathbf{f}(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ needs to be minimized. Now, instead of a (possibly single) optimal solution, there are *efficient* (or Pareto-optimal) solutions, which can be more than one, or even infinite, and collectively form the efficient set $\mathcal{X}^*$ in the variable space and the efficient front $\mathcal{Z}^*$ in the objective space.

Together with the Pareto-optimality, two more concepts of efficiency were described: the weak and the proper efficiency. The first comprises the points that do not have

a candidate with better fitness in all objectives, while the second represents solutions that are efficient but that provide a finite trade-off. These two are more convenient for computational purposes, since some methods are usually able to find weak Pareto-optima, but under some conditions, minimal or proper minimal solutions can also be found. In what follows, it will be assumed that the DM cares about efficient solutions (so properly efficient are allowed).

I have shown in this chapter three classes of algorithms for computing minimal solutions: scalarizing methods (section 2.4.1), deterministic techniques that do not scalarize the problem (section 2.4.2), and evolutionary algorithm (section 2.4.3). Among them, the last algorithms will be of main interest in this work.

As mentioned in the chapter, just finding efficient points is not enough to solve a multi-objective problem, because, in the end, only one or maybe a few will be implemented in practice. Therefore, these problems require a greater participation of the DM in the selection of a final solution, that is, he needs to use his preferences in order to pick the best alternative. While this chapter was dedicated for computing minimal points, the next one will deal with the process of selecting a final solution and expressing the DM's desires in the optimization process.

# Chapter 3

# Decision Making in Multi-objective Optimization

It was seen in the last chapter that the sentence "minimize a function over all possible feasible alternatives", mathematically represented by equation (2.3), lacks meaning when the function is vector-valued. The concept of optimum is converted into efficiency, and, in place of optimal solutions, we start to look for efficient solutions.

However, this does not solve the decision making problem. After all, even when restricting the attention to only the minimal solutions, they may still come in a huge number, possibly infinite, and only one or maybe a few of them are implemented in practice. Therefore, one needs a person, more precisely, a *decision maker* (DM), possibly assisted by an *analyst* [12], to decide which of these solutions will be the final solution. Then, the first sentence of the last paragraph can be translated into "find a feasible point(s) that give(s) the most preferred outcome vector(s)" [5]. Whilst the last chapter showed how to find efficient solutions, this chapter will occupy on the task of expressing the preferences in order to tell what is a "most preferred outcome".

When the whole decision process of computing minimal points and choosing the final solution is taken into account, it is said that we are in the field of *multi-criteria decision making* (MCDM) [33]. It is, however, difficult to separate it from multi-objective optimization (MOO) because, in the literature, these terms are sometimes used alone with no reference to each other, sometimes as synonyms, and once in a while MOO as a sub-field of MCDM. To clarify matters, I will use the definition presented in [5]: multi-criteria decision making is used to broadly describe all decision making problems in which multiple measures of solution quality exist. It is further divided into (i) multi-objective optimization and (ii) *multi-attribute utility theory* (MAUT). The first refers to

problems with a large number of feasible alternatives, while the second indicates problems with a smaller number. We say that a set is large if the analyst considers it too big to totally enumerate it. With that classification, the workout example, since it has a small number of alternatives, can be considered to belong to the MAUT field, while other problems, like the continuous ones, belong to the MOO area.

The main reason why I consider this classification is that it allows multi-objective optimization to be viewed as an extension to the single-objective case. They both share the property of receiving a large (possibly infinite) feasible set and narrowing it into a smaller set (possibly singleton) of recommended solutions. With that in mind and with the problems considered in this work, MOO and MCDM can be viewed as synonyms.

## 3.1 Decision Theory

### 3.1.1 Basic concepts of Decision Theory

As was already stated, the Decision Theory provides a general framework for choosing between alternative courses of action when the outcomes resulting from this choice are imperfectly known [34]. In order to understand it better, some basic concepts are required. Consider the following decision making problem.

You are hungry at your home, and decide to prepare an omelet using three eggs. The first two eggs were broken successfully directly into the frying pan, but, for the third, you start to distrust this action, because, if this last egg is rotten, you end up with no omelet. You have, then, three possible actions to follow: (i) break the egg directly into the pan; (ii) break it into a separate cup to inspect its quality; and (iii) throw it away without checking. For each action, the outcomes are conditioned by the events (a) the egg is fine and (b) the egg is rotten. Table 3.1 summarizes the results for all of these possibilities.

This simple example introduces some concepts very common in decision making problems, that are the *alternatives*, *outcomes* and *states of nature* [1].

The alternatives were already described: they are the possible courses of action that are available to the DM at the time of decision. Depending on the situation, the whole feasible space $\mathcal{X}$ is at disposal for inspection, but, when its cardinality is too big (or maybe infinite), it is restricted to some points only. In the example above, the possible alternatives are to break the egg directly into the frying pan, inspect it first, or just throw it away.

| | **Break egg directly into pan** | **Inspect it first in a cup** | **Thrown third egg away** |
|---|---|---|---|
| **Egg is good** | Three-eggs omelet | Three-eggs omelet, but one more cup to wash | Two-eggs omelet, but one good egg lost |
| **Egg is rotten** | No omelet, and two good eggs lost | Two-eggs omelet, but one more cup to wash | Two-eggs omelet |

TABLE 3.1: Decision making example for the omelet problem. The columns show the available actions to be taken, and the rows, the possible events to happen. The entries are the outcomes.

The *states of nature* represent the factors that are outside the DM's control. The quality of the egg is usually not in the cook's hands. Together with the alternatives, they define the outcome of the DM's decision. These outcomes are indicated in Table 3.1 for the omelet example. Another situation would be your decision to bring or not a coat in a trip. If it is cold in your destination, the result is a warm person if you bring it, and a possible sickness otherwise. But, if it is warm, taking the coat would mean more things to carry.

These states of nature normally happen in a certain way. Because of that, the outcomes usually have a *probability of occurrence* associated. Depending on these probabilities, the decision making process is said to be under [1]:

- *certainty*, if each action leads exactly to a specific outcome. If the egg is known to be rotten, we know exactly what will happen for each alternative chosen;
- *risk*, if each action leads to possible determined outcomes, and each one has a chance, i.e., a probability to happen. Moreover, this probability is known by the DM. For instance, in a game that involves the toss of a fair coin, the probability of heads is 0.5;
- *uncertainty*, just like the decision under risk, but now the probabilities are not known or maybe not even meaningful.

The last two are not exactly exclusive, as, for example, a person who has no idea if it is going to rain today may check the weather forecast, and then, even if the exact probability is not know, he has an idea of the possibilities.

The decision making under uncertainty is actually scary, so the analysts normally try to search for more information in order to at least assess estimates of the probabilities.

I will not present these techniques because they do not concern this work. If you are interested, check references about decision making theory like [1] and [34].

When making a decision, it is usual to organize this information in a tabular form, like shown in Table 3.1. This is called *decision matrix* [1]. Once the probabilities of the outcomes are assigned, the next step is to evaluate your degree of satisfaction with each outcome. This is done by using the concept of *utility function*.

#### 3.1.1.1 Utility theory

Given two outcomes of a problem, it is possible to compare them, that is, a DM is able to tell which one he prefers, or if he is indifferent about them. For instance, I prefer chocolate over tomatoes, but I am indifferent between tomatoes and blackberries, because I dislike them both.

Thanks to this, it is possible to attribute a number indicating how much something pleases you most. In my case, I could write

- Chocolate: 10;
- Tomatoes: 1;
- Blackberries: 1.

and my preferences would be clear.

According to some moral theorists [1], all values can be reduced into a single entity, called *utility*, $u(\cdot) : \mathcal{Z} \mapsto \mathbb{R}$. It is basically a function that receives an outcome $O \in \mathcal{Z}$ and maps it into a scalar, normally a real number, $u(O) \in \mathbb{R}$. This value indicates how much $O$ satisfies a criterion of a decision maker. It may be related to his happiness, monetary value, or virtually anything, and, normally, the higher its value, the best the outcome is for the DM.

Notice that these numbers do not have to be in a specific scale. For example, it does not mean that I prefer chocolate ten times more than tomatoes or blackberries; they are just required for comparisons. The main properties a utility function must follow are [34]

$$u(A) > u(B)$$

if outcome $A$ is preferable to $B$, and

$$u(A) = u(B)$$

if they are equally desirable or undesirable, that is, indifferent.

### Constructing a utility function

There are some methods for creating a utility function. The most straightforward is to check each alternative and outcome and "manually" assign values based on how desirable they are. This is common in opinions surveys, where questions like "Please rate with a number from 0 to 10 how comfortable our toilet seats are" are possible and, even if there are no established indices for coziness, people who used some of them are still able to provide an answer.

There are more operational methods, and I will present briefly one indicated in [35] for the omelet example. First, define which are the most desirable and undesirable outcomes, independent of their probabilities. According to Table 3.1, I would say that getting three-eggs omelet (entry [1,1]) is the best, and no omelet plus the loss of two good eggs (entry [2,1]) is the worst. Then, assign a utility 1 for the first and 0 to the second.

Now, concentrate on the two-eggs omelet, outcome [2,3]. Suppose a genie appears in your house offering you two alternatives: (i) you get the two-eggs omelet, or (ii) he flips a magical coin with probability $p$ of landing heads. If the result is heads, you get the three-eggs omelet (best outcome), but if it is tails, you receive the worst result. What is your choice? If $p$ is, for instance, 0.5 (a fair coin), you may prefer the certain choice, even if it has one egg less. But, as the probability of receiving the three-eggs omelet increases, you may become more interested in the lottery. If for, say, $p = 0.6$ you are willing to take the risk, then your utility of this outcome will be 0.6. This process should be repeated for the remaining outcomes. Notice that, the most desirable they are, the higher the probability of success should be so you could trade the sure but not best result to a lottery between the best and the worst.

The references [5] and [34] show other methods for assessing utility values for the outcomes. For further details, check the relevant technical literature.

### Utility theory in MCDM

When your problem has multiple criteria, each outcome will be a vector of objectives. In order to assign a utility value to this whole vector, the DM usually creates a utility

function for each criterion separately (either by means of the procedure described above or any other), and then they are combined into a single value.

The most common aggregation is an additive one [5], so the utility of the outcome $\mathbf{f}(\mathbf{x})$ is

$$u(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{m} w_i u_i(f_i(\mathbf{x})) \tag{3.1}$$

wherein $w_i$ is the weight of the utility of the $i$-th function. Notice that if each utility were directly proportional to its objective, that is, if it were a linear function of it, the equation (3.1) would reduce to a linear aggregation, or the weighting method[1] in equation (2.6).

Other aggregation of the utilities can be seen, for instance, in the references [8] and [5].

### 3.1.2 Solving a Decision Problem

Once the alternatives and the outcomes of a decision problem are organized, and their respective probabilities of occurrence and utilities are assigned, there are some methods for picking a final solution in the literature. The most popular is known as *Maximum Expected Utility* [1].

In order to illustrate it, assume the utilities of the outcomes of the omelet example were computed in a suitable way, and the probability that the egg is rotten is[2] 0.1. Table 3.1 then is converted into Table 3.2.

| | **Break egg directly into pan** | **Inspect it first in a cup** | **Thrown third egg away** |
|---|---|---|---|
| **Egg is good** (0.9) | 1 | 0.9 | 0.4 |
| **Egg is rotten** (0.1) | 0 | 0.5 | 0.6 |

TABLE 3.2: Omelet example of Table 3.1 changing the outcomes by their utilities and assigning probabilities to the events "rotten egg" and "good egg'.

The expected utility maximization implies that the best decision is the one with the best mean utility, such that this value is weighted by the probabilities of the states of the nature. Therefore, the utilities of each action would be:

[1] This is another reason why the linear aggregation is not very recommended: our preferences are not usually linear. In the workout routine example, the smaller the value of Maria's body fat, the better. However, too low indices could make her become malnourished, and therefore would not be preferable.

[2] This could be assessed, for example, by previous experience, counting the number of times you encountered a rotten egg.

$$\begin{aligned} \text{Break egg directly into pan:} \quad & 0.9 \times 1 + 0.1 \times 0 = \mathbf{0.9} \\ \text{Inspect it first in a cup:} \quad & 0.9 \times 0.9 + 0.1 \times 0.5 = 0.86 \\ \text{Throw the egg away:} \quad & 0.9 \times 0.4 + 0.1 \times 0.6 = 0.42 \end{aligned}$$

Therefore, the best solution in this case would be to break the egg directly into the frying pan. Of course, for different probabilities and utilities the best alternative would possibly be different.

Virtually *any* decision problem can be solved by this method [35]. It does, however, have some critiques for not fully explaining how people actually *do* their decisions [36]. For example, if this method was always followed, people would only consider lotteries and gambling when their probability of winning is high enough, but if it was like this, how to explain the popularity of gambling with thousands of people? This and other reasons influenced the creation of other theories, like the *regret theory* [1] and the very famous *prospect theory* [36, 37]. They are, however, not required for this work, so the interested reader can check the references provided here.

### 3.1.3 Choosing a solution in MOO

Decision making under risk and uncertainty is more common in problems of multi-attribute utility theory, where the number of alternatives is not so high. In typical multi-objective problems, it is more common to assume that we work under certainty [5], so the outcomes depend only on the alternatives. This allows us to express $\mathbf{x}$ as alternatives and $\mathbf{f}(\mathbf{x})$ as its outcome as it was already been done.

Under certainty, the decision process is easier: after assigning utilities for the alternatives presented, the DM chooses the candidate with the highest utility. If this value is shared among more than one solution, choose one or a few of them [1]. The main difficulty for applying the methods presented here in a MOO problem is that it usually has too many alternatives. Among them, the DM could focus only on the efficient points, but, even so, this number may be still too high. Therefore, considering the DM wants as a final solution an efficient point (or some of them), the whole decision process can be divided in two steps: finding efficient points, and choosing the most preferred of them. Keep in mind that there is no need to perform these steps in a specific sequence.

In the end, the most used method is to effectively evaluate each solution until the most preferred is found (see some examples of real world problems in [12] and [9]). The DM can also have some tools at disposal, like the visualization of sets of solutions. This is more common in problems with two or even three objectives, and higher dimensional

situations require the use of special techniques. See [12] and [38] as references for these methods.

## 3.2 Solution process in Multi-criteria Decision Making

The complete proceeding of finding a solution in MCDM is called *decision process* [12]. As already stated, this process requires an analyst to, among other things, help the DM into getting efficient solutions, and the decision maker to express his preferences until the final solution is reached.

In this section I will present a classification of methods for solving a MCDM problem, and then I will discuss some ways that the DM can express his preferences to help the analyst in this procedure.

### 3.2.1 Classification of MOO methods

As already stated, the decision making process is divided in finding efficient points and choosing the most preferred of them (which usually means one or a few), and these steps do not have to be taken in a specific sequence. Depending on how the DM organizes the optimization party, the methods for solving MOO are usually classified as [6] *a priori*, *a posteriori*, interactive and non-preference based.

#### 3.2.1.1 *A priori* methods

The Decision Maker expresses his preferences *before* the search for efficient points, so they replace or supplement the Pareto-dominance, inducing a total order in the space instead of the partial order of before. The typical way of doing this is by having a utility function such that, when optimized (usually with scalar-based algorithms), the best solution is achieved.

This is the simplest kind of methods for arriving at a final solution. The difficulty lies at creating a utility function that accurately describes the DM's preferences when he usually is not aware of all of the alternatives. Normally in the end we get a disappointed DM because his desires were too far from what is possible, or he achieves his goals, but with possible better options he wasn't aware of.

It was demonstrated [12] that people are not very good in encoding their preferences into a utility function in these situations, so instead of this function, the DM's preferences

are better expressed as weights, priorities or aspiration levels. See [6] and [12] for a survey of these methods.

*A priori* methods have the desirable property that, once the utility function is defined or other parameters like weights are set, the optimization *per se* is very easy, and the vast body of theory and algorithms of single-objective optimization can be readily adopted. Nevertheless, these techniques are more applied in multi-attribute utility theory, mostly because of the difficulty of evaluating the huge number of alternatives beforehand. Among the methods described in the last chapter, all of the scalarizing ones can also be used as an *a priori* technique if their parameters were given before the optimization.

#### 3.2.1.2 *A posteriori* methods

The DM takes place *after* the optimization. Therefore, in this case, the analyst first computes the whole efficient front, and then the DM decides which are his preferred solutions. When it is infeasible to find all of the minimal points, like when they are infinite, then a *good representation* of it is computed instead. This "good representation" is normally seen as a finite set with points that are the closest possible to the optimal front, and well distributed in order to provide a global overview of it (see Figure 3.1).

For most people, this is a more natural way of thinking: first find efficient solutions, and then let the DM worry about which one to choose. This does not require the existence of a utility function, nor it tries to approximate one. Despite this advantage, it is, in my opinion, an overrated approach.

For two-objective problems, the minimal solutions can be actually seen, and trying to distinguish among, say, 100 points in a curve is not a hard task [5]. Even with three criteria, the visualization is still possible, despite being trickier. But, with four or more, the number of points required to give a "good approximation" may be too large. This means that more computational budget is required. Also, approximating the whole front wastes time with some regions that are of no interest at all. Finally, when you deliver some hundreds of efficient points to the DM, you are not helping him to solve his problem, but creating a different one of choosing among all of these alternatives.

*A posteriori* methods should, therefore, be restricted to a small number of objectives. Among the techniques described in the last chapter, all of the scalarizing methods can be used with the *a posteriori* philosophy by adjusting their parameters to a lot of different combinations. The Evolutionary Algorithms, however, thanks to their better control of diversity and capability of approximating a lot of solutions in a single run, are their best representative.

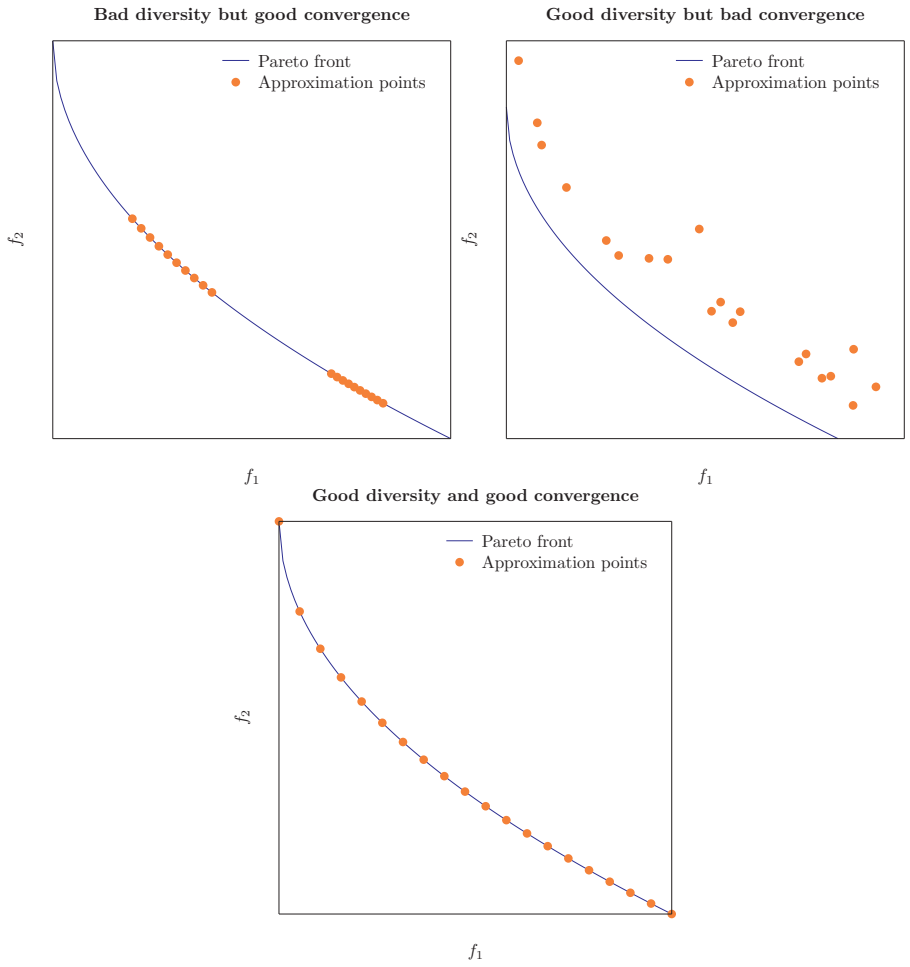


FIGURE 3.1: Possible representations of the efficient front by *a posteriori* methods. In the top left, the points are too concentrated in specific regions, while in the top right, they are far from the efficient front. The figure at the bottom gives a better representation of the front, with good proximity and uniformity.

#### 3.2.1.3 Interactive methods

The interactive methods can be viewed as a balance between the *a priori* and *a posteriori*. Here, the decision maker gives his preferences in parallel with the optimization. It is usually assumed that the DM has a utility function which should be optimized, but it is constructed during the process as he *learns* [12] more and more about the problem. Therefore, they do not require a formidable previous knowledge about the problem like *a priori* techniques do, nor they waste time with undesirable solutions like the *a posteriori* approach.

In every (or in a specific number of) iteration(s), the DM guides the search by answering to some questions the analyst can make. These questions are normally related to trade-offs, aspiration levels or opinions. For example, if the analyst wishes to go from the point $\mathbf{x}$ to $\mathbf{x}'$, he may ask "The trade-off changing from $\mathbf{x}$ to $\mathbf{x}'$ is $t$. How desirable do you find this trade-off?" or then "Do you prefer $\mathbf{x}$ over $\mathbf{x}'$, the opposite, or are you indifferent

with them?" or other possibilities as can be seen in [6]. They usually assume that the analyst arrive somehow into a Pareto-optimal solution[3], and then start to "walk" along the front according to the answers obtained until the best solution is found.

Interactive techniques have the advantages already mentioned, and a better chance that the final solution will be actually the best possible. However, they have the peculiarity of bothering the DM way more than the other methods, and can be very time consuming. In some algorithms, the questions proposed may require differentiability or other properties of the objective functions [6]. Furthermore, some methods are criticized for making too much subjective questions [9], like "how important do you think this solution is with regard to that?". Since this is more related to psychology than to mathematics or engineering, the reader is referred to [6] and [9] for a more complete study about interactive methods.

#### 3.2.1.4 Non-preference methods

Here, there is no decision maker, or maybe he just shouted "Surprise me!" and closed the door of his office. In either way, there is no one to ask for preferences, so the idea is to find a solution that gives a compromise among the objectives, typically in the "middle" of the front.

Some examples of methods following this philosophy are the weighted metrics using the ideal point as reference point, and all weights equal to one (assuming the objectives are dimensionless). Other possibility is called *neutral compromise solution*, as can be seen in [12].

The non-preferences methods are normally used for preliminary tests, and often to furnish a starting point to the interactive methods, as mentioned before.

#### 3.2.1.5 Other classifications

The classification described before is by no means complete or exclusive, that is, some methods can belong to two or more types. For example, the weighting method can be used as *a priori*, but when the weights are varied it can be employed as an *a posteriori one*.

This classification is very popular, but there are other options. One different form of classifying methods for solving MOOs is [5]:

[3] This is not necessary, tough. But when that is assumed, this can be achieved by, e.g., using a prototype of a *a priori* utility function or a non-preference method, to be discussed next

- Explicit utility maximization;
- Implicit utility maximization;
- Partial generation of the efficient set.

The first class is similar to the *a priori* methods. It assumes the DM has enough knowledge about the problem so he can build an accurate utility function, which can be optimized to give the final solution. It is more common in multi-attribute problems.

The second type also require that the DM has an accurate utility function, but its formulation does not need to be explicitly known. It resembles the interactive methods, because, when the evaluation of the utility is required, an interaction between the DM and the analyst is performed by answering similar questions to the examples given. The difference is that the form of the utility function is sometimes pre-specified, so the answers are usually more direct.

Finally, the third type is considered one of the most promising methods [5]. It can be viewed as a mixture between interactive and *a posteriori* techniques. Here, only a small fraction of the efficient front is desired, so the DM is faced with a smaller number of alternatives, and they are already good candidates for being in his region of interest. What is required here is to define a method to express his preferences so as to focus in this region, which can be easier to express (see the next section).

This method can be considered better suited to problems with a higher number of objectives than the *a posteriori* methods, and can reduce the degree of interaction of the DM in the optimization process that is usual in interactive techniques. It is, actually, a preferred classification here in this work.

### 3.2.2 Expressing the DM's Preferences

When expressing preferences, the DM is in reality defining a region (or a direction to it) of preferable solutions in the Pareto front. The points belonging to this portion, called here *region of interest* (ROI), are considered better than the solutions outside it, even if they are non-dominated according to Pareto dominance.

As stated before, the DM manifest his desires by means of a utility function, which can be given beforehand or be constructed as the optimization progresses or some efficient alternatives are evaluated. However, according to [9], the two most common alternatives for this is by means of weighting coefficients or specifying goals or reference points.

The first technique corresponds to the weighting method already discussed in section 2.4.1.1. Usually the weights are used to express the relative importance of the objectives[4], so different regions of the efficient front can be approximated by this. Remember, however, that this method requires some assumptions like convexity of the front, and its control of the location of the final solution in the front is not very good.

An alternative is the use of reference points, which was already presented in section 2.4.1.3. In this case, the DM decides for some *aspiration levels* $z_i$ for the $i$-th objective, that is, for values that may represent good outcomes for him, combine them in a reference point $\mathbf{z}^r$, and find the solution that is the closest possible to it.

This last method has as advantages its more intuitive character and a better control of the position of the solution in the efficient front. It may be difficult for Maria to give an absolute value of how more important losing body fat would be to reducing her effort, but she could come up with aspiration levels like "I wanna get -8% of body fat percent after 3 months, requiring only 0.3 unit of effort". This would be a good reason for its popularity. Nevertheless, much of the theory of vector optimization was already described by the use of weights [10], while the methods that apply reference points or goals have no guarantee of generating efficient solutions [14], and the reference point cannot usually be freely chosen.

However, assuming that the use of aspiration levels is a promising approach, Wiezrbicki in [39] proposed a survey with the entire theory of basic multi-objective optimization - necessary and sufficient conditions of optimality and existence of efficient solutions - based completely on reference points. In simple words, instead of minimizing a norm like it was been done in the weighted metrics method, he proposed to optimize a different, more complicated function, but capable of generating Pareto-optimal solutions for *any* reference point. This function is called *achievement scalarizing function* (ASF), and I will briefly describe it in the following.

#### 3.2.2.1 Achievement Scalarizing Functions

The Achievement Scalarizing Functions (ASF) can be seen as a generalization to the use of norms in the weighted metrics. Basically, it consists of minimizing the distance to the reference point if it is unattainable, and maximizing it if it is feasible. Therefore, any reference point can be chosen for the optimization, and not just the ideal (or better) point. Also, the optimal solution can be changed just by using another reference. This allows the DM to express his preferences in a more controllable manner than by setting weights or changing the norms.

[4] Remember this only makes sense when the objectives are standardized.

The ASF of a point $\mathbf{x}$ is represented by $s(\mathbf{f}(\mathbf{x}), \mathbf{z}^r) : \mathbb{R}^m \mapsto \mathbb{R}$, wherein $\mathbf{z}^r$ is the reference point. The optimization problem becomes the following scalarization method:

$$\begin{aligned} \text{minimize} \quad & s(\mathbf{f}(\mathbf{x}), \mathbf{z}^r) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{3.2}$$

There are some properties this function may have [39].

*Definition* 8. An ASF is said to be, for any two feasible points $\mathbf{x}_1$ and $\mathbf{x}_2$ and a given reference point $\mathbf{z}^r$:

- **increasing**, if $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2), \forall i = 1, 2, \ldots, m$ implicate $s(\mathbf{f}(\mathbf{x}_1), \mathbf{z}^r) \leq s(\mathbf{f}(\mathbf{x}_2), \mathbf{z}^r)$;
- **strictly increasing**, if $\mathbf{f}(\mathbf{x}_1) < \mathbf{f}(\mathbf{x}_2)$ (weakly dominates) implicate $s(\mathbf{f}(\mathbf{x}_1), \mathbf{z}^r) < s(\mathbf{f}(\mathbf{x}_2), \mathbf{z}^r)$;
- **strongly increasing**, if $\mathbf{f}(\mathbf{x}_1) \prec \mathbf{f}(\mathbf{x}_2)$ implicate $s(\mathbf{f}(\mathbf{x}_1), \mathbf{z}^r) < s(\mathbf{f}(\mathbf{x}_2), \mathbf{z}^r)$;

These conditions are important, because, if the ASF is strictly increasing, then the solution to (3.2) is weakly efficient, but if it is unique, then it is Pareto-optimal. Also, if the ASF is strongly increasing, then the solution to (3.2) is Pareto-optimal [39]. A comparison between the minimization of a simple norm and the ASF is shown in Figure 3.2, where, for the given reference point, the solution of the norm is dominated, while the optimization of the ASF results in a Pareto-optimal point.

The ASF can assume various forms, like shown in [39]. The most common and used in this work is the (strongly increasing) augmented Chebyshev norm:

$$s(\mathbf{f}(\mathbf{x}), \mathbf{z}^r) = \max_{i \in \{1,2,\ldots,m\}} (f_i(\mathbf{x}) - z_i^r) + \rho \sum_{i=1}^{m} (f_i(\mathbf{x}) - z_i^r) \tag{3.3}$$

wherein $\rho > 0$ is a small parameter. This formulation assumes the functions are dimensionless already. If not, some weights are used in order to account for the difference in ranges.

When using an ASF of the form expressed in (3.3), the solutions are actually *properly efficient*. It can be proved that any proper Pareto optima can be found with this function by just varying the reference point. However, these solutions are, more rigorously, $\rho$-properly efficient, which is a definition slightly different from the one presented in section

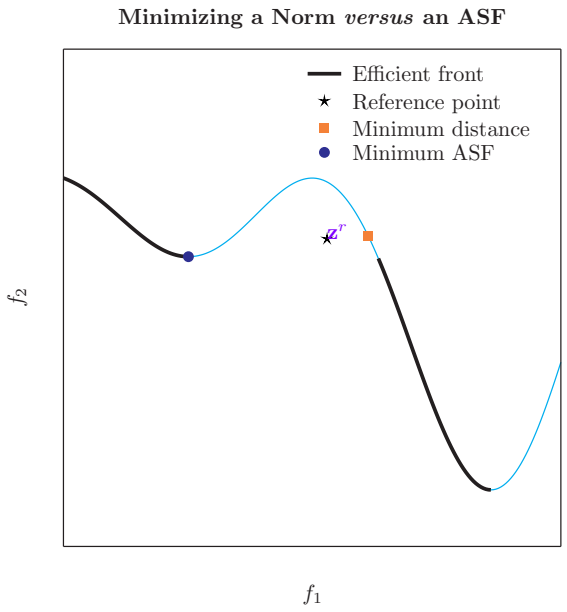


FIGURE 3.2: Getting different results in a MOO when minimizing the distance (Euclidean norm) to a reference point and minimizing an ASF. The solution of the first is dominated, while the solution for the second is efficient.

2.3.1.2. Here, the solutions have trade-offs bounded by [12] $\rho$ and $1/\rho$, and they can be seen as equivalent to the Geoffrion's definition by setting $M = 1 + 1/\rho$.

The ASF can be viewed as a utility function, such that the resultant point is the closest to the DM's desires, so it is the most preferred solution. This would make it a great *a priori* technique. However, this is not the intention. The idea is to view it as an abstraction of a utility function [39]. The reference point procedure is intrinsically an interactive method, so the purpose is to approximate locally the DM's preferences by asking questions he can understand well. The analyst can ask the following: "Dear Lord of the preferences, you have asked me to achieve the objective levels $\mathbf{z}^r = [z_1^r \; z_2^r \; \dots \; z_m^r]^T$. Under the limitations of the current model, the best I can do is $\mathbf{f}(\hat{\mathbf{x}}) = [f_1(\hat{\mathbf{x}}) \; f_2(\hat{\mathbf{x}}) \; \dots \; f_m(\hat{\mathbf{x}})]^T$. Do your highness accept this or prefer to change to new aspiration levels?".

The beauty of this method is that the DM can achieve different solutions just by changing the reference point[5]. It therefore avoids the costly and exhausting questions like "Do you prefer $a$ to $b$?" or "If you trade $a$ to $b$ you will have such a trade-off. How desirable do you find this trade?", and the preferences are expressed as aspiration levels, which are more natural for the DM.

Finally, in the same paper [39], it is proposed an interactive way for arriving at a final solution. In the beginning, the DM is presented with all possible information about the

[5]It is possible also to use weights like the weighted metrics, or even by changing the ASF, like the work presented in [40]. However, the modification of reference point is still preferred.

functions, like the ideal and Nadir points, if possible. Then he is asked to express his aspiration levels in a reference point $\mathbf{z}^r$.

The rest of the procedure goes like:

1. Find the solution to (3.2) using the provided reference point;
2. Create $m$ new perturbed reference points, where $m$ in the number of objectives, by adding to each coordinate of $\mathbf{z}^r$ the distance $d$ between this point and the solution obtained in step 1:

$$\mathbf{z}^{r,i}_{aux} = \mathbf{z}^r + d\mathbf{e}_i \tag{3.4}$$

   wherein $\mathbf{e}_i = [0 \quad 0 \quad \dots \quad \underbrace{1}_{i-\text{th}} \quad \dots \quad 0]^T$ is a vector of zeros in every coordinate, except on the $i$-th one, where its value is one.
3. Solve again (3.2) for each of these perturbed reference points.

The process is illustrated in Figure 3.3. Notice that this also helps to map the efficient front. The further the reference point is (that is, the DM had too optimistic desires), the more spaced the $m+1$ solutions will be, and vice-versa. After seeing these candidates, the DM may wish to change the reference point or to opt for one of these solutions as the best one.

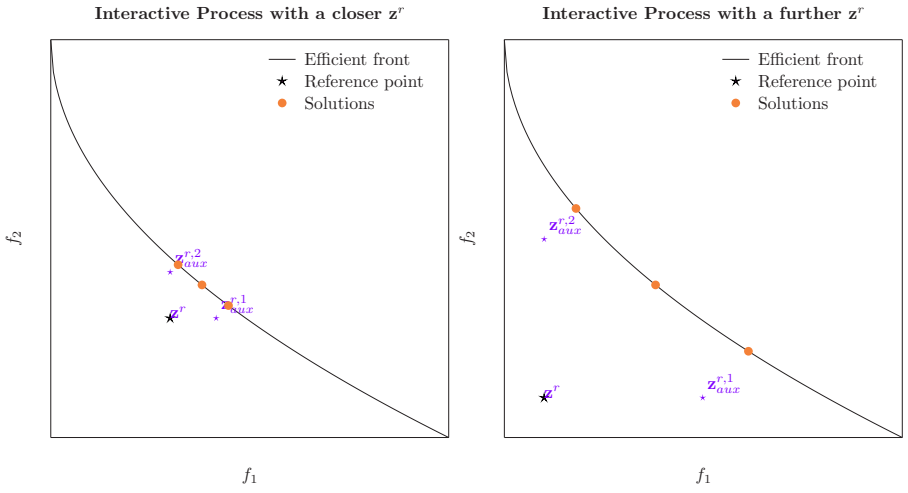


FIGURE 3.3: Interactive procedure for changing the reference point and obtaining new solutions. Reference points closer to the efficient front (left figure) result in less spaced auxiliary reference points, and thus, in closer solutions. An opposite reasoning applies when $\mathbf{z}^r$ is further (right figure).

### 3.2.3 Using Reference Points in Evolutionary Algorithms

Evolutionary Algorithms are typically used as *a posteriori* methods. But, since they are able to approximate regions of the Pareto front, why not use them to focus on a particular portion, more specifically, on the region of interest? This is in fact possible, as long as one elaborates a way of including preferences in an EA. Coello in [41] provides a survey with a lot of different methods for handling preferences in evolutionary algorithms, but the focus here is in methods that adopt the concept of reference points.

Basically, the idea is to complement the Pareto-dominance with some kind of distance (like norms or achievement scalarizing functions) to the reference point, so it can be used as another tie-breaker when comparing non-dominated points. The way how this information is used depends on the algorithm. I will describe two instances that will be used later in this work, and then briefly mention some other possibilities.

#### 3.2.3.1 Reference Point NSGA-II

This method is described in [42], and it is called Reference point NSGA-II, or shortly R-NSGA-II. Despite it was first proposed with the NSGA-II in mind, any Pareto-based algorithm can be used as a base.

As described in the previous chapter, in a typical iteration $t$, after the variation operators the population $\mathcal{R}(t)$ needs to be truncated to create the next population $\mathcal{P}(t+1)$. Here are the steps required for this truncation in this algorithm:

1. Begin with the usual non-dominated sorting, ranking $\mathcal{R}(t)$ in fronts. Starts to fill the new population with each of these layers, until, say, the $k$-th front has too many points;

2. Instead of the simple crowding distance, compute the distance of these individuals to the reference point, $\mathbf{z}^r$, using:

$$d(\mathbf{x}) = \sqrt{\sum_{j=1}^{m} w_j \left( \frac{f_j(\mathbf{x}) - z_j^r}{f_j^{max} - f_j^{min}} \right)^2} \tag{3.5}$$

   wherein $d(\mathbf{x})$ is the (Euclidean) distance of $\mathbf{x}$ to $\mathbf{z}^r$. Also, $w_j$ is the weight of the $j$-th objective, which could mean a relative importance, and $f_j^{max}$ and $f_j^{min}$ are, respectively, the maximum and minimum objective values within the population, in the current iteration. Normally all the weights are set to 1.

Notice that this is a regular distance to the reference point, not an achievement scalarizing function. However, even when $\mathbf{z}^r$ is feasible, the population does not get stuck at this point because the Pareto-dominance is checked first.

3. Just computing distances is not enough. In this method, in order to choose the remaining points from this $k$-th front, some kind of 'box-dominance', called $\epsilon$-clearing in the proposed article, is employed. Putting it simply, imagine the space being divided in a grid of size $\epsilon$ (which is furnished by the user). Then, if, for instance, $\mu_{missing}$ points are missing from this front, then pick the closest ones to the reference point. However, inside each grid there can be only one solution, usually the one with the smaller distance. Hence, if there are three solutions with distances, say, 1, 1.01 and 1.02, and the next closest has distance 2, if $\epsilon$ is such that the first three are inside the same "box", only the one with distance 1 is chosen, and the other two are discarded.

In this method, the population is guided by minimizing the distance to $\mathbf{z}^r$, and the size of the region of interest (ROI) is given indirectly by the value of $\epsilon$. Bigger values mean bigger boxes, which provide more spaced solutions and, then, a larger ROI, and vice-versa for smaller values. An example can be seen in Figure 3.4.

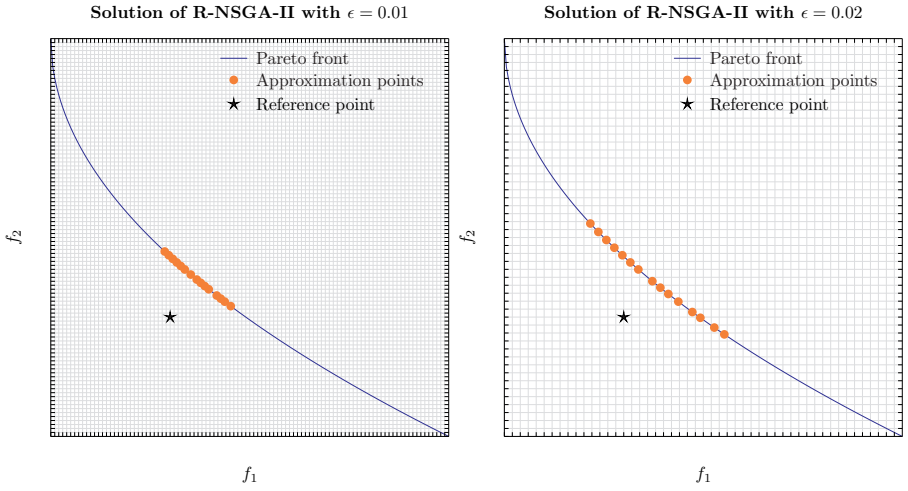


FIGURE 3.4: Typical outcomes of the R-NSGA-II method with two different values of $\epsilon$: 0.01 (left) and 0.02 (right). A smaller value allows closer solutions and, therefore, a narrower region of interest. An opposite reasoning can be made with the larger value.

#### 3.2.3.2 Preference-based Evolutionary Algorithm

The previous method is an adaptation of the original NSGA-II with Pareto-dominance when the DM's preferences are taken into account. The present algorithm, called Preference-based Evolutionary Algorithm (PBEA), was proposed in [43], and it is a

simple adaptation of the IBEA (described in section 2.4.3.2). Remember the IBEA uses a binary indicator $I(\cdot,\cdot)$ to assign fitness to each individual. The current method weights this fitness to give a better value to points that are closer to the reference point and, thus, satisfy better the DM's desires.

Consider iteration $t$, where the same population $\mathcal{R}(t)$ needs to be reduced to the size $\mu$. For each point $\mathbf{x}_i$, first compute its ASF value, $s(\mathbf{f}(\mathbf{x}_i), \mathbf{z}^r)$ using equation (3.3). Now, guarantee that it will always be a positive value by summing to it the minimum value of the current population:

$$\bar{s}(\mathbf{f}(\mathbf{x}_i), \mathbf{z}^r) = s(\mathbf{f}(\mathbf{x}_i), \mathbf{z}^r) - \min_{\mathbf{y}\in\mathcal{R}(t)} \{s(\mathbf{f}(\mathbf{y}), \mathbf{z}^r)\} + \delta \tag{3.6}$$

where $\delta > 0$ is a small constant used to prevent a zero value.

In the original IBEA, in order to compute the fitness $\varphi(\mathbf{x}_i)$ of $\mathbf{x}_i$, the indicator $I(\mathbf{x}_j, \mathbf{x}_i)$ is first calculated for all $\mathbf{x}_j \in \mathcal{R}(t)\backslash\{\mathbf{x}_i\}$. Here, this same process is adopted, but the indicator is divided by the modified ASF given in equation (3.6):

$$\bar{I}(\mathbf{x}_j, \mathbf{x}_i) = I(\mathbf{x}_j, \mathbf{x}_i)/\bar{s}(\mathbf{f}(\mathbf{x}_i), \mathbf{z}^r) \tag{3.7}$$

Figure 3.5 shows typical outcomes of the regular IBEA with no preferences, and the PBEA. Notice how the population in the latter case (right figure) is more concentrated next to the reference point, while in the former situation (left figure) has a more well distributed population in the whole efficient front.

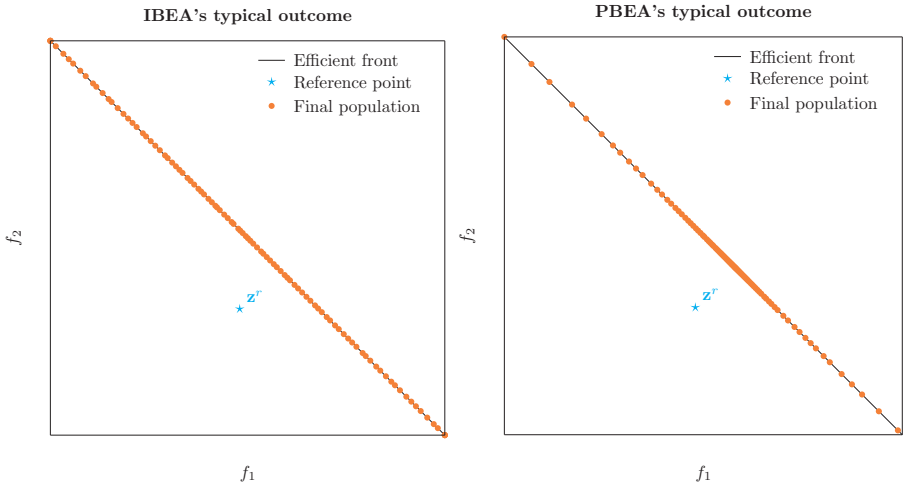


FIGURE 3.5: Typical outcomes of IBEA (no use of preferences) and PBEA (using preferences). The first tries to approximate the whole efficient front with a good diversity in the final population, while the second prefers the individuals that are closer to $\mathbf{z}^r$.

With this discussion, the PBEA steps for performing selection can be summarized as:

1. Given the population $\mathcal{R}(t)$, compute the ASF value for each individual $\mathbf{x}_i \in \mathcal{R}(t)$;
2. Calculate the fitness $\varphi(\mathbf{x}_i)$ using equations (2.16) or (2.17) given in the last chapter. Remember to use the modified indicators as shown in equation (3.7);
3. Find the point with the worst fitness and remove it from the population;
4. Repeat the process until the number of elements in $\mathcal{R}(t)$ is $\mu$. In that case, this will be the new population $\mathcal{P}(t+1)$.

Finally, notice that in this method it is not possible to control the size of the region of interest.

#### 3.2.3.3 Related methods

The two previous algorithms were described in more detail because they will be used in this work, but there are other possibilities of encoding the DM's preferences into a reference point and using it to guide the search. Some examples are:

- Wickramasinghe and Li proposed in [44] the minimization of distances instead of the non-dominated sorting. In the selection procedure, first compute the following (Chebychev) distance of each individual to the reference point:

$$d(\mathbf{x}) = \max_{i \in \{1,2,\ldots,m\}} \{w_i(f_i(\mathbf{x}) - z_i^r)\} \tag{3.8}$$

  wherein $w_i$ is a scaling weight of objective number $i$. Then, find the closest point to $\mathbf{z}^r$, with distance $d_{min}$. For the selection, pick every individual that has a distance smaller than $d_{min} + \delta$, wherein $\delta$ is a user set parameter. If there are too many points, remove the furthest ones, and if there are too few, pick some random solutions from the remaining population. See Figure 3.6 for a graphical example.

  A first comment about this method is that there is also a light-beam search variation (check [44]). The second one is that, according to some of my preliminary tests, this simple selection procedure is not enough to guide the population to the efficient front. Therefore, I propose, for future experiments, to add non-dominated sorting when the number of points in this region of interest is too large.

- In [45] a weighted crowding distance is introduced. The selection undergoes the usual non-dominated sorting, and the crowding distance is used as a secondary selection criterion. However, this crowding value is weighted by a given probability

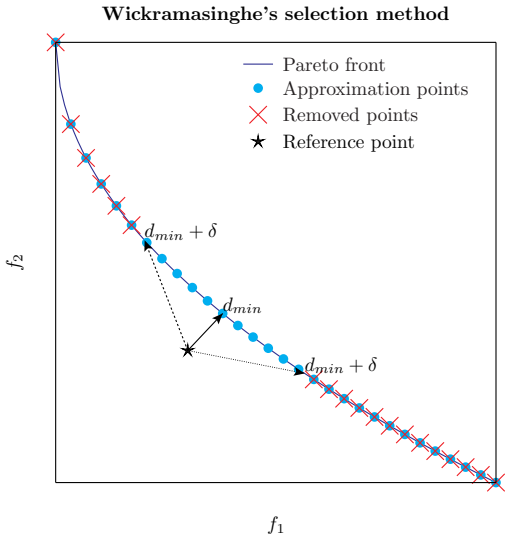


FIGURE 3.6: Wickramasinghe and Li's selection method. All of the solutions within $d_{min} + \delta$ are considered to be in the region of interest.

function. They propose three options, and one of them is a ridge-like distribution that emphasizes the points in the "middle" of the front and closer to $\mathbf{z}^r$.

There are two observations that the authors did not make clear in their paper: the first is that, if the crowding distance of the extreme solutions is considered infinite, these points will be kept; another one is that this distribution is explicitly written for the two-objective case. It *may* be extended to more objectives, but I am not really sure how or if this is enough to provide good solutions in this case.

- Auger et al. in [46] used a similar method as before, but instead of weighting the crowding distance, the hyper-volume is weighted instead. This indicator is well-known to grow in computational cost when the number of objectives increases, so a different version, using Monte-Carlo sampling methods, is adopted in order to tackle problems with many objectives.

  The weighting distribution here is a multi-dimensional Gaussian with center on the reference point, and covariance matrix formed by some parameters defined by the DM, which control the size of the ROI. See the original paper for more details.

## 3.3 Summary

The complete process of solving a multi-objective optimization (MOO) problem involves computing efficient solutions, and choosing one or a few of them that pleases a human

decision maker (DM) the most. It is sometimes assumed that each alternative of a problem has an outcome with a given *utility value*, which represents the degree of satisfaction of the DM. The best solution will be the one with highest utility value.

In typical MOO problems, it is impractical to enumerate the whole feasible set, so the DM needs to be supported by an *analyst* in order to narrow this myriad of points to some representative solutions. Depending on how the DM expresses his preferences, the MOO methods are usually classified as *a priori*, *a posteriori*, interactive and non-preference based (see section 3.2.1), each one with their strengths and flaws.

Even though the DM may have a utility function to guide his decisions, this is not the only way of expressing preferences in the optimization process. Two more practical methods are using weights and reference points. This last one has the advantage of being more natural and requiring more easy-to-provide answers from the DM. Also, thanks to the contribution of Wierzbicki in [39], the whole multi-objective theory was formulated in terms of reference points, allowing them to become more popular methods.

Evolutionary Algorithms are used typically as *a posteriori* methods, so they usually have the goal of approximating the whole efficient front. This philosophy can work well with a few objectives, but for four or more, it is more likely that the DM will end up with a problem of selecting between hundreds or maybe thousands of efficient solutions, which is no more easier than the initial problem. Because of that, it is preferable to concentrate their search in a smaller region of the Pareto front, following the class of methods that "generate a partial representation of the front" [5].

There are, however, more setbacks evolutionary algorithms have when dealing with problems with a higher number of objectives. The next chapter, then, will be dedicated to the field of optimization when the number of criteria is too large.

# Chapter 4

# Many-objective Optimization

I will start this chapter by showing the formulation of problems with many objectives:

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \end{aligned}$$

If you look closely, you will see that this is exactly the same definition of (2.3) of chapter 2. This is to be expected, since, when formulating a multi-objective problem, there is no condition on the number of criteria $m$, except for it being greater than one.

So, why create a new sub-field of optimization? And even so, how many is many? Despite not having definite answers, it is useful to briefly discuss these questions before moving on. Later in this chapter, I will point some setbacks that some algorithms (especially EAs) have in many objectives, and then mention some methods that have been proposed to overcome these issues.

## 4.1 Why Many-objective Optimization?

First of all, keep in mind that the term *many-objective optimization*, abbreviated MaOO, is normally cited only in the context of Evolutionary Optimization. As mentioned in section 2.4.3.6, the evolutionary algorithms considered state of the art (like NSGA-II, SPEA2 and other Pareto-based methods) had their moments of glory mostly because the problems they solved had two or three objectives. Because of that, the *a posteriori* philosophy worked like a charm, since the Decision Maker could simply visualize the options before taking a final decision.

However, some studies (summarized in [47]) verified that, when the number of objectives increased, the same state of the art did not return good results, i.e., the final population had individuals too far from the efficient front. This was seen, for instance, in test bench problems where the Pareto-front is known or can be easily generated. Because of this, some researches started to acknowledge problems with many objectives and developed some special techniques to better handle them.

But what is the threshold number of objectives that separates "multi" from "many"? The answer is "when the state of the art is not able to provide good results". This is possibly not a satisfactory answer, since it may depend on the algorithm and on the problem. The situation is similar to Newton's laws from physics. They are actually approximations for relatively big bodies moving at relatively low speeds and, when these premises are not verified, the quantum mechanics and relativity theory must be employed instead. However, there is no formal threshold of size and speed such that the results stop being accurate enough, since it may depend on the experiment (and on the experimenters). Even with this discussion, I will adopt here the convention that $m \geq 4$ [47] is enough to characterize a problem as belonging to the *many-objective* class.

The other question is about the need for creating this new field, since, as one may imagine, an algorithm that solves 10 objectives may be able to solve a two-objectives easily[1], so why worry about the state of the art if it is more limited? The possible answer to that is: because they are the state of the art! They are well-known, possibly easier to understand, there are available software and code about them, and the scientific community has an inertia to start using new methods and stop employing the old ones. And, to the extent of my knowledge, there is no state-of-the-art for many objectives yet, with the field being still well open to new studies [47].

## 4.2 Many-objective issues

After some studies reported the not so good results of the usual Pareto-based algorithms in problems with more than 3 objectives, some researches tried to figure the reason for this. Basically, the difficulties faced by EAs in MaOO are [47, 48]:

- **Deterioration of the selective pressure:** as the number of objectives grows, the more solutions start to get non-dominated among themselves. Imagine an employer interviewing one thousand people for a job. If the director wants the person with the best motivation, probably only one will pass the test. But the director requires the most motivation and competence, more people will pass, since

[1] Of course, provided the space of variables is not changed drastically from one problem to the other.

some may not be so competent but be more motivated instead, and vice-versa. Now, if the director looks for most motivation, competence and appearance, then even more people will be selected... As he increases the number of criteria, there will be more and more people that may fail a few ones, but that excel in others. That is, the number of non-dominated people increases.

In the optimization context, the problems with this are:

  - The region **II** of Figure 2.15 becomes smaller, and then, it is harder to generate an offspring that produces an improvement in all objectives;
  - A higher number of individuals in population $\mathcal{R}(t)$ becomes non-dominated, even when they are very far from the efficient front. Because of this, the selective pressure is deteriorated, and not even the crowding distance is enough to help it [19].

- **Increase of the efficient front:** As mentioned in section 2.3.3, the efficient front, for a $m$-objectives problem, is normally a $(m-1)$-dimensional hyper-surface. This means that, normally, the size of this front increases with $m$ and, therefore, the required number of points to give a good representation of it (in the *a posteriori* sense) grows exponentially. This not only increases computational cost, but also makes it harder for decision makers to select a final solution;

- **Difficulty of visualization:** some DMs choose the final solution by visualizing the alternatives. When $m > 3$, they lose this luxury. This is problematic not only with *a posteriori* methods, but also in some interactive ones, where the DM usually compares two solutions by actually seeing them [6];

  Deb [48] also mention these other setbacks:

- The evaluation of diversity indicators becomes more expensive, as it is trickier to identify neighbors in high dimensional spaces. Furthermore, approximations to accelerate this may result in a not so diverse final set;

- Some quality indicators also become more costly with higher values of $m$. The hyper-volume is such an indicator, but other examples are hardly or never mentioned in the literature.

The first issue is normally verified by experiments, as can be seen in [47] and [49]. There are some other studies that follow a more theoretical approach. Schutze et al. studied in [50] the influence of $m$ on the "hardness" of problems. This was measured as the capability of an Evolution Strategies Algorithm to provide good solutions. They concluded that, in the unimodal class of problems tested, the inclusion of an objective

does not make a problem harder; however, this does not seem to be extensible for more complex models.

Santos and Takahashi in [51] provided a similar but more rigorous analysis. They used $m$ quadratic functions and computed the probability of generating a successful individual in this problem for a given distance to the Pareto front. They showed that this probability is high when we are far from the efficient front, but it gets smaller by a power law function of the distance with exponent $m-1$, that is, it gets smaller quickly when $m$ is large. Finally, Teytaud in [52] presents a very rigorous study showing that, in summary, all comparison-based algorithms (this includes Pareto-based ones) have a performance comparable to a simple random walk when the number of objectives grows.

After recognizing these difficulties, a lot of different methods were proposed in order to cope with the non-scalability of the current techniques. In the next section I will present some of them.

## 4.3 Methods for Solving a Many-objective Problem

Given the issues presented before, the proposed new methods tried to handle them in different forms. I will briefly present some ideas in this section. The interested reader can find a survey about MaOO in [47].

### 4.3.1 Modification of the Pareto dominance

Since one of the problems with Pareto dominance is its deterioration of selective pressure when $m$ grows, some studies decided to modify it or simply replace it. Sato et al. decided in [53] to control the "aperture" of the dominance cone. Figure 4.1 shows an example. By "opening" the cone, the dominance influence of each point is increased, and so is the selective pressure. The aperture is controlled by a parameter given by the user, whose best value depends on the problem.

Some different studies replace the Pareto dominance with the concept of *fuzzy dominance*, as shown in [54] and [55]. They argue that the number of improved objectives is not taken into account by the usual dominance, nor the (standardized) size of the improvements. For example, imagine the point $\mathbf{x}_1$ with all 50 objectives equal to 1, and point $\mathbf{x}_2$ with the first 49 objectives equal to 100, but the last one equals to 0.9. They are non-dominated among themselves, but, according to fuzzy dominance, since $\mathbf{x}_1$ is better than $\mathbf{x}_2$ in 49 objectives, the first should be preferred. The details can be seen in the original papers.

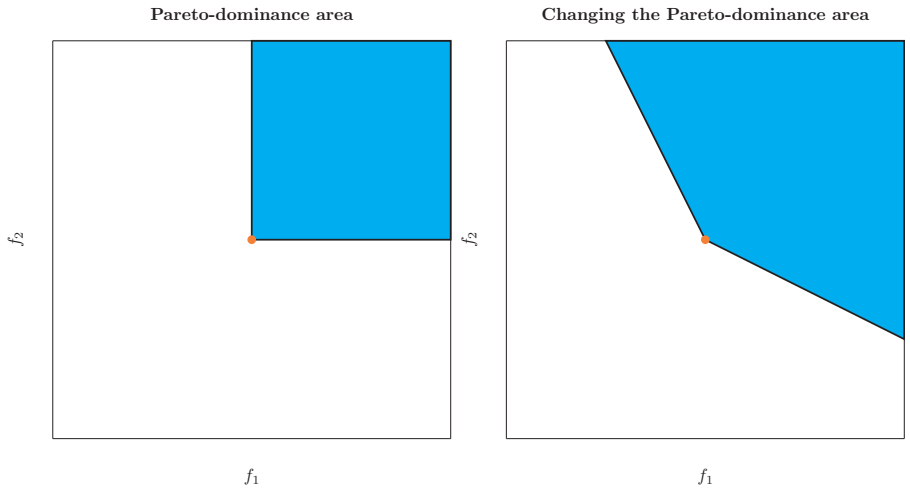


FIGURE 4.1: Handling with many-objective by changing the dominance region. By "opening" the cone, each point increases its dominance region and, then, the selective pressure is increased.

Another approach is to use indicators instead of dominance. The basic idea was presented in section 2.4.3.2. The most common indicators are the $\epsilon$ additive and multiplicative, together with the binary hyper-volume (see [24]), but more options were proposed, like can be seen in [56], [57], [58] and [59]. The advantage of indicators is that most points have different fitnesses, even if they are non-dominated. Since it is very unlikely that all individuals have the same fitness, even for higher dimensions, they do not lose the selective pressure and showed better results in arriving at the efficient front when compared to the simple Pareto-based methods.

### 4.3.2 Reducing the number of objectives

"My state of the art algorithm works so well with a few objectives. I wish my problems had less criteria...". This thought can actually be implemented in practice.

One common thing in multi-objective problems is that the objectives are *conflicting*, that is, when you improve one some of the others may deteriorate. However, this does not happen in the whole feasible space. For the functions of Figure 4.2, when $x \leq 0$ or $x \geq 1$, both functions decrease or increase simultaneously, and the conflict happens only in $0 < x < 1$. The first portions are called regions of *harmony*. The important thing about them is that it is possible to minimize both objectives by focusing only in one of them. When more criteria are considered, there are greater chances to find functions that are in harmony, and then, only one of these needs to be considered in these regions. This can simplify some high-dimensional problems.

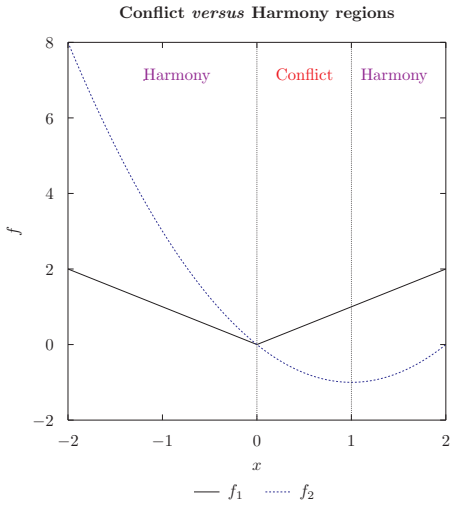


FIGURE 4.2: Given two functions, the parts of the feasible region where both increase or decrease simultaneously are named regions of *harmony*, and the portions one decreases while the other increases are called regions of *conflict*.

The difficulty about this method is how to find which functions are in harmony, and in which parts of the feasible region. Until the present moment, this is still a challenge. For the interested reader, the reference [60] presents a survey about these methods.

### 4.3.3 Supplementing the Pareto-dominance

Instead of completely replacing the Pareto-dominance, some methods decided to supplement it in order to distinguish non-dominated points. One attempt is given in [61], which proposes some metrics to rank solutions, even when they are non-dominated among themselves.

Another idea is provided by Köppen and Yoshida in [62]. They adopt the non-dominated sorting with the regular Pareto-dominance. However, instead of the crowding distance as a second selection criterion, they employ indicators in the same way as described in 2.4.3.2. Some examples used are the $\epsilon$ additive and fuzzy indicators.

A similar concept was adopted in a revamped version of the NSGA-II, called NSGA-III [48], which was created with many-objective in mind. They require first the computation of the ideal point. The selection process starts with the usual non-dominated sorting with Pareto-dominance. Then, a crowding distance replacement is adopted. The idea is to generate some auxiliary points in a hyper-plane that contains the ideal point, and draw a line passing in each point. This is basically the same process as described for the NBI method in section 2.4.1.4. The individuals in the population are associated to these lines, such that the closest ones are preferred, and preferably only one solution should be associated to each line. This guarantees a better selective pressure and diversity,

assuming the auxiliary points are uniformly spaced. If you are interested, check the detailed procedure in the original paper [48].

Since human imagination is boundless (just like the stupidity[2]), this work is not able to mention all of the possible methods proposed to cope with many objectives. One last method described here is to supplement the Pareto dominance with the preferences of the DM. In this way even non-dominated points can be ranked, and the attention can be focused in only a portion of the efficient front. A survey of how to include preferences in EAs can be found in [41], and I have already presented some techniques that use reference points in section 3.2.3. Personally, I think this kind of methods are the best-suited for solving a problem with many objectives whenever the preferences can be modeled or consulted.

## 4.4 Summary

When a multi-objective problem has more than 3 objectives, it falls into the classification of *many-objective optimization* (MaOO). In this class, the usual state of the art (specially Evolutionary Algorithms) start to have its performance deteriorated.

The setbacks presented before for MaOO, especially the first three, are mainly due to the (in)famous goal of *a posteriori* methods that aim at giving an approximation of the whole efficient front. The intention of sparing the DM by presenting efficient solutions (or a good approximation of them) only may be good, but it does not work well in large fronts.

Let us be frank: you do not have to approximate the *whole* front. Just because you can it does not mean that you must (or even that you should)[3]. A lot of researches, like Rosenthal in [5], already believed that the best way of helping the DM in selecting a final solution is by a *partial* generation of the Pareto front.

Most of the techniques presented to handle many objectives try to solve just the problem of not getting close to the Pareto front. In that way, they are still worshiping the philosophy of "Let us just try to get efficient solutions and let the DM worry about choosing one of them later. At least our job is done". The other issues like high computational cost for using a large number of individuals for approximating undesired solutions are still there.

[2] Adapted from a quote usually attributed to Albert Einstein: "The difference between stupidity and genius is that genius has its limits."

[3] Especially with many objectives, where it has been shown that you usually can not!

Therefore, I think that if the decision maker is able to have a certain degree of participation in the optimization process, the problems with many objectives can be solved more efficiently: preferences can be used to supplement the Pareto dominance, alleviating the convergence issue; there is less time wasted with solutions that the DM simply does not care; and there is no need to use enormous population sizes, since only a smaller fraction of the optimal front is desired[4].

I have presented in the last chapter some methods that use preferences. In the next chapter I will introduce the method proposed in this work that uses DM's preferences to handle many-objective problems.

[4]Of course, do not forget the problem of selecting the minimum population size, which may also depend on the number of variables. So, even if the DM desires only 5 final solutions, a problem with 40 variables may not be well solved by a population of 5 individuals only.

# Chapter 5

# The Proposed Method

The idea of using reference points to express the decision maker's preferences and use them to guide the population in an evolutionary algorithm is not new. I have presented some of these methods in section 3.2.3. So, why bother making another method with all of those existing ones? To be honest, I had no intention on creating another technique, because some of the existing approaches, like PBEA [43] and Wickramasinghe's method [44], already give pretty good results when this last one is augmented with my suggestion of including non-dominated sorting. However, the way they control the DM's region of interest (ROI) is too indirect and usually results in an unpredictable ROI size.

Because of that, I decided to propose a method that does not require any parameter to set the size of the ROI: it is created according to the reference point, $\mathbf{z}^r$. I think this is a good approach because, as it will be seen, it extends the Wierzbicki's interactive idea (described in section 3.2.2.1), and the DM can set new reference points if desired. With that in mind, let us go to the presentation.

Typically, in order to create a method compliant with the DM's preferences, one needs to:

1. Of course, define how the DM expresses his preferences;
2. Decide how to create the region of interest, and which points belong to it;
3. Determine how to organize these points within the ROI (using diversity, sensitivity etc.).

In this chapter, I will delve into each one of these steps.

## 5.1 Expressing DM's preferences

It may be clear already that, in this work, it is assumed that the decision maker is able to provide aspiration levels $z_i^r$ for the $i$-th objective, which, collectively, form a reference point $\mathbf{z}^r = [z_1^r \quad z_2^r \quad \dots \quad z_m^r]$. This point has no restriction of being unattainable, or dominate every feasible point. Also, in principle, it does not have to be close to the optimal front, but, as will be seen later, this distance controls the size of the region of interest. A possible future work will, hopefully, take care of this issue.

## 5.2 Forming the Region of Interest

In the methods reviewed, the size of the region of interest is decided by different ways (like using a box-domination in Deb's method [42], or adjusting the variance in a normal distribution in Auger's technique [46] etc.), and they require user-defined parameters. Among all of them, I think the most straightforward and intuitive is Auger's method of choosing a variance. Anyway, what I propose here is a self-adaptive one.

The motivation for this is the belief that, in the beginning of the optimization process, it is o.k. to let the algorithm explore the whole space, so a big region of interest is nice. When the solutions start to get close to the reference point, it is time to narrow the region in order to improve local exploration around the desired portion, and also to increase the selective pressure in case the Pareto dominance becomes insufficient.

Even if you are not familiar with self-adaptive methods, there is one thing you may probably be asking yourself: how exactly can somebody say that "the solutions are starting to get close to the reference point"? Also, what can be considered a "close enough" solution so the region starts to get narrower? *And*, "get narrower" by how much, how often?

One solution for this problem is to introduce a lot of user-defined parameters, and, in the tests, define the values that generate the best results and write in the conclusion "The proposed method is able to produce competitive results in comparison to the state of the art". While very common, this approach can be criticized as arbitrary or overly ad hoc.

Instead of this, a simple method is adopted. The size of the region will depend on the distance, or, more precisely, on the achievement scalarizing function (ASF) value of the closest solution to the reference point. In this way, in the beginning, the points are possibly far from $\mathbf{z}^r$ so the region is broader, and, as the population improves and gets closer to it, the region becomes smaller.

The approach here is inspired by an idea of Wierzbicki in [39], very similar to the described in section 3.2.2.1. Even though the method there had a different goal, it is adapted here to define the ROI. In order to understand it, check Figure 5.1. The steps are as follows:

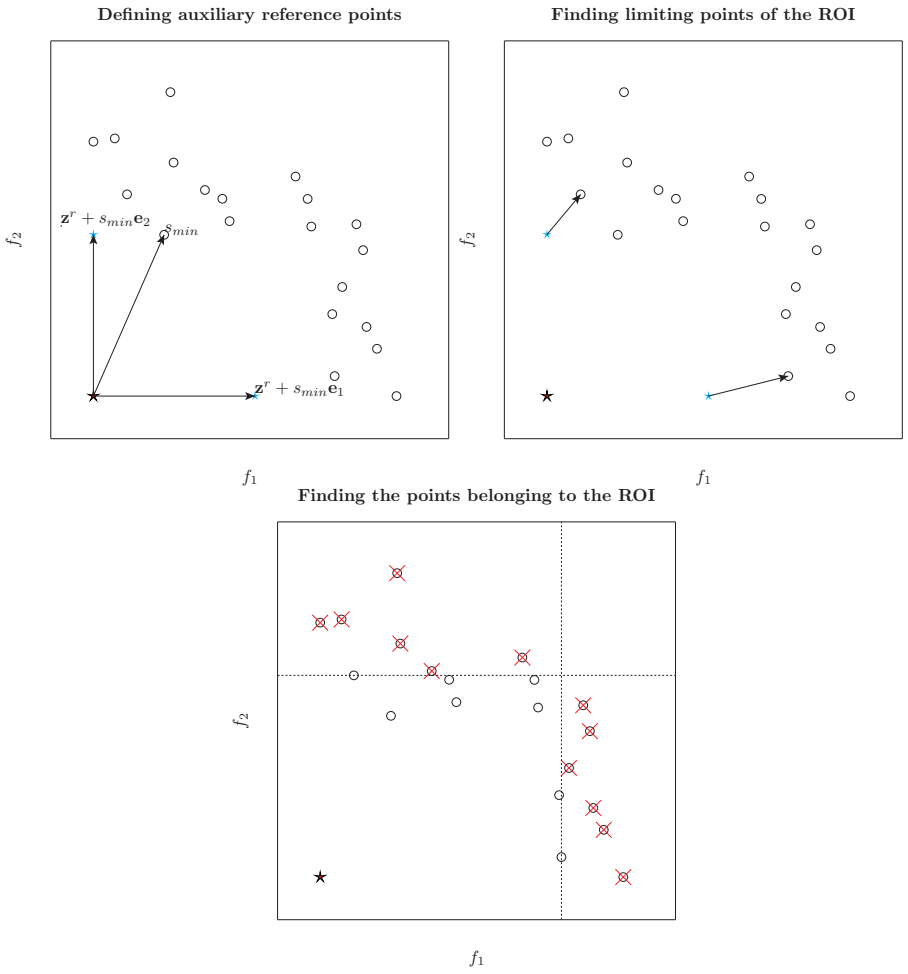


FIGURE 5.1: Defining the points in the Region of Interest in the proposed method. In the top left figure, the point with minimum ASF, given by $s_{min}$, is found, and the auxiliary $m$ (equal to 2 here) reference points are created. In the top right, the closest points to these auxiliary references are also found. They will be used to define the region of interest, as shown in the bottom figure.

1. Given the population in the objectives space, compute the achievement scalarizing function of each point with respect to $\mathbf{z}^r$. Any suitable ASF, like the one in (3.3) can be used;

2. Find the point with smallest ASF value, and call it $s_{min}$;

3. Compute auxiliary reference points by adding "perturbations" to the original one. If $m$ is the number of objectives, the $i$-th auxiliary point, with $i = 1, 2, \ldots, m$, will be given by

$$\mathbf{z}^r_{aux,i} = \mathbf{z}^r + s_{min}\mathbf{e}_i \tag{5.1}$$

wherein $\mathbf{e}_i = [0 \quad 0 \quad \dots \quad \underbrace{1}_{i-\text{th}} \quad \dots \quad 0]$ is a vector of zeros in every coordinate, except on the $i$-th one, where its value is one. In other words, the auxiliary reference points are perturbed in a direction parallel to the coordinate axes, with a magnitude equal to $s_{min}$ (see the top left graph in Figure 5.1). In Wierzbicki's method, the Euclidean distance is used. I preferred to adopt the ASF directly because, if the reference point is dominated, $s_{min}$ will be negative, and them the auxiliary points will be in the direction of this current best point.

4. Compute the ASF of each individual to each of the auxiliary reference points. Associate, for each $\mathbf{z}^r_{aux,i}$, the individual with the smallest corresponding ASF (top right in Figure 5.1);

5. Assume the point $\mathbf{x}_i$ is associated to the $i$-th auxiliary reference point. The individuals with $i$-th objective value smaller than $f_i(\mathbf{x}_i)$ will be included in the ROI (bottom in Figure 5.1). That is, the points $\mathbf{y}$ that belong to the ROI satisfy

$$\mathbf{y} \in \text{ROI} \implies f_i(\mathbf{y}) \leq f_i(\mathbf{x}_{closest,i}), \ \forall i = 1, 2, \dots, m$$

wherein $\mathbf{x}_{closest,i}$ is the point with best ASF with regard to the $i$-th auxiliary reference point $\mathbf{z}^r_{aux,i}$.

For example, in the two-objective case, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are associated with the perturbed reference points $\mathbf{z}^r_{aux,1}$ and $\mathbf{z}^r_{aux,2}$ respectively, the points with $f_1$ value smaller than or equal to $f_1(\mathbf{x}_1)$ *and* with $f_2$ value smaller than or equal to $f_2(\mathbf{x}_2)$ are in the region of interest.

It is probably clear that, when the population cannot improve anymore (hopefully when it arrives at the optimal front), the size of the final region of interest will depend on how far the reference point is, like Figure 5.2 shows. When choosing a further point, the region gets larger, and vice-versa when $\mathbf{z}^r$ is brought closer to the population.

### 5.2.1 Discussion about this method

Defining the ROI in this way brings an advantage that is the lack of any additional parameters to be set by the user. The size of the ROI is controlled simply by the selection of $\mathbf{z}^r$.

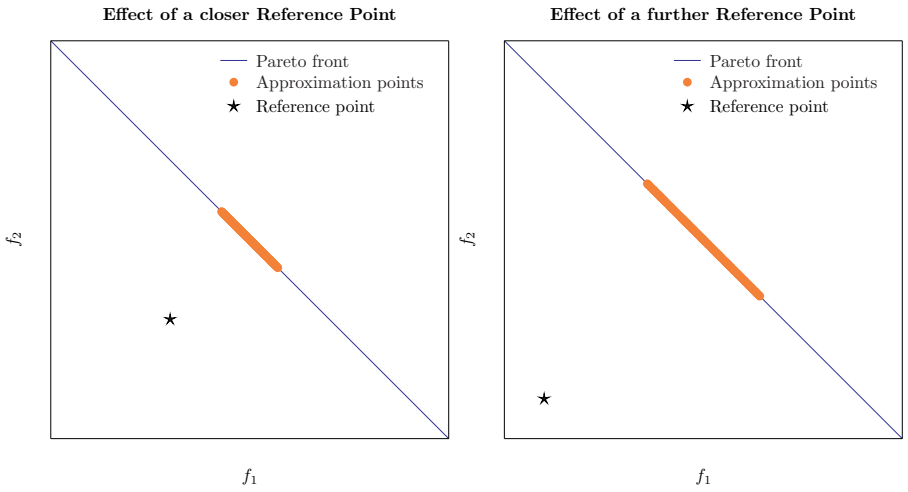


FIGURE 5.2: Effect of changing the distance of the reference point to the Pareto front.

The reference point method is intrinsically an *interactive method*. The DM does not need to give his opinion at every single iteration, but he can check that, if the points start to get too close to the reference point (or even surpass it), there is the possibility to change it so that a broader region is considered. Similarly, if he sees that the ASF value is not getting smaller, he may choose another (less ambitious) goal as well. The difference from Wierzbicki's method from section 3.2.2.1 is that the proposed technique extends the idea by not just presenting the current best and extreme points of the ROI, but also some internal solutions. This is in the spirit of the "partial generation of the efficient front" idea.

This method for selecting the ROI has a flaw: there seems to be an association between the size of the ROI and the selective pressure of the population, even though there are no formal studies about it. Intuitively, the smaller the ROI, the less points will belong to it, so there will be less ties in the selection. Therefore, the performance of this method is directly related to the choice of the reference point: if it is too far from the efficient front, the ROI will be large and, hence, the selective pressure may not be enough to guide the population to the optimal solutions. This is annoying, because the DM is the one to be satisfied, so it is not fair to tell him "Your reference point is inadequate. *You* are ruining my method. It is your fault, not mine!".

Unfortunately, I do not present a solution for that here. For this work, it is assumed that the DM is satisfied with his first choice of $\mathbf{z}^r$, *and* it is close enough to the Pareto front of the test functions[1]. Hopefully later studies will remove both of these (somewhat restrictive) assumptions.

[1]For the experiments, this was achieved by generating a random efficient solution, and adding some small perturbation in the objective space.

One more comment about this method is: if the DM already has an idea of how to define the size of his region of interest like "a 10% tolerance above and below the solution with best achievement value", this can be easily incorporated in the method by making the ROI be no smaller than that, for instance.

## 5.3 Organizing the points within the ROI

The previous section showed how to get points that may be in the region of interest of the Decision Maker. Inside it, it is assumed that no solution is preferable to another. This is the same idea as the points in the Pareto front, but here with a smaller portion.

As soon as the algorithm proceeds, the population and its offspring start to gather inside this region (besides some mutations that may create points in distant places). So, at first, the job is done, since any point inside the ROI is good enough for the DM. However, this opens space to include another criterion to improve even more the population, that is, there is a way to "organize" the points inside the ROI.

The most preferred method is diversity: it is good to have points uniformly spaced in order to provide alternatives. This criterion is also adopted here, but you can use your imagination to include any other requirement.

The usual technique of diversity is the crowding distance. It is a good option, but it is not well suitable for more than two objectives [48] and it is also only applicable in the objective space. I present here a different method that can be used in both spaces, and is at least conceptually extensible to any number of dimensions.

### 5.3.1 The Maximum Diversity Problem

Suppose you have a set $\mathcal{N}$ with $n$ objects, which could be points, people, books... or virtually anything. The *maximum diversity problem* (MDP) (see [63] for alternative definitions) consists of selecting a subset $\mathcal{M} \subseteq \mathcal{N}$ with $m \leq n$ objects such that a diversity function $\tau(\mathcal{M})$ is maximized (see Figure 5.3).

The resultant subset will depend on the diversity function used. There are lots of possibilities [64], and the most common are the sum or the minimum of the inner distances $d_{i,j}$. Here, $d_{i,j}$ is any distance measure between the $i$-th and $j$-th objects, like the Euclidean one[2]. In this work, the second function (the minimum) is used. Then, the

[2] I am using the fact that the Euclidean distance of points in an Euclidean space is something clear for everyone, which is the case in this work. Now, if you are wondering how to use this same Euclidean distance in people or other things, check further references on this topic, like [63].

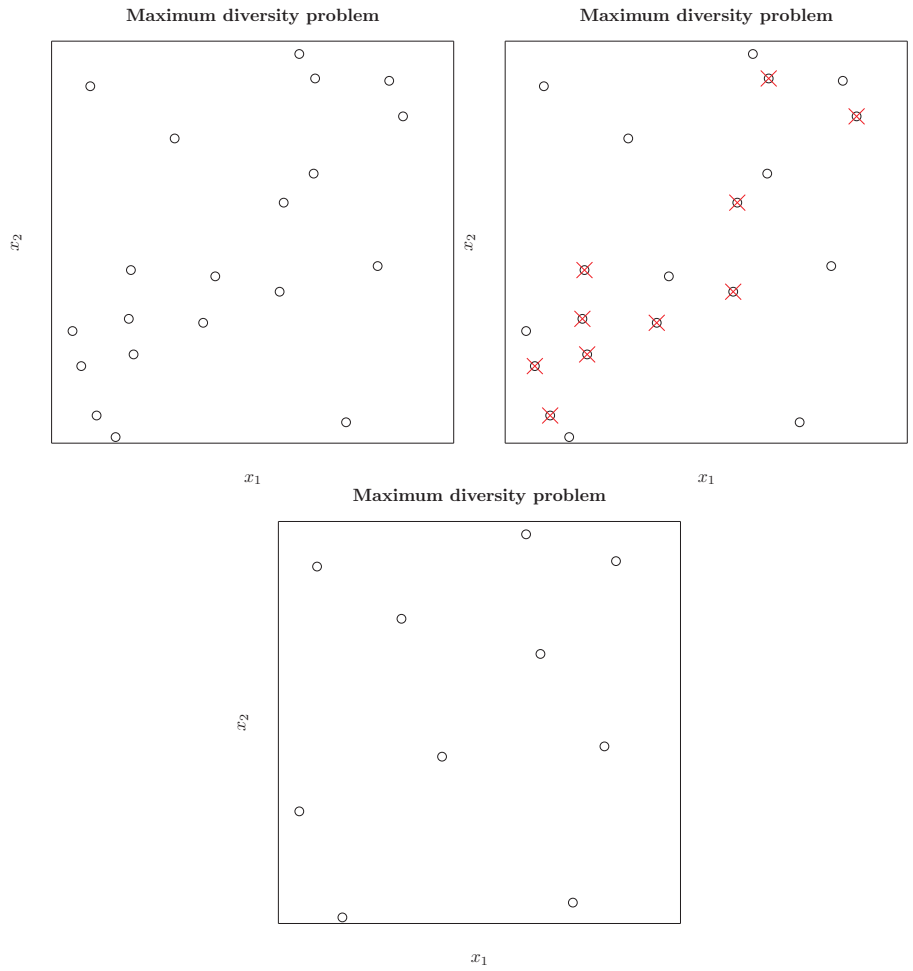


FIGURE 5.3: Maximum Diversity Problem example: finding a set of 10 points out of 20 that provides the maximum diversity. The top left figure shows the original set, while the top right the points to be removed. The bottom figure illustrates the subset with the best diversity.

problem is to find the subset $\mathcal{M}$ which has the greatest minimum distance between its elements (you may wish to read this sentence more times to fully understand it).

In order to write the mathematical formulation, assume $b_i \in \{0, 1\}$ is a binary variable with value 1 if the $i$-th element belongs to the final subset $\mathcal{M}$, or 0 otherwise. Then, the maximum diversity problem can be described as:

$$\begin{aligned} \text{maximize} \quad & \min_{\substack{i,j=1,2,\ldots,n \\ j \neq i}} d_{i,j} b_i b_j \\ \text{subject to} \quad & \sum_{i=1}^{n} b_i = m \\ & b_i \in \{0, 1\} \end{aligned} \tag{5.2}$$

in which $n$ is the total number of elements, and $m$ the cardinality of the resultant subset $\mathcal{M}$ with possibly maximum diversity. Check [63] for alternative formulations.

The Maximum Diversity Problem is useful in a lot of fields. For example, in a talent's show, it may be nice to find a group of judges with diverse opinions so the outcome is not too biased. The other application that hopefully is clear by now is in the selection operator of a multi-objective EA. I will translate it for the present context: given a set of $n$ points equally preferable in the region of interest, find the subset of $m$ points with the best diversity. This is my idea: use as a diversity criterion of EAs the MDP approach.

### 5.3.2 Solving the MDP

So, how to solve this problem? Unfortunately, it is shown [63] that the MDP is NP-hard. As an example of its implications, consider the following: the sure way to find the best subset is by picking all different subsets of $m$ points out of $n$, and keeping the one with highest diversity. So, there are $\begin{pmatrix} n \\ m \end{pmatrix}$ different combinations to try. Suppose a typical case in EAs where you have a population of 100 points, and there are 200 (parents plus offspring) at disposal. To guarantee an optimal solution, there are

$$\begin{pmatrix} 200 \\ 100 \end{pmatrix} = \frac{200!}{100!100!}$$

different combinations to try. This is a really big number. The software Octave gives as a result something near $9.0549 \times 10^{58}$, but with a warning of "almost infinity". What is important is that it doesn't matter how powerful your computer is; if you perform all possible combinations in each iteration, it is probable that not even the grandchildren of your grandchildren will see the final iteration of your algorithm.

Being aware of that, the researchers provided different formulations using diverse methods, some exact with limitations [65], and other using meta-heuristics [66] to provide faster results, even if they are sometimes just approximations. The approach taken here is similar to the last one.

I propose to use a greedy method, which is very simple to understand. First, start with a feasible pair of points (two points that would probably belong to the best subset). For this work, I suggest to begin with the two points that are furthest apart (in case of ties, pick one such pair randomly). Then, check all of the other remaining points and include the one that generates the greatest minimum distance between all pairs. This is repeated until all $m$ points are included. Even if optimality is not guaranteed,

the result is expected to be "good enough" for practical purposes. See Figure 5.4 for a comparison between the executions of the DEMO using the regular crowding distance (top left) and the MDP approach proposed here (top right) solving the DTLZ2 function [67] (optimal-front in the bottom), considering diversity in the objective space.

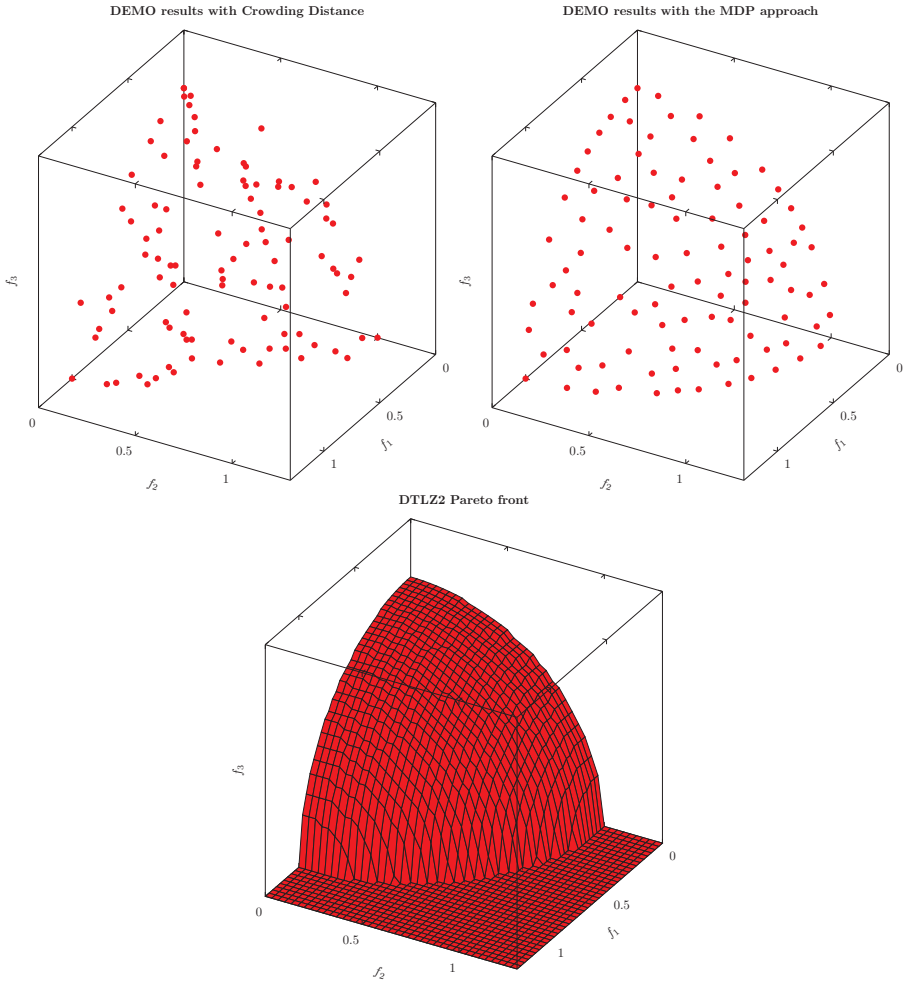


FIGURE 5.4: Typical execution of DEMO solving the DTLZ2 test function using the crowding distance (top left) and the MDP approach (top right), promoting diversity in the objective space. The optimal front of DTLZ2 in three objectives is the positive octant of a sphere. At least qualitatively, the MDP approach yielded more uniform results than the crowding distance. NOTE: both results have similar distance values to the efficient set, so the big difference is in the diversity.

### 5.3.3 Final remarks

This method does not necessarily has to promote diversity in the objective space. It may work in the variables' space as well. Why, you may ask, would diversity in this space be desired? Well, if you can make a car with the same performance using two different fuels (of similar price), you have two good alternatives for the same goal (you may even produce both cars and sell them on different places, probably matching the

type of fuel they use in each location). If you have five different people with equally good curricula which live in very different places, you have five good alternatives (you can even assign each person to a different branch of your company). In summary, diversity in the variable space can be an even better concept than diversity on objective space, and some recent studies are considering it [68–70], even tough research in this field is still in its early stages.

Another important remark is: there is no guarantee that including the MDP approach in the algorithm the points will be equally spaced in the region of interest, *even if an exact solving procedure was used*. This is because the method is "given a set of $n$ points, find the $m$ ones that generate the best diversity", and not "given a region of interest, like a hyper-square, find a subset of $m$ points with the best diversity". In other words, the method is constrained to the points generated by the algorithm; it only stresses the best combination found so far. The only way that the previous statements coincide is if, by chance, the algorithm generate perfectly spaced individuals, and any other combination will not be better than that.

## 5.4 Including the method in an EA

I have described how to define the points that are inside the region of interest (ROI), and also how to promote diversity within this region using the maximum diversity problem (MDP) approach. The final task is to show how this method can be included in your favorite Evolutionary Algorithm.

The steps of a typical iteration are described as follow. Notice that any EA that has a population of parents and creates offspring suits my needs[3].

1. First, given a population of parents, perform the usual variation operators to create the offspring, and compute its objective values;
2. With the combined population $\mathcal{R}(t)$, execute the non-dominated sorting, separating it in ranks of dominance. Then, fill the next population with points from the first rank, then with the second, until the, say, $k$-th rank has too many individuals;
3. Using this $k$-th front, find the points that are inside the region of interest by using the method described in section 5.2: (i) compute the achievement scalarizing function (ASF) of each solution; (ii) find the closest one; create the auxiliary

[3] I have not thought about EAs that use an external archive. I will possibly, in a future work, include this approach, but, for now, this kind of algorithm is left out.

reference points; (iii) associate each of these with their closest individuals in this front; and (iv) keep only the points between these limiting solutions;

4. If the points inside this region are enough to fill the population, stop and go the next iteration. If there are too few, include more individuals from this rank by picking the ones with smaller ASF value. If there are too many, then choose the ones that guarantee best diversity according to the MDP procedure described before.

## 5.5 Summary

This chapter described the proposed method to use the DM's preferences in order to solve problems with many objectives. The region of interest is created by an extension of Wierzbicki's method (presented in section 3.2.2.1), and it assumes the points inside this region are equally preferable to the DM. In order to organize the solutions inside it, a diversity approach based on the Maximum Diversity Problem is adopted.

Once this technique was described, it is necessary now to verify its performance when solving many-objectives problems. The next chapter will execute this and other methods in a benchmark of test problems in order to compare their results.

# Chapter 6

# Experimental Results

Whenever you create a new algorithm (or a method), you probably need to test its performance and possibly compare its results with the ones of different algorithms (or methods). Sometimes it is possible to assess this performance analytically (like it is normally done with deterministic algorithms under some conditions [4]), but, since evolutionary algorithms have a stochastic nature and their results may vary from execution to execution, the evaluation of their performance needs to be done by means of experiments [71].

This chapter is devoted to the analysis of results of some EAs in problems with many objectives. The first section explains how the quality of Pareto front approximations can be measured, the second describes the design of experiments used for the comparisons, and the third discuss the results.

## 6.1 Quality assessment

In single-objective problems, it is conceptually easy to compare the results of two or more algorithms: the one that achieves solutions with smaller objective values (assuming minimization) will be the best. It is possible to include other tests, like the time required to achieve a final solution, or maybe the number of function evaluations used for that etc.

In vector optimization, especially in the field of Evolutionary Algorithms, the outcome of a complete execution is normally a set of non-dominated points. Thanks to their partial ordering property, the comparisons among different sets is not so immediate as in the scalar case.

A first attempt to perform the comparisons is by use of the Pareto-dominance, but this time applied to whole sets. In this case, we talk about a set $\mathcal{P}_1$ *outperforming* another set $\mathcal{P}_2$ [72]. So, for instance, in Figure 6.1, the set $\mathcal{P}_1$ completely outperforms $\mathcal{P}_2$ and $\mathcal{P}_3$, because, for each element $\mathbf{y} \in \mathcal{P}_2$, there is at least one element $\mathbf{x} \in \mathcal{P}_1$ such that $\mathbf{x} \prec \mathbf{y}$, and similarly for $\mathcal{P}_1$ and $\mathcal{P}_3$. The same can be said about $\mathcal{P}_2$ outperforming $\mathcal{P}_3$. In [72], some variations of this outperformance relation are also studied.

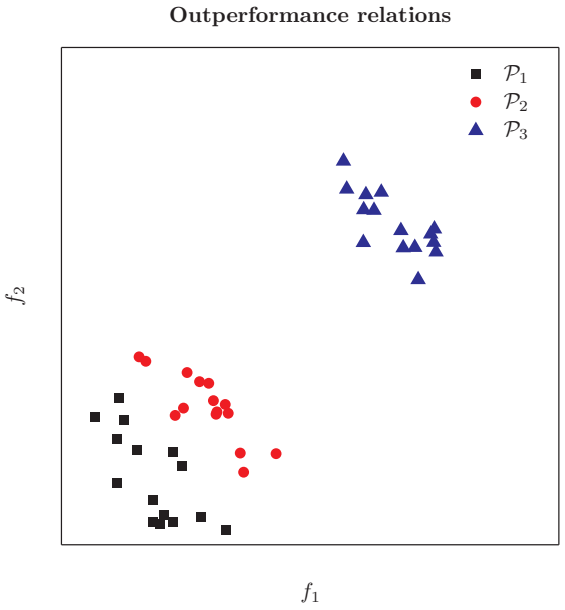


FIGURE 6.1: In this figure, it can be said that $\mathcal{P}_1$ is better than both $\mathcal{P}_2$ and $\mathcal{P}_3$, and that $\mathcal{P}_2$ is better than $\mathcal{P}_3$ according to the outperformance relations.

With this extension, it is possible to make comparisons in the same way as is done for single-objective optimization: the set that outperforms the other sets will be winner. However, this approach has basically two limitations. First, especially for many objectives, the chances are great that two or more sets will not be outperformed, and then they will be incomparable among themselves. Second, even when there is a complete outperformance, this test is only qualitative, that is, it does not tell *how much* better is a set with regard to another. In order to overcome these limitations, a different approach will be followed here by adopting quality indicators [12].

Indicators were already presented in section 2.4.3.2. Basically, if you have, say, $N$ sets of points to be compared, it is possible to employ an $N$-ary indicator $I(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N)$ to compare all of them at once. One indicator that follows this approach is presented in [73]. However, in practice, unary indicators are preferred [12]. In that way, each set $\mathcal{P}_i$ will receive a number $I(\mathcal{P}_i)$ indicating its quality, and the set with the best indicator value will be considered the winner.

It is important to point that the use of an indicator implicates some kind of preference by the analyst. For example, if, in the situation of Figure 6.2, the analyst decides to

use a convergence index, it is clear that $\mathcal{P}_1$ will be better, because it is closer to the efficient front. However, $\mathcal{P}_2$ has points closer to the reference point, which could mean that they represent better the DM's preferences, so they could be preferred. Therefore, in this case, the convergence index is either inadequate or insufficient.

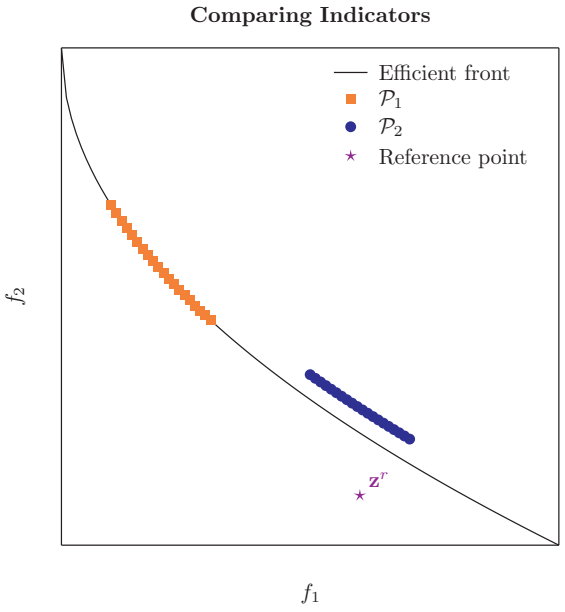


FIGURE 6.2: According to a convergence indicator, $\mathcal{P}_1$ would receive a better value than $\mathcal{P}_2$. However, for an indicator that measures the degree of satisfaction of the DM, the second set, since it is closer to $\mathbf{z}^r$, would be the winner.

In this work, I am interested in checking the convergence capabilities of the algorithms, their capacity of satisfying the DM by generating solutions as close as possible to a reference point, and the ability to maintain a diverse set of solutions within the region of interest. I will, therefore, employ three different indicators, to be described in the section 6.2.3.

### 6.1.1 Handling stochasticity

The EAs have a stochastic nature, indicating that their final populations will probably be distinct in each execution, and so will be their indicator values. Therefore, it is necessary to adopt statistical methods in order to perform the comparisons.

Assume a typical evolutionary algorithm $A$ and a given indicator $I$. Among all possible outcomes $A$ may result (for possibly different test functions, with distinct number of objectives etc.), their indicators will have a mean value $\mu_A$, which is fixed. So, in order to compare $A$ with $B$ under the indicator $I$, it is possible to compare their mean values $\mu_A$ and $\mu_B$, and the algorithm with best mean will be the winner.

Unfortunately, this parameter may be fixed, but it is unknown and unfeasible to compute. A practical overcome to this is to generate a sample of observations and try to *estimate* it by the information provided in the data. But, since the goal of this section is to compare the performance of some algorithms, instead of actually estimating the mean, we can generate confidence intervals [74] with the difference of the indicator values of these algorithms. Therefore, the possible conclusions to be taken from this analysis are of the type "Under a significance level $\alpha$, algorithm $A$ is significantly superior (or inferior, or even not significantly different) to $B$ considering the indicator $I$".

## 6.2 Experimental Setup

A typical scenario of a comparison involving two or more EAs can be described as [75]:

1. Select the algorithms to be compared;
2. Choose a set of existing (possibly benchmark) test problems, or create a new set;
3. Choose a method for comparing the results of each EA (like dominance, indicators, or attainment functions);
4. Obtain results for each algorithm on each test problem, whether from previous references (like in the web) or by software implementation;
5. Generate the observations and compare the data;
6. Draw conclusions.

In this section I will describe the test problems, the algorithms to be compared and the indicators used to assess the quality of each one.

### 6.2.1 Benchmark test functions

Benchmark problems in multi-objective optimization is a field that has not received the necessary attention over the last years, especially for scalable test problems in order to be used in the many-objective context. Huband et al. presents in [75] a survey of some benchmark problems presented for Evolutionary Algorithms.

For this work, I have chosen as test functions the DTLZ family [67], namely the functions DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6 and DTLZ7 (their formulation can be found in the previously cited paper). These functions have the desirable property

of being scalable to any number of objectives, so they are well suited for the present purposes. Also, their objectives are already standardized, which can be a liability when trying to mimic real life problems but, since the present algorithms require this characteristic from the objectives, this feature will actually be useful for the tests.

The DTLZ family has another characteristic that each problem has dimension $n = (m-1)+k$, wherein $m$ is the number of objectives, and $k$ is an integer parameter with a different value depending on the test function: 5 for DTLZ1, 20 for the DTLZ7, and 10 for the rest. The variable vector $\mathbf{x}$ has the following structure:

$$\mathbf{x} = [\underbrace{x_1\ x_2\ \ldots\ x_{m-2}\ x_{m-1}}_{m-1 \text{ position variables}}\ \underbrace{x_m\ x_{m+1}\ \ldots\ x_{n-1}\ x_n}_{k \text{ distance variables}}]^T$$

That is, the last $k$ variables, called *distance variables*, need to achieve a given value $V$ (which is 0.5 for DTLZ1 to DTLZ5, and 0 for the remaining two) in order to the point belong to the efficient front. The first $m-1$ components, named *position variables*, are varied from 0 to 1 in order to map the Pareto front, assuming the last $k$ components have the proper value.

For this work, I considered all of these problems with the following number of objectives: 5, 10, 15 and 20.

### 6.2.2 Algorithms compared

All of the methods here have as a base algorithm the Differential Evolution for Multi-objective Optimization (DEMO), presented in section 2.4.3.4. The adopted parameters of the algorithm are, as used in the original paper [32]:

- Population size $\mu$: 100;
- Scale factor of the mutation $F$: 0.5;
- Crossover factor of the recombination $CR$: 0.3;
- Stopping condition: Stop the execution after 30,100 function evaluations[1] (corresponding to 300 iterations).

This population size is very common in two-objectives problems, even tough there is no exact explanation for this choice. For the number of objectives considered, the

[1]The first population requires 100 evaluations before the iterations effectively start to be counted.

problems' dimensions range from 9 (DTLZ1) to 14 (DTLZ7) with 5 objectives, and from 24 (DTLZ1) to 39 (DTLZ7) with 20 objectives. Even if there is no exact relationship between $n$ and $\mu$ or between $m$ and $\mu$, the number 100 is possibly not too large for these experiments.

With regard to the methods, I will compare the performance of the following techniques:

- Original DEMO with non-dominated sorting and crowding distance [32];
- Indicator-based Evolutionary Algorithm (IBEA) [24] with the additive $\epsilon$-indicator and parameter $\kappa = 0.05$ in the fitness assignment (equation (2.17));
- Preference-based Evolutionary Algorithm (PBEA) [43] with the additive $\epsilon$-indicator and parameter $\kappa = 0.05$ in equation (2.17) and $\delta = 0.01$ in (3.6);
- Deb's reference point (DEB) method [42] with a fixed $\epsilon = 0.001$;
- The method proposed here with the MDP approach for diversity maintenance, called here MyIdea.

The methods PBEA and the proposed one that use an achievement scalarizing function will adopt equation (3.3) with $\rho = 0.001$.

### 6.2.3 Quality indices

It is desired in this work to evaluate the capacity of each method to help the evolutionary algorithm in producing solutions the closest possible to the efficient front, satisfying the DM, and producing diverse solutions. I will describe the quality indices[2] used to assess these demands.

#### 6.2.3.1 Mean distance to the Efficient front

The first criterion can be understood as a convergence condition: the final population needs to have the smallest mean distance (considering all individuals) to the Pareto front.

There are some indices used to compute this distance to the optimal front, among which one very common is the *generational distance* [76]. It requires the generation of $N$

[2]Some studies talk about "quality metrics" or "performance metrics". However, as pointed in [76], "metric" is a well-defined term in mathematics, and many indicators used in MOO do not satisfy the conditions to receive such title. Therefore, in this work I will refer to them as indices or indicators.

efficient solutions $\mathbf{z}_k^*$, $k = 1, 2, \ldots, N$, and them, for each point $\mathbf{x} \in \mathcal{P}$, where $\mathcal{P}$ is the final population of an algorithm, compute the (Euclidean) distance to the closest efficient point:

$$dist(\mathbf{x}) = \min_{k=1,2,\ldots,N} \sqrt{\sum_{i=1}^{m} (f_i(\mathbf{x}) - z_{k,i}^*)^2}$$

and the Generational distance will be the average distance among all individuals in the population. Other convergence indicators can be found in [76].

It may be clear that, in order to have an accurate result, the value of $N$ must be large, especially when the efficient front is big, which usually happens with many objectives.

In order to avoid the high computational cost a big value of $N$ may cause (or a non-accurate result when it is not high enough), I propose here a different method that takes advantage of the structure of the DTLZ variables. Since the last $k$ components of each variable must be equal to a value $V$ (which is 0 or 0.5 depending on the problem), instead of measuring the distance in the objective space, I compute it in the variables' space:

$$d(\mathbf{x}) = \sqrt{\sum_{i=n-k+1}^{n} (x_i - V)^2} \tag{6.1}$$

Therefore, the mean distance of the whole population $\mathcal{P}$ will be

$$\bar{d}(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} d(\mathbf{x}) \tag{6.2}$$

wherein $|\mathcal{P}|$ is the cardinality, or number of elements of the population.

This method for computing the distance is more efficient and does not depend on generating samples from the Pareto front.

Some readers may be disapproving the fact that I am using the knowledge of the real Pareto front when in actual problems this information is not available. There are, in fact, some indicators capable of comparing two or more sets in terms of convergence without requiring the efficient front. Nevertheless, it is not enough to show that one algorithm yields better solutions that another. It is even more important to show that the algorithm yields *good solutions* for the given problem!

Figure 6.3 shows the outcomes $\mathcal{P}_1$ and $\mathcal{P}_2$ of algorithms $A_1$ and $A_2$, respectively, for a two-objective problem. Based just on these two results, one could say "Algorithm $A_1$ is more suitable for solving the problem than $A_2$". However, as the figure shows, they are both far from the efficient front, and a more appropriate answer would be "Neither are good options for solving this problem". So, in my opinion, if the efficient front is known, then it should be used to avoid this issue and prevent the literature from getting overloaded with equally awful algorithms (for an example, see the results reported by Silva et al. [77]).

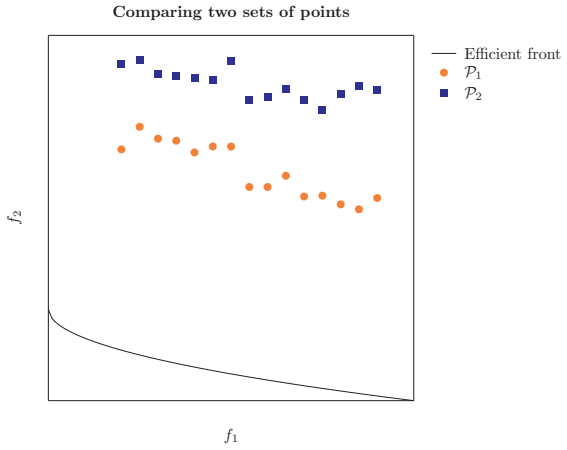


FIGURE 6.3: Comparing the results of two different algorithms. It is clear that $\mathcal{P}_1$ has better solutions than $\mathcal{P}_2$, but, in reality, both sets are far from the efficient front and, thus, represent bad results.

#### 6.2.3.2 Satisfying the DM's preferences

In the experiments, I assume the DM is satisfied by the minimization of an achievement scalarizing function. So, given a reference point $\mathbf{z}^*$, his desires will be better achieved for smaller ASF values.

I propose then to compute here the *minimum* ASF of the whole population:

$$s_{min}(\mathcal{P}) = \min_{\mathbf{x} \in \mathcal{P}} \{s(\mathbf{f}(\mathbf{x}), \mathbf{z}^r)\} \tag{6.3}$$

In that way, the algorithms that do not use preferences but that try to approximate the whole Pareto front may be expected to have at least one point close to this reference point. This single point will be the one responsible for the minimization of this indicator.

#### 6.2.3.3 Diversity

Diversity indicators sometimes measure the distribution or the spread of a population, and sometimes both [76]. I define here *distribution* as the uniformity of a non-dominated set, and *spread* as the extent covered by the points. Typically, this last one is measured by the distance of the boundary solutions in $\mathcal{P}$.

The individual measure of these two indices is not enough to elect a best set. In Figure 6.4, set $\mathcal{P}_1$ has a better uniformity while $\mathcal{P}_2$ is more spread on the Pareto front, so it is hard to tell which one is the best. Because of that, it is customary to use indicators that measure both criteria at the same time. Refer to [72], [78], [79], [80], [81] and [82] for some references about diversity indicators.

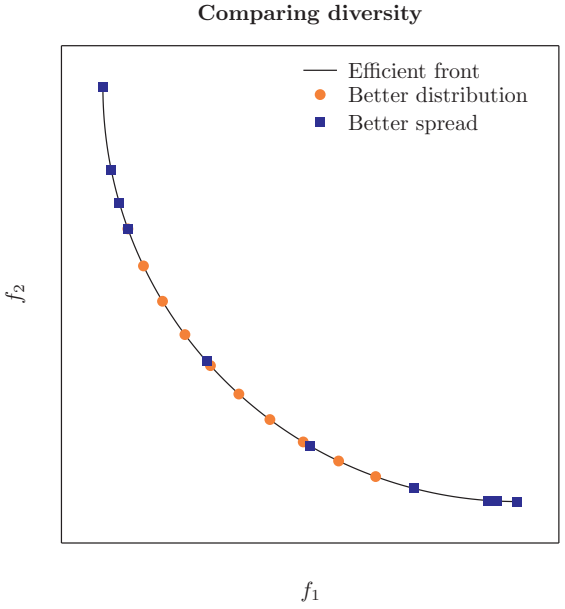


FIGURE 6.4: Comparing spread and uniformity of two sets. $\mathcal{P}_1$ (indicated by circles) has a more uniform set, but $\mathcal{P}_2$ (indicated by squares), even if not so well-distributed, covers a larger region of the front, having, them, a better spread.

It is actually hard to choose such an indicator. References [83] and [84] show some properties a good diversity indicator $\tau(\cdot)$ should have. The most important are:

- **Monotonicity in differences:** the diversity should increase if a new solution is included in the set, that is, $\tau(\mathcal{P} \cup \{\mathbf{y}\}) > \tau(\mathcal{P})$, if $\mathbf{y} \notin \mathcal{P}$;
- **Twin property:** whenever a point that is already in the set is included again, the diversity should not change, that is, $\tau(\mathcal{P} \cup \{\mathbf{y}\}) = \tau(\mathcal{P})$, if $\mathbf{y} \in \mathcal{P}$. This is reasonable, because identical solutions represent in fact only one alternative, so they should not count;

- **Monotonicity in distances:** Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same number of elements, and there is a one-to-one function $\phi(\cdot)$ mapping $\mathcal{P}_1$ onto $\mathcal{P}_2$. If

$$d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) \geq d(\mathbf{x}_1, \mathbf{x}_2), \quad \forall \mathbf{x}_1 \in \mathcal{P}_1, \forall \mathbf{x}_2 \in \mathcal{P}_1, \ \mathbf{x}_1 \neq \mathbf{x}_2$$

  then

$$\tau(\mathcal{P}_2) \geq \tau(\mathcal{P}_1)$$

  This property simply says that, if the all points of a given set $\mathcal{P}_2$ have inner distances superior or equal to $\mathcal{P}_1$, then the diversity of the first is also superior or equal to the second.

Apart from these properties, it is desirable that the computation of $\tau(\cdot)$ is not influenced by any external parameter, that is, the diversity should depend on the points, not on user-defined parameters.

Unfortunately, many of the diversity indicators proposed in the literature do not respect these properties, or, when they do, are dependable on control parameters. Ulrich et al. use in [83] an indicator that does possess these characteristics, but they require a matrix inversion. Hence, in this work, I decided to use the *Hierarchical Cluster Counting* (HCC), presented by Guimarães et al. in [85].

The idea of the HCC is to agglomerate the points into clusters, until the whole data is comprised in a single one. The basic steps of the method are:

1. Start by assuming that there is a cluster $\mathcal{C}_i$, $i = 1, 2, \ldots, |\mathcal{P}|$ for each different point in a set $\mathcal{P}$;
2. Set the counter $k \leftarrow 1$, the diversity $\tau_0 \leftarrow 0$ and an auxiliary variable $r_0 \leftarrow 0$;
3. Find the clusters $\mathcal{C}_{i1}$ and $\mathcal{C}_{i2}$ with the smallest separation distance. There are different ways of computing this distance, as presented in the original paper [85]. In this work, I adopt the *complete linkage*:

$$d_{CL}(\mathcal{C}_p, \mathcal{C}_q) = \max_{\mathbf{x}_i \in \mathcal{C}_p, \mathbf{x}_j \in \mathcal{C}_q} d(\mathbf{x}_i, \mathbf{x}_j) \tag{6.4}$$

   that is, the distance between two clusters is the (Euclidean) distance between the farthest points in each cluster. Assign this minimal value to $r_k$ and join these clusters into one;

4. Update the diversity as

$$\tau_k \leftarrow \tau_k + (r_k - r_{k-1})(|\mathcal{P}| - k + 1)$$

5. Augment the counter: $k \leftarrow k + 1$;

6. Go back to step 2 until there is only one cluster left. Return $\tau_k$ as a value of the diversity of the set $\mathcal{P}$.

A more detailed explanation of these steps is given in the original paper [85]. The HCC is able to measure both the spread and the uniformity of a set of points, it satisfies the aforementioned properties, and do not require any external parameter.

There is one last comment before moving on: In this text, since different algorithms that work with different ROIs are treated, it is unfair to compare them according to spread. For example, it is expected that the non-preference methods yield a set covering a larger extent of the efficient front. Also, even among the preference-based algorithms, they define the ROI in different ways. This problem could be easily solved if there was a real decision maker with a good knowledge of his region of interest, so the diversity could be computed inside this portion.

Since this is not the case, in order to avoid unjust comparisons, I will adopt the HCC as a uniformity indicator only. For that, the final population of each algorithm will be standardized in the interval $[0, 1]^m$ in the *objective space*, and the diversity will be computed for these transformed data.

### 6.2.4 Reference point choice

Since some of the methods seem to be dependent on the reference point choice (especially MyIdea), for this work, I decided to set it close to the efficient front. For that, a random Pareto optimal solution $\mathbf{x}^*$ was generated, and then a small perturbation $\Delta f$ was added to its objective components, that is,

$$z_i^r = f_i(\mathbf{x}^*) - \Delta f, \; i = 1, 2, \ldots, m \tag{6.5}$$

wherein, here, $\Delta f = 0.1$.

### 6.2.5 Summary of the experimental setup

All of the algorithms presented before will be executed $N = 20$ times in each test function for each number of objectives, and after each execution, the three indicators (mean distance, minimum ASF and diversity) will be computed.

EAs are intrinsically stochastic, and therefore they have random characteristics. However, they will be implemented in a computer software; and computers are deterministic. The best they can offer are pseudo-random numbers. In order to account for that, I used a different generating *seed* for each of the 20 replications, so each algorithm started with the same population in each problem and number of objectives.

In summary, the characteristics of the experimental setup are

- Independent variables, or factors:
  - Algorithms, with five levels: DEMO, IBEA, PBEA, DEB and MyIdea;
  - Test functions, with seven levels: DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6 and DTLZ7;
  - Number of objectives $m$, with four levels: 5, 10, 15 and 20;
  - The seed used to generate the initial population, with 20 levels: from 1 to 20.
- Dependent variables, or response factors:
  - Mean distance to the efficient front;
  - Minimum ASF value;
  - Diversity.

  All of these variables are continuous.

The functions were implemented on the software Octave, which is open source "clone" of the well known MatLab, and the statistical analysis were performed in the R language and environment for statistical computing [86].

The main objectives of this experiment are: (i) to check if the regular Pareto dominance is enough for guaranteeing near-optimal solutions; (ii) to verify if the inclusion of preferences is able to provide selective pressure and improve the solutions, in the three response factors described; and (iii) to compare the proposed method with the other ones. It is desirable to compare their performances *independently* on the test problems, on the number of objectives and on random seed used. Therefore, all of these will be considered *blocking factors* [74].

## 6.3 Results

### 6.3.1 Mean distance

Before proceeding with any computation, it is instructive to visualize the data to see if we can get a previous insight on the results. Figure 6.5 shows the mean distance for each algorithm when executed in each test problem, for $m$ varying from 5 to 20. Since there were 20 replications, the observations in the graphs are expressed by the median of these replications, with a line ranging from the 25th to the 75th percentile. So, if only the point is visible, it means that the variability of the observations is small.

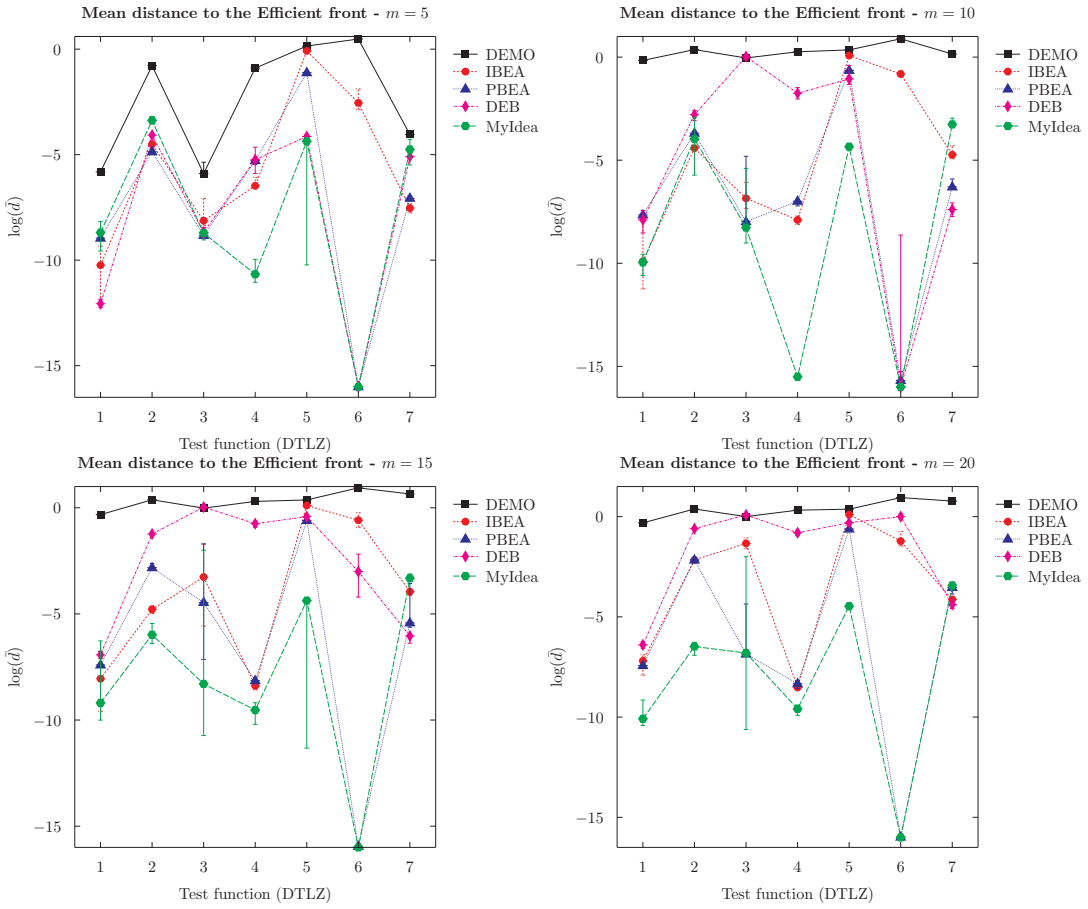


FIGURE 6.5: (Logarithm of the) Mean distance results for each method on each test function, with different numbers of objectives. The smaller the distance, the best the indicator value. All logarithms used base 10.

Since the range of the results was too wide, the distances were plotted as a logarithm:

$$\bar{d}_{log} = \text{mean}\{\log(d + 10^{-16})\}$$

wherein the $10^{-16}$ term was required because some distances had the exact value of 0[3].

Figure 6.5 may be a little bit "heavy" to visualize, but some patterns can be perceived. The DEMO results were basically the worst in all test problems. On the other hand, MyIdea was the best in test problems DTLZ3, DTLZ4 and DTLZ5 for all $m$, getting draws and exchanging places with the other methods for the remaining problems.

The best way to verify these preliminary conclusions is by computing confidence intervals of the differences for each pair of methods [74]. The elected method here is to use *bootstraps* [87]. Being non-parametric tests, they do not require the assumption of normality, and also have better accuracy and are more general than other non-parametric methods. The only necessary requirement is independence of the observations, which is already guaranteed thanks to the nature of the experiment.

The confidence intervals were computed in the software R with the package `boot` [88] with the intention of getting a *familywise error rate* of 0.05, so a 95% of overall confidence can be expected. Since this test requires multiple comparisons, a correction must be applied to achieve that goal. In this work, I use the [89] *Bonferroni* (conservative) correction, that requires the use of the following adjusted significance level:

$$\alpha_{adj} = \alpha / g \tag{6.6}$$

wherein $g$ is the number of planned comparisons. For the 5 different methods, there are $\begin{pmatrix} 5 \\ 2 \end{pmatrix} = 10$ pairwise combinations, so $\alpha_{adj} = 0.005$. Therefore, by computing each individual interval with a 99.5% confidence, the overall intervals will be in the specified value of 95%.

By doing that, Figure 6.6 is generated. To understand the intervals, check, for instance, the pair "myidea - demo". Since it is completely on the left of the line in 0, it can be said that their difference in mean distances is negative, and, because the smaller the distance the better, MyIdea is significantly better than DEMO for this indicator. A corresponding reasoning can be made for intervals completely on the right of line 0. Finally, the intervals crossing this line indicate that the two algorithms are not significantly different at the chosen confidence level.

From Figure 6.6, it can be seen that the DEMO's final population is significantly worse than all of the other methods, suggesting that the pure Pareto-dominance is inferior in many objectives. Comparing the IBEA, which does not take preferences into account,

[3]Note that, by doing this, I am assuming that a mean distance of $10^{-16}$ is considered, for practical purposes, as good as 0.

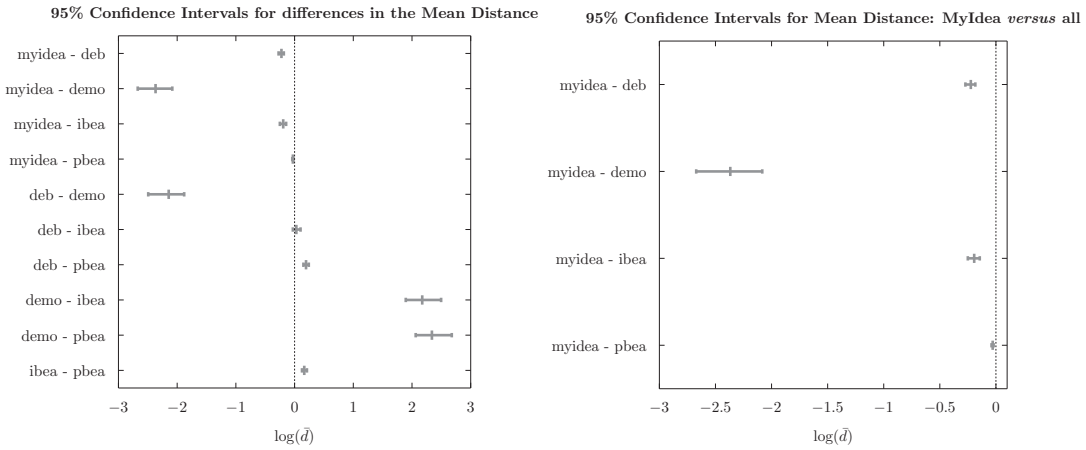


FIGURE 6.6: 95% confidence intervals for differences in the mean distance. The left panel shows all pair-wise differences, while the right one focus only on the pairs involving the proposed method.

with the preference-based methods, it can be seen that it is significantly worse than both MyIdea and PBEA, but it is not different from DEB. It is interesting to see that, by incorporating preferences into the IBEA, its convergence results were improved, as shown in the pair "ibea - pbea". MyIdea, for its turn, is significantly better than all of the other methods. Considering the mean differences, it can be computed that the DEMO's results are $10^{2.368} \approx 233.35$ times bigger than MyIdea's, DEB's method $10^{0.224} \approx 1.67$ times greater, IBEA's $10^{0.194} \approx 1.56$, and PBEA's $10^{0.029} \approx 1.07$. However, for practical purposes, all algorithms excluding the regular DEMO presented similarly good performances in terms of mean distance to the efficient front.

### 6.3.2 Minimum ASF

Figure 6.7 illustrates the results for the $s_{min}$ in the same fashion as Figure 6.5 does for $\bar{d}_{log}$. Again, for better visualization, I used a logarithm transformation in the data. A small modification had to be introduced here, because the ASF can have negative values, which are not allowed for the logarithm. Therefore, all of the data was subtracted by the smallest value present, generating a translated ASF, $\hat{s}_{min}$, and then the transformation followed just like before:

$$\hat{s}_{log} = \log(\hat{s}_{min} + 10^{-16})$$

From Figure 6.7, a similar behavior for the mean distance can be seen here: the DEMO values were the worst in all cases, whilst MyIdea was sometimes in the first place, and other times tied with DEB (for lower values of $m$) and with PBEA, except for the DTLZ7, where these two methods surpassed it. It can also be seen that the differences

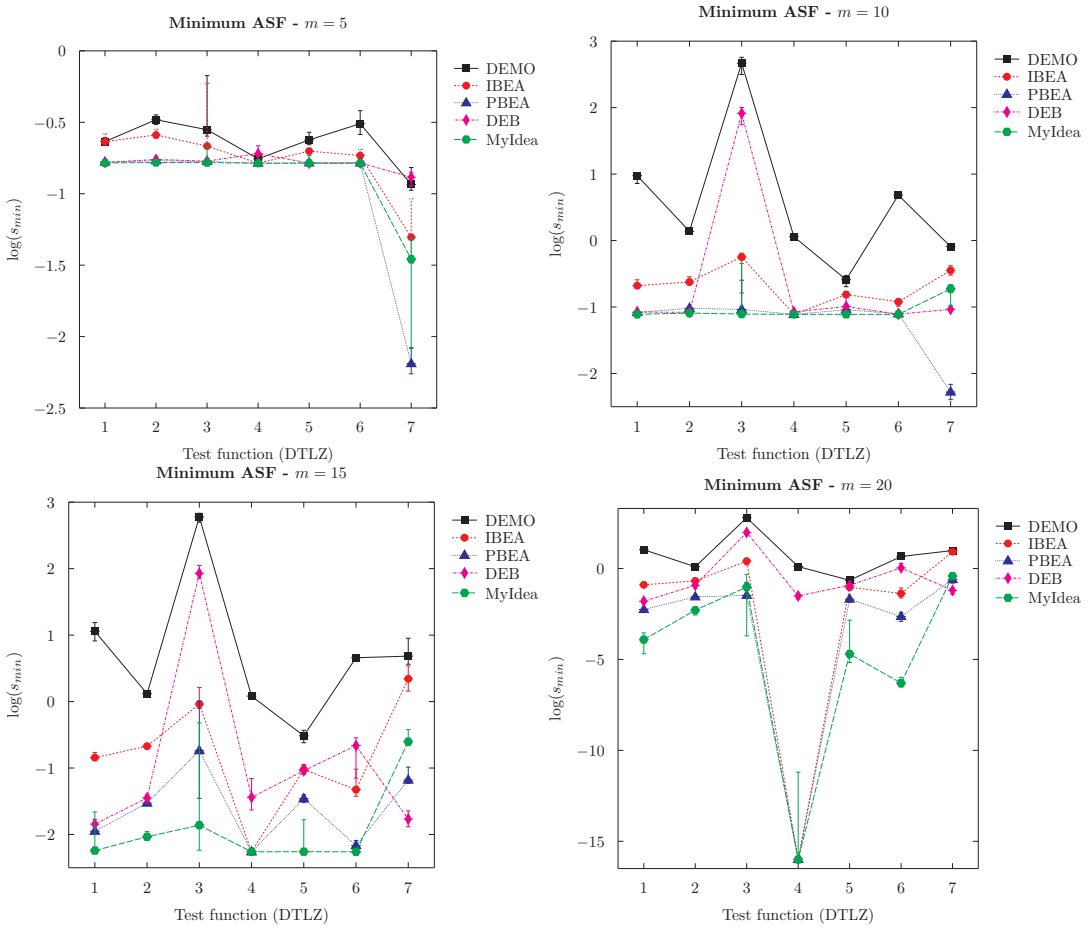


FIGURE 6.7: (Logarithm of the) Minimum ASF for each method on each test function, with different numbers of objectives. The smaller the ASF, the best.

between the DEMO and the other methods (especially MyIdea and PBEA) enlarged as $m$ increased. Notice that, since this is a logarithm scale, if the difference shown in the graph was, say 2, the actual difference is $10^2 = 100$. In fact, these differences were not uncommon in the results.

Let us compute some confidence intervals for the pairwise differences, just as it was made for the mean distance. The familywise 95% confidence intervals for $s_{min}$ are shown in Figure 6.8. Considering that the $s_{min}$ is also better when smaller, it can be seen that the DEMO was the worst among the methods.

The preference-based methods had an advantage here, except for the DEB's technique, which was equally good with IBEA, but worse than MyIdea and PBEA.

From the right panel, it can be seen that MyIdea was significantly better than all methods, except PBEA, whose difference was not significant. This is possibly because of the DTLZ7 function, which, as can be seen in Figure 6.7, was the test that MyIdea did not score very well. If this function was not considered, MyIdea would possibly beat PBEA in this indicator as well.

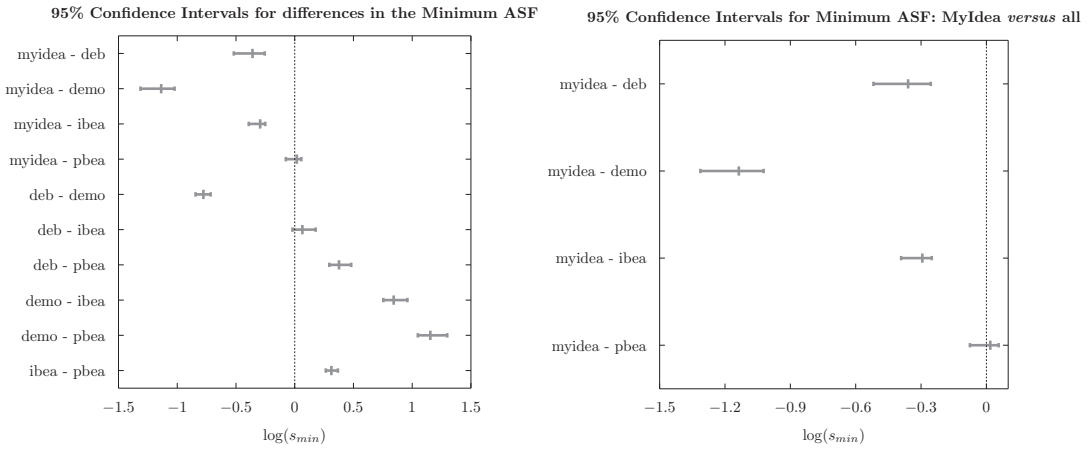


FIGURE 6.8: 95% confidence intervals for differences for the minimum ASF. The left panel shows all pair-wise differences, while the right one focus only on the pairs involving the proposed method.

### 6.3.3 Diversity

Figure 6.9 shows the diversity obtained for each algorithm in each test problem. From these graphs alone it is harder to extract patterns like it was made previously. Considering that, the higher this value, the best, this time MyIdea did not excel like for the other indicators. It alternated between the fourth and even last place, competing with the DEB technique. The remaining methods seem to be alternating in the first three places, so it is very difficult to draw any preliminary conclusion.

The overall differences in performance were evaluated with the help of confidence intervals, as shown in Figure 6.10. This time, apparently the opposite happened: the DEMO's results were the best compared to all of the other methods. It seems that, even if its selective pressure does not work well, at least its diversity mechanism does the job properly.

Another interesting fact is that IBEA's diversity is significantly worse than PBEA's. Since this last algorithm was supposed to concentrate its population in the neighborhood of the reference point while the regular IBEA aims at giving a good approximation of the whole front, this result is quite surprisingly. A possible explanation lies in the fact that the solution sets returned by all algorithms were standardized to the interval $[0, 1]^m$, that is, differences in spread between the "whole Pareto" approaches and the ROI ones were probably neutralized by this data preconditioning.

Finally, when comparing MyIdea with the other methods, it can be seen that it outperforms the DEB technique, but it is significantly worse than the others. The MDP approach was not enough for guaranteeing a significantly better diversity.

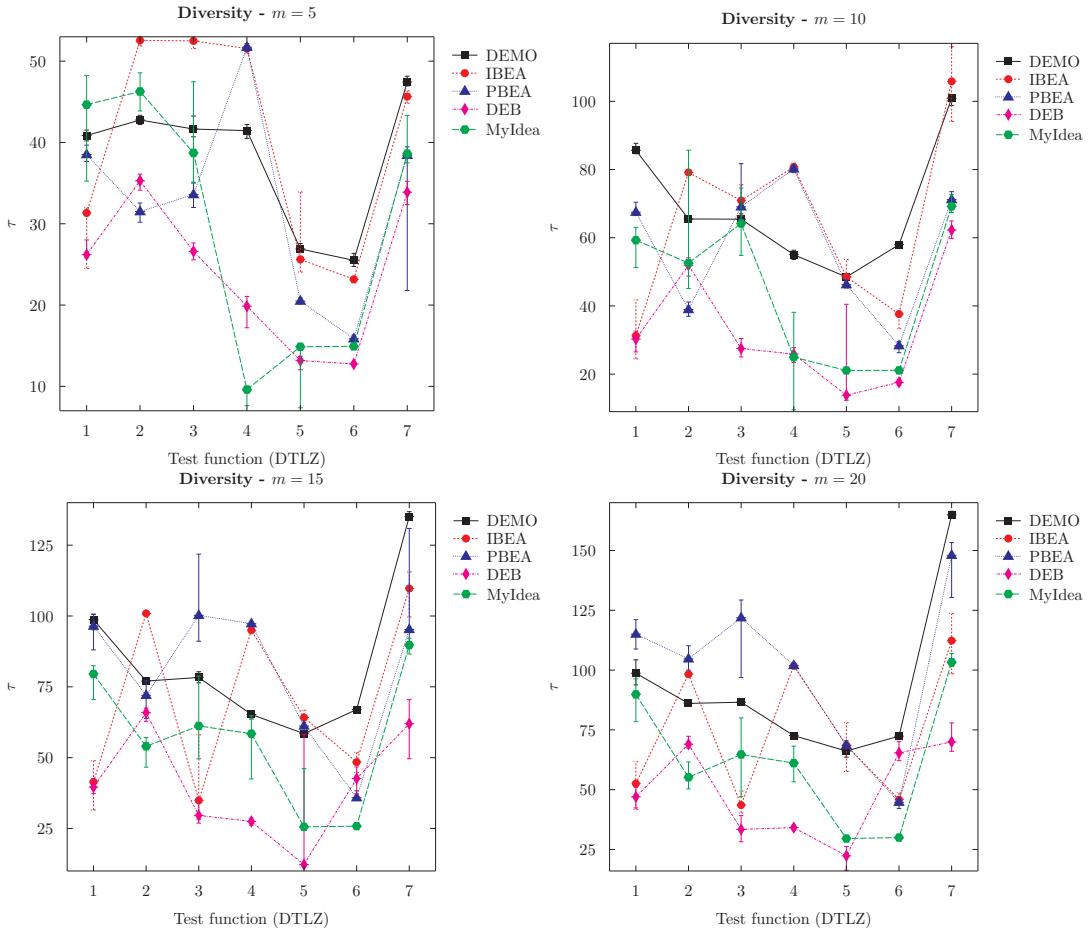


FIGURE 6.9: Diversity results for each method on each test function, with different numbers of objectives. The greater the values, the best.

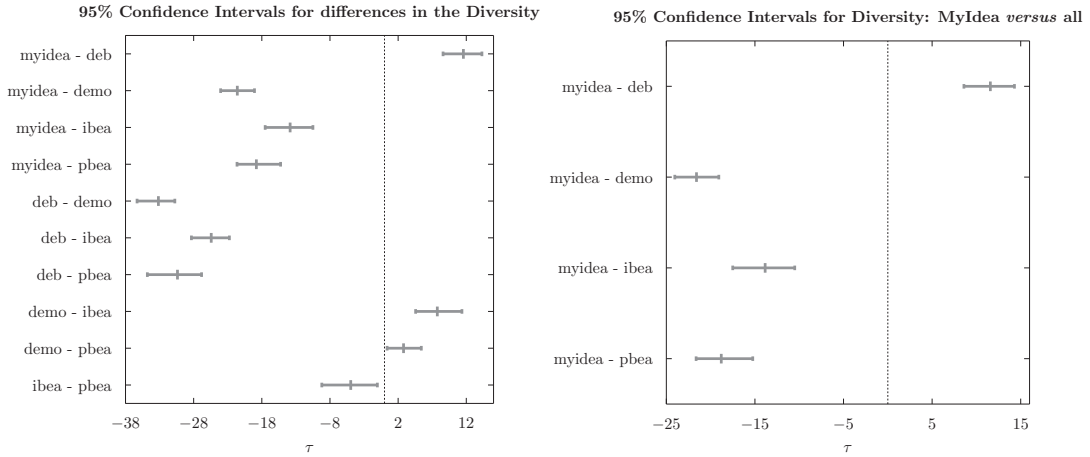


FIGURE 6.10: 95% confidence intervals for differences for the Diversity. The left panel shows all pair-wise differences, while the right one focus only on the pairs involving the proposed method.

One possible reason for this result is that the distance to the reference point was too small, and then the ROI was also tiny. In some cases, the diversity was zero, indicating that the whole population converged into one single point.

This leaves us with an impasse: either choose a further reference point and get a better diversity but a possibly bad convergence, or pick a closer one with solutions with smaller distance to the Pareto front but with an awful diversity. This issue, unfortunately, needs to be postponed to a future work.

### 6.3.4 Discussion

Now, let us check the goals of this work, described in section 6.2.5. First, according to the results of mean distance and minimum ASF, it can be seen that the pure Pareto-based DEMO is not capable of generating a final population with a good convergence as the other methods do, and also it is not reasonable to count that at last one of its individuals will be close to a given reference point, so the DM's preferences are far from being satisfied.

Second, as can be seen by comparing MyIdea and DEB with DEMO, the inclusion of preferences did improve the convergence and the minimum ASF that can be achieved, with the cost of reducing the diversity. However, this should not be taken as a general rule, because all three methods use different diversity management indicators, and the ROI's size seems to influence in the result of MyIdea. A similar comparison can be made between the IBEAs: the PBEA had a significant improvement over all three subjects.

Third, when comparing all three preference-based methods, it can be seen that DEB got the worst results among them, while MyIdea was significantly better in convergence than PBEA, equally good in the DM's satisfaction, but significantly worse in the diversity. The best that can be inferred about these considerations is that the selection of $\mathbf{z}^r$ induced a ROI such that the selective pressure was enough to guarantee these results to MyIdea, but, since it is so dependable on the reference point, the results may be very different if other values of $\mathbf{z}^r$ are used.

Finally, it can be said that PBEA and MyIdea are promising approaches for solving problems with many-objectives. A practical issue with my presented method is that it still is too dependent on the reference point's choice, while PBEA has no control over the region of interest, so it would not be possible to fine-tune the search, if so desired. The solution for the setback of MyIdea can be proposed in future studies.

## 6.4 Summary

This chapter dealt with the comparison of the performances of different algorithms when executed in some benchmark problems. The test bed was chosen as the DTLZ

family; the methods compared were the regular DEMO, DEB, IBEA, PBEA and the proposed method, referred here as MyIdea; and the indicators used to assess the quality of their outcomes were the mean distance to the efficient front (a convergence indicator), the minimum ASF value achieved (the satisfaction of the DM's preferences), and the Hierarchical Cluster Counting (as a diversity indicator).

The results showed that the proposed method was significantly better than all of the other algorithms in terms of convergence and also in tems of DM's satisfaction, while the difference in this last indicator was not significant when compared to PBEA. However, it received the third place in terms of diversity.

It is interesting to see the effect of the inclusion of preferences. MyIdea is basically the regular DEMO that uses the reference point information to break the Pareto-dominance ties, and this small modification allowed it to generate results much superior in terms of convergence. A similar thing can be said about the indicator-based EAs: PBEA was significantly superior than IBEA in both the convergence and the satisfaction of DM's preferences subjects, and surprisingly also better in terms of diversity.

One more important observation is that the maximum budget of function evaluations used is usually set to problems with 2 objectives, and this number normally increases when more objectives are used. In this work I kept this value fixed for all options of $m$, and even with this the algorithms managed to get really close to the efficient front. So, this may be an indicator that it is time to come with a better stopping criteria for MOO than just fixing a maximum number of iterations or evaluations.

Finally, it is important to point that these results are conditioned by the reference point choice, which was purposely close to the efficient front. The outcomes of MyIdea are heavily influenced by this point, so I propose as an idea for future works a solution for this inconvenience.

# Chapter 7

# Conclusion

The solution process of a multi-objective optimization problem involves helping a human decision maker (DM) to choose a most preferred solution. Among the most widely used methods of the literature, Evolutionary Algorithms (EA) are very popular thanks to their ability to approximate regions of the efficient front, or even its whole extension.

Over the last years EAs were used with the aim of approximating the whole Pareto front in mind, and this procedure worked for a lot of real world problems. However, this way of thinking was not suitable for problems with many objectives, typically more than three.

I proposed in this work the incorporation of preferences in an EA in order to better help the DM in problems with many objectives. In this way, the main challenges faced by EAs could be solved more efficiently: there is no need to adopt huge population sizes because only a small portion of the front is required, so the computational cost is reduced; and the evaluation of the alternatives to choose a most preferred one becomes easier. Furthermore, the preferences provide a way for comparing even non-dominated solutions, solving the selective pressure the Pareto-based algorithms lack in many objectives. Also, in order to simplify the process of defining the size of the region of interest (ROI), I proposed a new self-adaptive method, named MyIdea, whose ROI's size depends only on the reference point choice, which is an extension of the method proposed by Wierzbicki in section 3.2.2.1, with a more natural way of thinking.

In order to check the differences in performance, the non-preference based algorithms DEMO and IBEA, and its preference based counterparts DEB, MyIdea and PBEA, were compared in a test suite composed of the DTLZ family, using 5, 10, 15 and 20 objectives. Their outcomes were measured according to their convergence to the efficient front, capacity of generating solutions close to the reference point (DM's satisfaction),

and diversity in the objective space (used here to measure the uniformity). The results imply that the preference based methods were significantly better than its counterparts in both convergence and DM's satisfaction, but MyIdea and DEB were significantly worse than DEMO in the diversity index, while PBEA excelled IBEA in this subject. Also, the proposed method was significantly better than all of the others in terms of convergence, tied with PBEA but over-matched the remaining ones in terms of DM's satisfaction, but achieved the fourth place in terms of diversity.

In practical terms, the methods based on Pareto-dominance (MyIdea and DEB) have their performance dependent on the size of the ROI, such that smaller regions are required for good values of convergence. In this work, then, I assured this condition for DEB by selecting a small value of the $\epsilon$ parameter, and, for MyIdea, I only adopted reference points relatively close to the efficient front. It is possible that, for different values of these parameters, their results would be worse, so PBEA would be the best. The reason why I still considered supplementing the Pareto dominance in MyIdea instead of adopting indicators is that this last approach still cannot control the ROI's size in an easier way, but this technique may be employed in future works.

Therefore, according to these results, the inclusion of preferences is a justified approach to handle problems with many objectives. It is nice to point that, even if this is already a well proposed method in the literature, not many studies that adopt them have the many-objectives field in mind. So, I hope to present another reason for that with this work.

Regarding future studies, I propose to solve the limitations already pointed of MyIdea and compare it in a more general set of benchmark problems. Also, it would be interesting to adopt one of these preference based methods (possibly a revamped MyIdea) to solve a real world problem with many objectives in order to provide a better justification for their use in applied optimization scenarios.

# Bibliography


[1] Sven Ove Hansson. Decision Theory: A brief introduction. Technical report, 2005.

[2] Jasbir S. Arora. *Practical Mathematical Optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 31. 2006. ISBN 0387243488 (acid-free paper)\r9780387243481 (pbk. acid-free paper)\r038729824X (pbk. acid-free paper)\r0387243496 (e-ISBN). doi: 10.1007/s00158-005-0595-0.

[3] S.S. Rao. *Engineering Optimization: Theory and Practice*. A Wiley Interscience publication. Wiley, 1996. ISBN 9780471550341.

[4] D.G. Luenberger. *Optimization by Vector Space Methods*. Professional Series. Wiley, 1969. ISBN 9780471181170.

[5] R.E. Rosenthal. Principles of multiobjective optimization. 1984.

[6] K. Miettinen. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US, 1999. ISBN 9780792382782.

[7] J. Jahn. *Vector Optimization: Theory, Applications, and Extensions*. Springer-Verlag Berlin Heidelberg, 2010. ISBN 9783642170058.

[8] Michael Emmerich and A Deutz. Multicriteria Optimization and Decision Making. *LIACS. Leiden university, NL*, 2006.

[9] A Wierzbicki, Marek Makowski, and J Wessels. *Model-based decision support methodology with environmental applications*, volume 2000. 2000. ISBN 0792363272.

[10] M Ehrgott. *Multicriteria Optimization*, volume 39. February 2005. ISBN 3540213988. doi: 10.1118/1.3675601.

[11] Indraneel Das and John Dennis. Normal-Boundary Intersection: An Alternate Method for Generating Pareto Optimal Points in Multicriteria Optimization Problems. (96), 1996.

[12] J Branke, K Deb, K Miettinen, and R Slowinski. *Multiobjective optimization: Interactive and evolutionary approaches*. 2008. ISBN 9783540889076.

[13] J. E. Dennis I. Das, I. Das, and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Surgical forum*, 6(1):295–6, January 1997. ISSN 0071-8041. doi: 10.1007/BF01197559.

[14] Wlodzimierz Ogryczak and Steve Lahoda. Aspiration/Reservation-Based Decision Support - a Step Beyond Goal Programming. *Journal of Multi-Criteria Decision Analysis*, 1(September 1991):101–117, 1992.

[15] PK Shukla. On the normal boundary intersection method for generation of efficient front. *Computational Science–ICCS 2007*, pages 310–317, 2007.

[16] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, January 2003. ISSN 0036-1445. doi: 10.1137/S003614450242889.

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

[18] J Fliege, LMG Drummond, and BF Svaiter. Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, pages 1–24, 2009.

[19] AL López, CAC Coello, and O Schütze. *Using Gradient Based Information to Build Hybrid Multi-objective Evolutionary Algorithms*. PhD thesis, 2012.

[20] Douglas A. G. Vieira, Ricardo H. C. Takahashi, and Rodney R. Saldanha. Multicriteria optimization with a multiobjective golden section line search. *Mathematical Programming*, 131(1-2):131–161, April 2010. ISSN 0025-5610. doi: 10.1007/s10107-010-0347-9.

[21] AP Engelbrecht. *Computational intelligence: An Introduction*. John Wiley & Sons, 2007.

[22] A Abraham, L Jain, and Robert Golberg. *Evolutionary multiobjective optimization - Theoretical Advances and Applications*. Springer London, 2005. ISBN 9781852337872. doi: 10.1007/1-84628-137-7.

[23] Ruhul Sarker, Joarder Kamruzzaman, and Charles Newton. Evolutionary optimization (EvOpt): a brief review and analysis. *International Journal of Computational Intelligence and Applications*, pages 1–15, 2003.

[24] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. In *Metaheuristics for Multiobjective Optimisation*, pages 3–37. 2004. ISBN 978-3-540-20637-8, 978-3-642-17144-4. doi: 10.1007/978-3-642-17144-4_1.

[25] T. Aittokoski and K. Miettinen. Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization, 2010. ISSN 1055-6788.

[26] Lucas Marti, Jesús García, Antonio Berlanga, and José Manuel Molina. An approach to stopping criteria for multi-objective optimization evolutionary algorithms: The mgbm criterion. In *IEEE Congress on Evolutionary Computation*, pages 1263–1270. IEEE, 2009.

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

[28] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2001. doi: 10.1.1.28.7571.

[29] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132, 2002.

[30] R Storn and K Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11: 341–359, 1997. ISSN 0925-5001, 1573-2916. doi: 10.1023/A:1008202821328.

[31] UK Chakraborty. *Advances in Differential Evolution*, volume 143. 2008. ISBN 978-3-540-68827-3. doi: 10.1007/978-3-540-68830-3.

[32] T Robič and B Filipič. DEMO: Differential evolution for multiobjective optimization. *Evolutionary Multi-Criterion Optimization*, pages 520–533, 2005.

[33] Silvia Poles. Multi-objective optimization and decision making process in engineering design. *Basis of Technology*, 7(3):15–18, 2008.

[34] DW North. A tutorial introduction to decision theory. *... Science and Cybernetics, IEEE Transactions on*, (3), 1968.

[35] P. Tryfos. *Methods for Business Analysis and Forecasting: Text and Cases*. Wiley, 1998. ISBN 9780471123842.

[36] J Levy. An introduction to prospect theory. *Avoiding Losses/Taking Risks: Prospect Theory and ...*, 13(2):171–186, 1994.

[37] D Kahneman and A Tversky. Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, 47(March):263–291, 1979.

[38] G Agrawal, K Lewis, and K Chugh. Intuitive visualization of Pareto frontier for multi-objective optimization in n-dimensional performance space. *Symposium on Multidisciplinary Analysis and Optimization*, pages 1–11, 2004.

[39] Andrzej P. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3(5):391–405, January 1982. ISSN 02700255. doi: 10.1016/0270-0255(82)90038-0.

[40] Yury Nikulin, Kaisa Miettinen, and Tucs Technical Report. A parameterized achievement scalarizing function for multiobjective optimization. Technical Report 969, 2010.

[41] CAC Coello. Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. *Evolutionary Computation, 2000. Proceedings of the 2000*, pages 30–37, 2000.

[42] Kalyanmoy Deb, J. Sundar, Rao N. Udaya Bhaskara, and Shamik Chaudhuri. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. *International Journal of Computational Intelligence Research*, 2(3):273–286, 2006. ISSN 09741259. doi: 10.5019/j.ijcir.2006.67.

[43] Lothar Thiele, Kaisa Miettinen, PJ Korhonen, and Julian Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436, 2009.

[44] UK Wickramasinghe and Xiaodong Li. A distance metric for evolutionary many-objective optimization algorithms using user-preferences. *AI 2009: Advances in Artificial Intelligence*, pages 443–453, 2009.

[45] Tobias Friedrich, Trent Kroeger, and Frank Neumann. Weighted preferences in evolutionary multi-objective optimization. *International Journal of Machine Learning and Cybernetics*, 4(2):139–148, March 2012. ISSN 1868-8071. doi: 10.1007/s13042-012-0083-y.

[46] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Articulating user preferences in many-objective problems by sampling the weighted hypervolume. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, (Gecco):555, 2009. doi: 10.1145/1569901.1569979.

[47] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. *2008 IEEE Congress on Evolutionary Computation IEEE World Congress on Computational Intelligence*, (March):2419–2426, 2008. doi: 10.1109/CEC.2008.4631121.

[48] Kalyanmoy Deb and Himanshu Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. *ieeexplore.ieee.org*, (c):1–1. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281535.

[49] Peter J Fleming, Robin C Purshouse, and Robert J Lygoe. Many-Objective Optimization: An Engineering Design Perspective. *Design*, 3410(6):14–32, 2005. ISSN 15393755.

[50] Oliver Schutze, Adriana Lara, and Carlos A Coello Coello. On the Influence of the Number of Objectives on the Hardness of a Multiobjective Optimization Problem, 2011. ISSN 1089778X.

[51] Thiago Santos and Ricardo H. C. Takahashi. On the performance degradation of dominance-based search algorithms in many-objective optimization. X(X):1–10.

[52] Olivier Teytaud. How entropy-theorems can show that approximating high-dim Pareto-fronts is too hard. In *Bridging the Gap between Theory and Practice - Workshop PPSN-BTP*, Reykjavik, Islande, 2006.

[53] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka. Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs. In *Evolutionary Multi-Criterion Optimization*, volume 4403, pages 5–20. 2007. ISBN 978-3-540-70927-5. doi: 10.1007/978-3-540-70928-2_5.

[54] M Farina and P Amato. A fuzzy definition of "optimality" for many-criteria optimization problems, 2004. ISSN 10834427.

[55] Gaoping Wang and Huawei Jiang. Fuzzy-Dominance and its Application in Evolutionary Many Objective Optimization. pages 195–198, 2007. doi: 10.1109/CIS.Workshops.2007.184.

[56] M. Basseur and E. K. Burke. Indicator-based multi-objective local search. *2007 IEEE Congress on Evolutionary Computation*, pages 3100–3107, September 2007. doi: 10.1109/CEC.2007.4424867.

[57] Pruet Boonma and J Suzuki. PIBEA: Prospect Indicator Based Evolutionary Algorithm for Multiobjective Optimization Problems. *Proc. 2011 IEEE Congress on Evolutionary Computation*, 2011.

[58] Dung H. Phan, Junichi Suzuki, and Isao Hayashi. Leveraging indicator-based ensemble selection in evolutionary multiobjective optimization algorithms. *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*, page 497, 2012. doi: 10.1145/2330163.2330234.

[59] Dung H. Phan and Junichi Suzuki. R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization. *2013 IEEE Congress on Evolutionary Computation*, pages 1836–1845, June 2013. doi: 10.1109/CEC.2013.6557783.

[60] AL Jaimes. *Techniques to Deal with Many-objective Optimization Problems Using Evolutionary Algorithms*. PhD thesis, 2011.

[61] David Corne and Joshua Knowles. Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others. *In Practice*, 1:8, 2009.

[62] Mario Köppen and Kaori Yoshida. Substitute Distance Assignments in NSGA-II for Handling Many-Objective Optimization Problems. In *EMO07*, pages 727–741, 2007. doi: 10.1007/978-3-540-70928-2_55.

[63] Ching-Chung Kuo, Fred Glover, and Krishna S. Dhir. Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming. *Decision Sciences*, 24(6):1171–1185, November 1993. ISSN 0011-7315. doi: 10.1111/j.1540-5915.1993.tb00509.x.

[64] Oleg a. Prokopyev, Nan Kong, and Dayna L. Martinez-Torres. The equitable dispersion problem. *European Journal of Operational Research*, 197(1):59–67, August 2009. ISSN 03772217. doi: 10.1016/j.ejor.2008.06.005.

[65] R Martí, M Gallego, and A Duarte. An Exact Method for the Maximum Diversity Problem. *Citeseer*.

[66] Daniel Cosmin Porumbel, Jin-Kao Hao, and Fred Glover. A simple and effective algorithm for the MaxMin diversity problem. *Annals of Operations Research*, 186 (1):275–293, May 2011. ISSN 0254-5330. doi: 10.1007/s10479-011-0898-z.

[67] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. Technical Report 1990, 2005.

[68] Kamyab Tahernezhadiani, Ali Hamzeh, and Sattar Hashemi. Towards Enhancing Solution Space Diversity in Multi-objective Optimization: A Hypervolume-based Approach. *International Journal*, 3(1):65–81, 2012.

[69] OM Shir, M Preuss, B Naujoks, and M Emmerich. Boosting Decision-Space Diversity in Multi-Objective Optimization using Niching-CMA and Aggregation. 2008.

[70] Tamara Ulrich, Johannes Bader, and Eckart Zitzler. Integrating decision space diversity into hypervolume-based multiobjective search. *Proceedings of the 12th annual conference ...*, page 455, 2010. doi: 10.1145/1830483.1830569.

[71] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Natural Computing Series. Springer, 2006.

[72] Crina Grosan, M Oltean, and D Dumitrescu. Performance metrics for multiobjective optimization evolutionary algorithms. *Proceedings of Conference on Applied and Industrial Mathematics (CAIM), Oradea*, 2003.

[73] Giovanni Lizárraga, Arturo Hernández, and Salvador Botello. G–indicator: An m–ary quality indicator for the evaluation of non–dominated sets. In Alexander Gelbukh and ÁngelFernando Kuri Morales, editors, *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 118–127. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-76630-8. doi: 10.1007/978-3-540-76631-5_12.

[74] G.W. Oehlert. *A First Course in Design and Analysis of Experiments*. W. H. Freeman, 2000. ISBN 9780716735106.

[75] Simon Huband, Phil Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(2006):477–506, 2006.

[76] Tatsuya Okabe. A critical survey of performance indices for multi-objective optimisation. In *Proc. of 2003 Congress on Evolutionary Computation*, pages 878–885. IEEE Press, 2003.

[77] A. L. Silva, J. A. Ramirez, and F. Campelo. A Statistical Study of Discrete Differential Evolution Approaches for the Capacitated Vehicle Routing Problem. *Genetic and Evolutionary Computation Conference*, 2013.

[78] Bakir Lacevic and Edoardo Amaldi. On population diversity measures in Euclidean space. *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010. doi: 10.1109/CEC.2010.5586498.

[79] ES Nicoară. Performance Measures for Multi-objective Optimization Algorithms. *bmif.unde.ro*, LIX(1), 2007.

[80] A. Farhang-Mehr and S. Azarm. Diversity assessment of Pareto optimal solution sets: an entropy approach. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, 1:723–728. doi: 10.1109/CEC.2002.1007015.

[81] GL Lizárraga and SB Rionda. On the Diversity of Non–dominated Sets. *micai.org*, 2009.

[82] LinLin Wang and Yunfang Chen. Diversity Based on Entropy: A Novel Evaluation Criterion in Multi-objective Optimization Algorithm. *International Journal of Intelligent Systems and Applications*, 4(10):113–124, September 2012. ISSN 2074904X. doi: 10.5815/ijisa.2012.10.12.

[83] Tamara Ulrich, Johannes Bader, and Lothar Thiele. Defining and optimizing indicator-based diversity measures in multiobjective search. *Parallel Problem Solving from Nature, PPSN XI*, 2010.

[84] ML Weitzman. On diversity. *The Quarterly Journal of Economics*, (May), 1992.

[85] Frederico G. Guimaraes, Elizabeth F. Wanner, and Ricardo H.C. Takahashi. A quality metric for multi-objective optimization based on Hierarchical Clustering Techniques. *2009 IEEE Congress on Evolutionary Computation*, pages 3292–3299, May 2009. doi: 10.1109/CEC.2009.4983362.

[86] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL http://www.R-project.org/.

[87] Michael Hughes. Introduction to Nonparametric Statistics Using R. Technical Report January, Miami University, 2012.

[88] Angelo Canty and Brian Ripley. boot: Bootstrap r (s-plus) functions., 2014. R package version 1.3-11.

[89] Felipe Campelo. Lecture notes on design and analysis of experiments. http://www.ppgee.ufmg.br/ fcampelo/LNDoE/, 2014. Version 2.1, Ch. 4; Creative Commons BY-NC-SA 4.0.