Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

UNIDAD ZACATENCO

DEPARTAMENTO DE COMPUTACIÓN

Estrategias Meméticas para Métodos de Puntos de Referencia

TESIS

Que presenta

Jesús Alejandro Hernández Mejía

Para obtener el grado de

Maestro en Ciencias

en Computación

Director de la Tesis:

Dr. Oliver Steffen Schütze

México, D. F. **Diciembre 2014**

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

ZACATENCO

COMPUTER SCIENCE DEPARTMENT

Memetic Strategies for Reference Point Methods

Submitted by

Jesús Alejandro Hernández Mejía

as fulfillment of the requirement for the degree of

Master in

Computer Science

Advisor:

Dr. Oliver Steffen Schütze

Mexico, D. F. **December 2014**

Resumen

En un problema de optimización multiobjetivo (POM) la tarea es optimizar varios objetivos concurrentemente que usualmente están en conflicto. Dado que la solución P de tales POMs está dada por un conjunto de $k-1$ dimensiones en donde k es el número de objetivos involucrados, la aproximación de P no es siempre deseada o incluso posible, particularmente en problemas de muchos objetivos, i.e., problemas con tres o más objetivos. En cambio, tiene sentido concentrarse en puntos particulares o regiones del conjunto solución. En caso de que el tomador de decisiones tenga una cierta idea sobre el desempeño esperado de su producto, se pueden usar los problemas de punto de referencia para encontrar soluciones que se parezcan lo más posible a sus preferencias (dadas por los puntos de referencia).

Las técnicas de programación matemática (PM) actuales para puntos de referencia (e.g. el problema de las métricas ponderadas) trabajan sólo con un punto de referencia a la vez. Más aún, pueden garantizar soluciones óptimas, pero sólo localmente. Por otro lado, los algoritmos evolutivos (AE) (e.g. RNSGA-II) son beneficiosos para el tratamiento de tales problemas en particular si hay múltiples puntos de referencia y/o si los objetivos son altamente multimodales, sin embargo, sufren de la desventaja general de velocidad de convergencia lenta.

Por tanto, se vuelve un paso natural hibridar un AE con una técnica de PM para tomar ventaja de ambos enfoques. En este trabajo, investigamos el método de Búsqueda Dirigida (Directed Search) para problemas de punto de referencia y discutimos su integración como motor de búsqueda local dentro de algoritmos evolutivos, concretamente, del algoritmo estado del arte RNSGA-II. Resultados numéricos en varios problemas estándar y de ingeniería (con y sin restricciones, unimodales y mul-

timodales) indican que el nuevo algoritmo memético incrementa significativamente el desempeño de su algoritmo base.

Abstract

The task in a multi objective optimization problem (MOP) is to optimize several objectives concurrently which are usually in conflict. Since the solution P of such MOPs is typically given by an entire set of dimension $k-1$, where k is the number of objectives involved, the entire approximation of P is not always desired or even possible, particularly in many objective problems, i.e., problems with three or more objectives. Instead, it makes sense to concentrate on particular points or regions of the solution set. In case the decision maker has a certain idea about the expected performance of his/her product, reference point problems can be used to find solutions that more closely resemble their preferences (given by reference points).

Current mathematical programming (MP) techniques for reference point problems (e.g. the weighted metrics problem) work with only one reference point at a time. Further, they can guarantee optimal solutions, but only locally. On the other hand, evolutionary algorithms (EAs) (e.g., RNSGA-II) are advantageous for the treatment of such problems in particular if there are multiple reference points and/or the objectives are highly multimodal. However, they suffer the general drawback of relative slow convergence rates.

Hence, it becomes a natural step to hybridize an EA with MP techniques to take advantage of both approaches. In this work, we investigate the Directed Search method for reference point problems and discuss its integration as a local search engine into evolutionary algorithms, namely, the state-of-the-art RNSGA-II. Numerical results on several benchmark and engineering problems (constrained and unconstrained, unimodal and multimodal) indicate that the novel memetic algorithm significantly increases the performance of its base algorithm.

Agradecimientos

Al CONACyT y al CINVESTAV, por la gran experiencia que me ofrecieron durante estos dos años.

A los investigadores del departamento de computación que siempre estuvieron dispuestos a compartir su conocimiento.

A las secretarias, Sofía, Erika y Felipa, por su gran actitud y apoyo.

Al Dr. Oliver Schütze, por brindarme el honor de trabajar como su alumno de tesis, por la gran paciencia y la fe que tuvo durante el desarrollo de la misma y por las experiencias que obtuve gracias a él.

Al Dr. Kalyanmoy Deb y a sus alumnos, cuya visita y consejo ampliaron mi percepción del mundo.

A mi madre, Haydee, pues sin su apoyo y enseñanzas no sería quien soy hoy ni hubiera llegado a este punto.

A mi padre, Noel, que siempre llevo en mi corazón.

A mi hermano, Ricardo, que ha sido un ejemplo a seguir.

A mis amigos del departamento de computación, que fueron un apoyo invaluable.

A mi mejor amigo Mario, con quien siempre se puede contar.

A mi novia Ana, por su amor y comprensión constante.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

SOP single-objective optimization problem

MOP multi objective optimization problem

MOO multi objective optimization

RP reference point

RPP reference point problems

RPMOP reference point multi objective optimization problem

RPMOEA reference point multi objective evolutionary algorithm

DM decision maker

MOEA multi objective evolutionary algorithm

DS directed search method

DDS discrete directed search method

1 | Introduction

Many real world problems in fields like economics, engineering, chemistry, and biology, to name a few, can be modelled by a function or a set functions which take as input the features of a product of interest (that can be controlled by a human decision maker or short DM) and give as output the quality of the solution they represent.

An example of a problem with one objective (a *single objective problem* or SOP) is the design of a car where manufacturers usually have the goal of producing a vehicle which is safe (crash worthiness). In this case, DMs can decide which materials and how much of them to take for the structure of the car and windshields, the shape of each component of the car and whether to include complementary safety devices such as airbags and seatbelts. These are called the input parameters. On the other hand, for each configuration of the input parameters, an associated value can serve as an indicator of how good the configuration is in terms of the goal (called objective function). Input parameters, such as the quantity of materials used in the car, oftentimes need to be restricted in certain ways to comply with the finite resources available for the problem in the real world. Problems of this kind are called *constrained problems*. If there are no restrictions on the inputs, the problem is called *unconstrained*.

However, when a model has more than one objective (*multi objective optimization problems* or MOPs) the solution is harder to reach. There are two main reasons for this. The first big problem is that the concept of what solutions are better is not

straightforward since the solution domain can no longer be compared nor ordered. The second big problem is that objectives are usually in conflict, meaning that there exists a whole set representing different trade-offs between the objectives. In order to illustrate this, reconsider the problem of producing a car stated before where DMs are interested not only in safety but also want the vehicle to be cheap. Now consider that for some given input parameters the following three hypothetical vehicles are produced: (i) very cheap and very unsafe (ii) very expensive and very safe and (iii) very expensive and very unsafe. The first issue is well represented when considering (i) and (ii). One solution can not be considered to be better against the other one since ideally each objective (cost and safety) is equally important. Nevertheless, (iii) can be said to be worse than the first two because it is worse in each objective ((i) and (ii) are optimal and 'dominate' (iii)). A glimpse of the second issue, that there is a whole solution set instead of a single one, comes when considering again the first two solutions. Both of them represent optimal realizations of the product which are interesting to the DMs. It can be inferred that the cheapest vehicle will be the least safe one, but as the cost increases the car can become safer, hence a trade-off is inherent to the problem.

Commonly, the goal in MOPs is to find the set of non-dominated solutions that represent the trade-off between objectives, the so called Pareto front (where it is known to form $(k-1)$-dimensional objects, where k is the number of objectives involved in the problem). The increase in the number of objectives gives rise to several problems regarding algorithm designers. For instance, the number of points required to represent the Pareto front grows exponentially with the number of objectives and the dominance relation between points become weaker, to name just a few. Furthermore, DMs will certainly be overwhelmed examining the entire solution set. Even more if they can not visualize the trade-off space when it exceeds our familiar two or three dimensional spaces.

A relatively novel approach for handling MOPs that becomes useful for both DMs and algorithm designers in many objective problems (problems with more than three objectives), requires from the DM to establish reference points (RPs) or aspiration levels he/she wants for each of the objectives. One major advantage is that the MOP is now transformed into a SOP which is, in principle, easier to handle. The new goal is then to find a solution that comes closest to the aspiration level established beforehand. The original approach is meant to be an interactive process where DMs can improve their preferences iteratively based on the solutions found in the process. However, recent studies have also found that RPs or sets in objective space can be used to find the complete Pareto set, hence its recent popularity.

Currently, there are two popular research approaches in optimization, mathematical programming techniques and evolutionary or set-based algorithms. Among set-based algorithms, subdivision techniques, cell mapping techniques and multi objective evolutionary algorithms (MOEAs [1, 2, 3]) have caught the interest of many researchers. Reasons for this include that these methods allow for the approximation of the entire set of interest in a single run of the algorithm, and that they are further characterized by a great robustness and minimal requirement on the model. Evolutionary algorithms are beneficial for the numerical treatment of reference point problems (RPPs), as they can handle several RPs in a single run. On the other hand, scalarization methods transform a MOP into a SOP (e.g., [4, 5, 6, 7]) to obtain a single solution. They are characterized by its faster convergence but require certain assumptions on the model, like differentiability or uni-modality. Hence, they lack the convergence properties from MOEAs. As for reference point-specific methods proposed so far ([8, 9, 5]), there are still many loopholes that are to be studied. The most important of them is the attainability of RPs. When a RP is attainable or feasible it is most likely not optimal and there will exist a subset of the Pareto front dominating the RP (which is more interesting to the DM). Since such solutions exist, it is unclear which solution should be presented to the DM, being the closest in the Pareto front the one commonly chosen. A recent method which can follow a given

direction in objective space, is the Directed Search (DS) Method ([10]). DS is well suited for handling RPPs because, plain and simple, the direction to follow is always towards the RP. Still, it suffers from the drawbacks mentioned above. A third, not so common approach combines both mathematical and set based techniques to create new methods which in the ideal case do not inherit the disadvantages but only the advantages of their predecessors. These hybrid algorithms are called memetic algorithms. At present, there exist some work related to memetic algorithms for general SOPs and MOPs ([11, 12, 13, 14, 15, 16]) that have shown in practice the advantages of both approaches over its standalone original counterparts. However, there is none specifically for RPPs.

1.1 Motivation

In contrast to a posteriori methods (where DMs are presented in the whole solution set and they are expected to inspect it thoroughly), a priori or interactive methods (such as the RP approach) can be put into practice to find regions of interest for them beforehand. This is based on the assumption that DMs have in many cases a good appraisal of the solutions they want for the given model (which is translated into aspiration levels/RPs). Moreover, although the use of RPs in the absence of a DM is a complicated task (being a correct positioning of RPs the most challenging one), this approach has demonstrated to be advantageous when dealing with many objective problems (see, for example, [17, 18]) The current trends of research in reference point optimization suffer from particular disadvantages which can be overcame with a hybrid variant of two base algorithms: A MOEA to explore the global fitness landscape and a local search technique to guarantee (locally) optimal solutions.

1.2 The Problem

The current approaches for the treatment of RPPs have major disadvantages. To name a few of the two classes of algorithms:

- Mathematical Programming Techniques
  - Commonly, for each RP provided by the DM, one RPP has to be solved.
  - They are easily trapped in locally optimal regions.
  - Only one solution is found for each RP.
- Evolutionary algorithms
  - Dominance selection schemes are not efficient.
  - Convergence towards RPs is slow.
  - Relatively higher computational effort is required.

In addition, there does not exist currently a meaningful quality indicator for measuring the output of RP based algorithms.

1.3 General and Particular Aims

General Aim

To develop a RP mathematical programming algorithm and hybridize it with an existing MOEA to create a novel memetic algorithm.

Particular Aims

Specifically, we achieved the following goals:

- To propose a RP version of the DS called RDS.
- To implement a gradient free realization of the method.
- To propose a memetic strategy with the gradient based and gradient free RDS as local search technique for unconstrained problems.
- To adapt the proposed algorithm and memetic strategy for constrained problems.
- To evaluate the performance of the proposed algorithm against state-of-the-art MOEA for both unconstrained and constrained problems.
- To solve a real world problem with the resulting method.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows: in Chapter 2, we state the background required for the understanding of this work as well as the related work in RP optimization. In Chapter 3, we present the DS for the treatment of reference point problem (RPPs) called RDS. In Chapter 4, we discuss how to integrate RDS into a specific MOEA, namely RNSGA-II, leading to a new memetic strategy. In Chapter 5, we present some numerical results of RDS and RDS-RNSGA-II on some widely used benchmark models and real world problems. Finally, we conclude in Chapter 6 and give possible paths for future work.

2 | Background

This chapter introduces to the basic optimization theory needed to understand the present work. The first section is dedicated to basic notions and formal definitions of SOPs and MOPs. The concept of DM is introduced in MOP theory, together with a classification of methods according to the moment of inclusion of the DMs preferences in the optimization process. A general overview of the main type of algorithms used to solve optimization problems is presented in Section 2.2. Special attention is put to the DS Method since the proposed hybrid algorithm is based upon it. Section 2.3 introduces concepts related to RPs and two state-of-the-art MOEAs. Performance indicators for MOPs which will be needed for discussion are presented in Section 2.4.

2.1 Theoretical Background

In this section we present the insights to identify and understand the components of an optimization problem starting with the theory of SOPs and MOPs and reference point problems (RPPs). The understanding of SOP theory is crucial since scalarization methods (methods which transform a MOP into a succession of simpler SOPs) come naturally in the scope of reference point optimization.

2.1.1 Single-Objective Optimization

Definition 1 (Single Objective Optimization Problem)**.** *A continuous single objective optimization problem (SOP) is defined as*

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{2.1}$$

$$s.t \quad g_i(\mathbf{x}) \leq 0 \quad i = 1,\ldots,p$$

$$h_j(\mathbf{x}) = 0 \quad j = 1,\ldots,q,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ *is called the objective function,* $g_i(x) : \mathbb{R}^n \to \mathbb{R}$*, and* $h_i(x) : \mathbb{R}^n \to \mathbb{R}$ *are the inequality and equality constraints, respectively. The restriction set is denoted by*

$$Q = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0 \text{ and } h_j(\mathbf{x}) = 0 \quad i = 1,\ldots,p \quad j = 1,\ldots,p\} \subset \mathbb{R}^n. \tag{2.2}$$

Solutions in Q *are called feasible as they comply with the restrictions, otherwise, they are called infeasible.*

Without loss of generality, we will aim in this study for minimization problems. An optimal solution of (2.1) is called a *minimum*. Depending on the scope of the minimum it is either a local minimum or a global minimum. As the name suggests, the overall solution of (2.1) is the global minimum, however, global minimizers do not have to be unique.

Definition 2 (Minimum of SOP)**.** *Let* x *be a point in* Q*,*

- x *is called a* local *minimum of* (2.1) *iff*

$$f(x) \leq f(y) \qquad \forall y \in N(x) \cap Q,$$

  where $N(x)$ *is a neighborhood of* x*.*

- *x is called a* global *minimum of* (2.1) *iff*

$$f(x) \leq f(y) \qquad \forall y \in Q.$$

In addition, a global minimum is also a local minimum since for any neighborhood the statement in (2) is fulfilled.

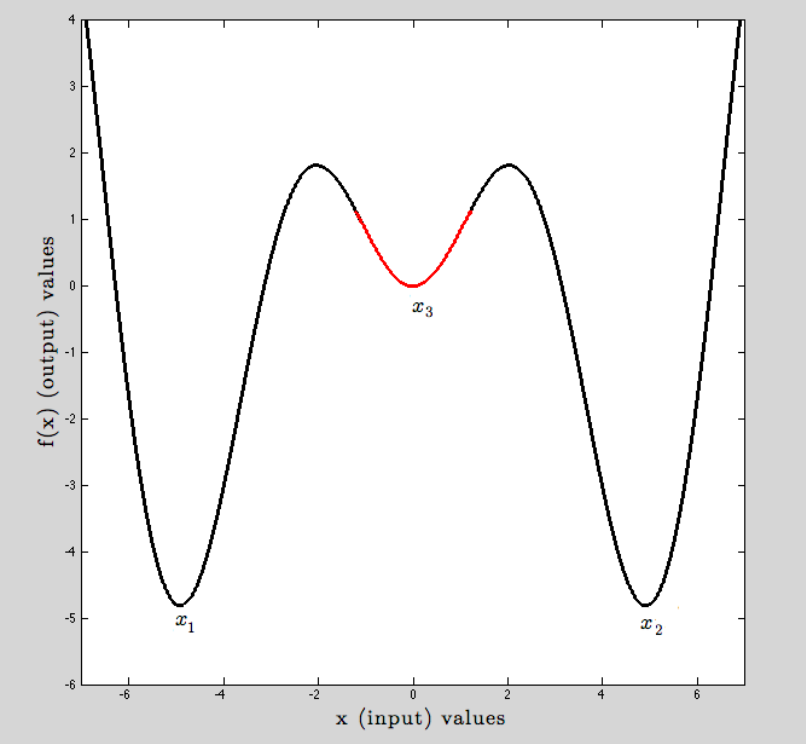


Figure 2.1: A SOP with one input value. The local solution x_3 is optimal within a chosen neighborhood (light area). Solutions x_2 and x_3 are global minimizers with the same function value.

It is impractical and sometimes impossible to test each feasible solution $x \in Q$ in order to find the minimum. Hence an intelligent search algorithm is needed. Any such algorithm needs to guarantee its solution's optimality. The means to do so are the so-called optimality conditions presented in the next pages. We will start by introducing the following definitions.

Definition 3 (Gradient)**.** *Let f be differentiable at $x \in Q$. The* derivative *or* gradient *of f at the point x is defined as*

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n} \right)^T \in \mathbb{R}^n.$$

Definition 4 (Hessian matrix)**.** *The second derivative of f at the point x is called the Hessian matrix,*

$$Hf(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

.

Definition 5 (Positive definite matrix)**.** *A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called positive definite if*

$$x^T A x > 0 \qquad \forall x \in \mathbb{R}^n \setminus \{0\}.$$

The *first order (necessary) conditions* for x^* to be a local solution of (2.1) are:

1. f has to be differentiable at x^*.

2. The gradient at x^*, $\nabla f(x^*)$, has to be equal to zero.

Points satisfying the previous conditions are not necessarily optimal, but certainly candidates. The second order (sufficient) conditions for x^* to be a locally optimal solution are

1. f has to be twice continuously differentiable at the point x^*.

2. The Hessian at the point x^*, $Hf(x^*)$, has to be positive definite.

2.1.2 Multi Objective Optimization

As mentioned in the previous chapter, real world optimization problems can involve several objectives which are in conflict with each other. We will now introduce the formalities of multi objective optimization and will then turn our attention to the particular kind of problems studied in this thesis.

Definition 6 (Multi Objective Optimization Problem)**.** *A continuous multi objective optimization problem (MOP) can be written as follows:*

$$\min_{\mathbf{x}\in Q}\{F(\mathbf{x})\}. \tag{2.3}$$

Analogously to the SOP formulation given by (2.1), $Q \subseteq \mathbb{R}^n$ is the feasible region which is a subset of the decision variables space. The function F is now defined as the vector of objectives functions:

$$\begin{aligned} F(x) &: Q \to \mathbb{R}^k \\ F(x) &= (f_1(x), \ldots, f_k(x))^T, \end{aligned} \tag{2.4}$$

where $k \geq 2$.

In contrast to a SOP a MOP has k output values. This new feature calls for a way of comparing solutions. The most commonly used definition for this purpose is the so called Pareto dominance (proposed in 1896 by Vilfredo Pareto, see [19]).

Definition 7 (Pareto dominance)**.** *(a) Let $\mathbf{v}$, $\mathbf{w} \in \mathbb{R}^k$. Then the vector $\mathbf{v}$ is less than $\mathbf{w}$ (denoted by $\mathbf{v} <_p \mathbf{w}$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$. The relation $\leq_p$ is defined analogously.*

(b) A vector $\mathbf{y} \in \mathbb{R}^n$ is dominated by a vector $\mathbf{x} \in \mathbb{R}^n$, denoted by $(\mathbf{x} \prec \mathbf{y})$, with respect to (2.3) if

$$F(\mathbf{x}) \leq_p F(\mathbf{y}) \text{ and } F(\mathbf{x}) \neq F(\mathbf{y}),$$

else $\mathbf{x}$ is called non-dominated by $\mathbf{y}$.

Pareto dominance is commonly used to define optimality with respect to a MOP. We can distinguish between different kinds of optimality:

Definition 8 (Pareto optimality)**.** *Let* $\mathbf{x} \in Q$*.*

(a) $\mathbf{x}$ *is (Pareto) optimal or a Pareto point if there exists no* $\mathbf{y} \in Q$ *which dominates* $\mathbf{x}$*.*

(b) $\mathbf{x}$ *is locally (Pareto) optimal if there exists no* $\mathbf{y} \in N(x) \cap Q$ *which dominates* $\mathbf{x}$*, where* $N(x)$ *is a neighborhood of* x*.*

(c) $\mathbf{x}$ *is weakly (Pareto) optimal if there exists no* $\mathbf{y} \in Q$ *s.t.* $F(y) <_p F(x)$*.*

Now we can explicitly state the set of optimal solutions of (2.3):

Definition 9 (Pareto Set)**.** *The set of Pareto points is called the* ***Pareto set*** *denoted by* P_Q*.*

Definition 10 (Pareto Front)**.** *The corresponding image in objective space of* P_Q *is called the* ***Pareto front*** *denoted by* $P_F = F(P_Q)$*.*

Under certain assumptions on the objective functions, we can assume that the Pareto front is a manifold of dimension $k - 1$ (for a thourough discussion of this topic, see [20]). Commonly, the ultimate goal in MOO is to find a good finite size representation of the Pareto set. This is. however, not always the case, as we will introduce later.

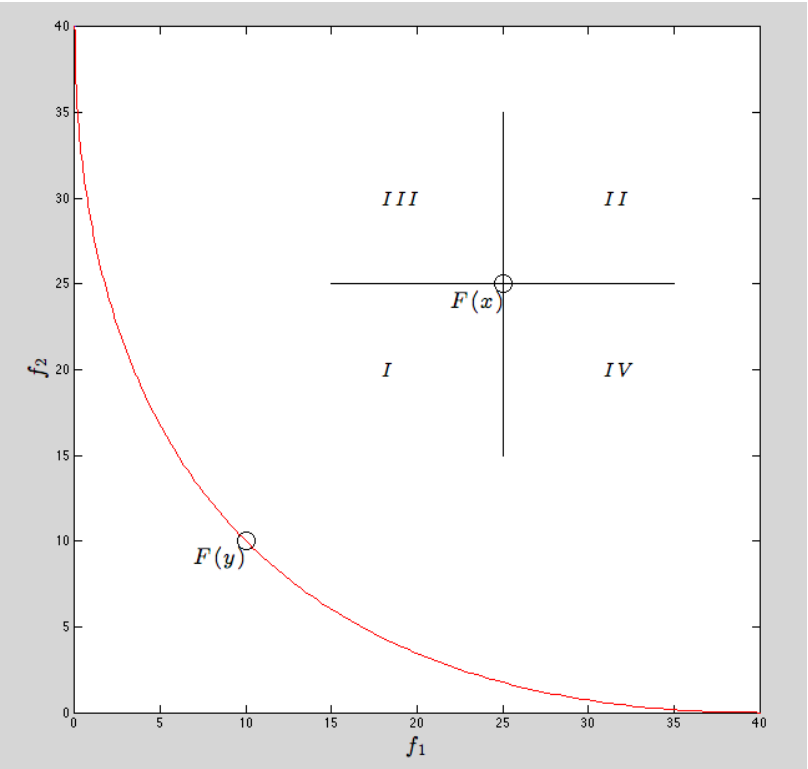


Figure 2.2: Objective space of a MOP with two objectives. Solutions whose components are in quadrant I dominate x. Solutions whose components are in quadrant II are dominated by x. Solutions whose components are in any of the two remaining quadrants are called non-dominated with respect to x. Solution y belongs to the Pareto set.

Analog to the SOP theory presented above, we need to stipulate optimality conditions in order to guarantee the (Pareto) optimality of a candidate solution. We will start with the definition of the derivative of the MOP, namely, the Jacobian matrix:

$$J(\mathbf{x}) = \frac{\partial F}{\partial x}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial f_k}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_k}{\partial x_n}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \nabla f_1(\mathbf{x})^T \\ \vdots \\ \nabla f_k(\mathbf{x})^T \end{pmatrix} \in \mathbb{R}^{k \times n}, \qquad (2.5)$$

where $\nabla f_i(\mathbf{x})$ denotes the gradient of objective f_i as in (3).

The famous theorem by Kuhn and Tucker [21], which is also credited to Karush

[22], gives a necessary condition for optimality.

Theorem 1. *Let* $\mathbf{x}^*$ *be a Pareto point and* $f_i \quad i = 1, \ldots, k$, *be continuously differentiable.*

1. *Let the MOP be unconstrained, then there exists* $\alpha \in \mathbb{R}^k$ *:* $\alpha_i \geq 0$, $i = 1, \ldots, k$ *and* $\sum_{i=1}^{k} \alpha_i = 1$ *s.t.*

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(\mathbf{x}^*) = 0. \tag{2.6}$$

2. *If the MOP is defined by* m *equality constraints such that each* h_i *is continuously differentiable, then there exists* $\alpha \in \mathbb{R}^k$ *and* $\lambda \in \mathbb{R}^m$*:* $\alpha_i \geq 0$, $i = 1, \ldots, k \quad \sum_{i=1}^{k} \alpha_i = 1$ *s.t.*

$$\begin{aligned} \sum_{i=1}^{k} \alpha_i \nabla f_i(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j \nabla h_j(\mathbf{x}^*) = 0 \\ s.t. \quad h_i(x^*) = 0 \quad i = 1, \ldots, m. \end{aligned} \tag{2.7}$$

3. *If the MOP is defined by* p *inequality constraints such that each* g_i *is continuously differentiable, then there exists* $\alpha \in \mathbb{R}^k$ *and* $\mu \in \mathbb{R}^p$*:* $\alpha_i \geq 0$, $i = 1, \ldots, k \quad \sum_{i=1}^{k} \alpha_i = 1 \quad \mu \geq 0$ *s.t.*

$$\begin{aligned} \sum_{i=1}^{k} \alpha_i \nabla f_i(\mathbf{x}^*) + \sum_{j=1}^{p} \mu_j \nabla g_j(\mathbf{x}^*) = 0 \\ s.t. \quad g_i(x^*) \leq 0 \\ \mu_i g_j(x^*) = 0 \quad i = 1, \ldots, m. \end{aligned} \tag{2.8}$$

Points satisfying the above equations are called Karush-Kuhn-Tucker (KKT) points.

2.2 Multi Objective Optimization Algorithms

A consequence of the fact that the solution of the MOP is given by an entire set of optimal solutions is that the decision of which single solution to actually implement in

the real world is out of the scope of the optimization problem solvers, be it an analyst or an algorithm. The person or persons in charge of making this decision is called DM. Any method for solving MOPs can be classified depending on the information available from the DM in time. There are many ways DMs can provide information of their preferences such as utility functions or aspiration levels (equivalent to a RP). We will present the classification of methods proposed in [5], which has four major classes:

1. *No preference.* A single solution from the Pareto front is given as output to the DM. The DM is not present in any time of the process.

2. *A priori.* In this kind of methods, the DM gives information of his/her preferences to the problem solvers beforehand. The optimization process then is focused on satisfying the goals of the DM.

3. *A posteriori.* A posteriori methods do not require any information of the DM during the optimization process. Commonly, all the non-dominated solutions of the MOP are computed and presented to the DM, who has the task of analyze and choose a pertinent solution.

4. *Interactive.* In this type of methods, the optimization process is done along with the DM. It can start as an a priori method. As a first solution is found and presented to the DM, he/she will gain knowldege and will refine his or her preferences in order to find a new solution. This iterative process will be repeated until the DM is completely satisfied.

Aside from the classification above, there exists yet another division of the algorithms based on the approach they use to solve MOPs (we mean by solving to acquire a good approximation of the Pareto set): *Mathematical Programming Techniques* which commonly generate a single solution iterated over time in a suitable way, and *Set Oriented Methods* which rely on a 'set' of solutions that solve the problem collectively. Methods of the first kind are well known for using mathematical

properties of the problem to reach the solution. Methods of the second kind have been widely studied for the past years because of their outstanding performance and minimal requirements on the model. We will now present and discuss the most widely used ones of both classes.

In the remainder of this section, we will focus on a posteriori methods and will present only one interactive approach. A posteriori methods are the most widely used ones since they give a general overview of all the possible optimal realizations of the problem.

2.2.1 Mathematical Programming Techniques

Most mathematical programming methods for the treatment of MOPs are based on a reformulation of the original problem into several SOPs. Methods of this kind are called scalarization methods. This methods exchange the original MOP by a sequence of SOPs which are solved by a standard optimization technique for SOPs ([23]). The advantage of using a mathematical programming algorithms is that (local) convergence towards a minimum can be guaranteed under certain assumptions (usually smoothness, i.e., first or even second derivative of the model is required). The main disadvantages are that the minimum found can be of local nature and/or that the assumptions may not always be satisfied. A reason for this is that the derivatives are not always available and when they are approximated by some means, the computational cost might be intolerable.

Weighted Sum Method

The weighted sum (WS) method or simply weighting method [24, 5] associates a weighting coefficient to each objective function to form a new SOP:

$$\min_{\mathbf{x}\in Q} \sum_{i=1}^{k} \omega_i f_i(\mathbf{x}). \tag{2.9}$$

It is common practice to set ω_i as convex weight, i.e. $\omega_i \geq 0$ and $\sum_{i=1}^{k} \omega_i = 1$. As an a priori method, the weighting coefficients might not be easy to set for the DM nor even hold a physical meaning for him/her, rendering it useless in certain cases. As an a posteriori approach, (2.9) must be solved many times with different weighting coefficients to find the Pareto set. However, for all it is known, the method can only find the solutions on the entire Pareto front where it is convex ([5]). In addition, the solution found may be weakly Pareto optimal.

Weighted Metrics Method

The weighted metrics (WM) method, also called compromise programming, was first presented by Zeleny in 1973 ([25]). This method was one of the first ones which used RPs in MOO. However, WM is not an a priori method, i.e., the RP used is not set by the DM. The RP to be computed is called the ideal RP (denoted by Z^*) whose components are the individual minimizers of each objective. The WM method tackles scalarization via a distance metric, for instance the $p-norm$ (compare to Figure 2.3):

Definition 11 (p-norm)**.** *Let $y \in \mathbb{R}^n$. The p-norm in $\mathbb{R}^n$ is defined as:*

$$||y||_p = (\sum_{i}^{n} |y_i|^p)^{\frac{1}{p}}$$
$$1 \leq p < \infty.$$

For the special case of $p = \infty$, the norm is called the maximum or Tchebycheff norm:

$$||y||_\infty = \max_{i=1,\ldots,n} |y_i|.$$

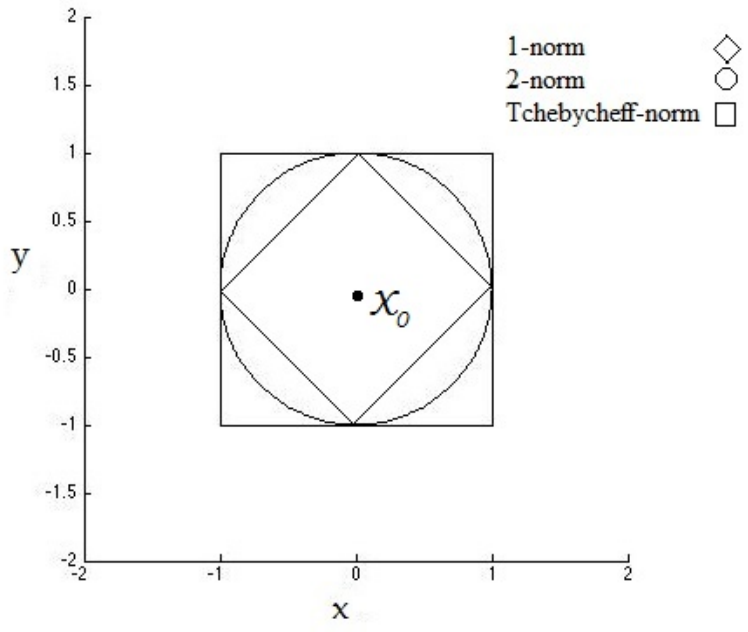


Figure 2.3: The p-norm equal to one in a two dimensional space for different values of p.

Definition 12 (Weighted p-norm)**.** *Let $W \in \mathbb{R}^{n\times n}$ be a diagonal matrix with positive entries $\omega_i, i = 1, \ldots, n$, and $y \in \mathbb{R}^n$. The weighted p-norm in $\mathbb{R}^n$ is defined as:*

$$||Wy||_p = (\sum_i^n \omega_i |y_i|^p)^{\frac{1}{p}}$$
$$1 \leq p < \infty.$$

The weighted Tchebycheff norm is defined as:

$$||Wy||_\infty = \omega_i |y_i|$$
$$1 \leq p \leq \infty.$$

Using the above definitions, WM reads as follows:

$$\min_{\mathbf{x}\in Q} ||Wy||. \tag{2.10}$$

For $p = \infty$, the problem is called the weighted Tchebycheff problem:

$$\min_{\mathbf{x} \in Q} ||Wy||_\infty. \tag{2.11}$$

Equation (2.11) carries the disadvantage of being nondifferentiable. Instead of it, one could solve:

$$\begin{aligned} \min_{(x,\alpha)\in\mathbb{R}^{n+1}} \quad & \alpha \\ \text{s.t.} \quad & \alpha \geq \omega_i(f_i(\mathbf{x}) - Z_i^*) \quad i = 1, ..., k. \end{aligned} \tag{2.12}$$

In this case, the variable $\alpha \in \mathbb{R}$, has to be optimized as well.

One interesting feature of the method is that the solution depends completely on the choice of p as can be seen in Figure 2.4 on a hypothetical example.

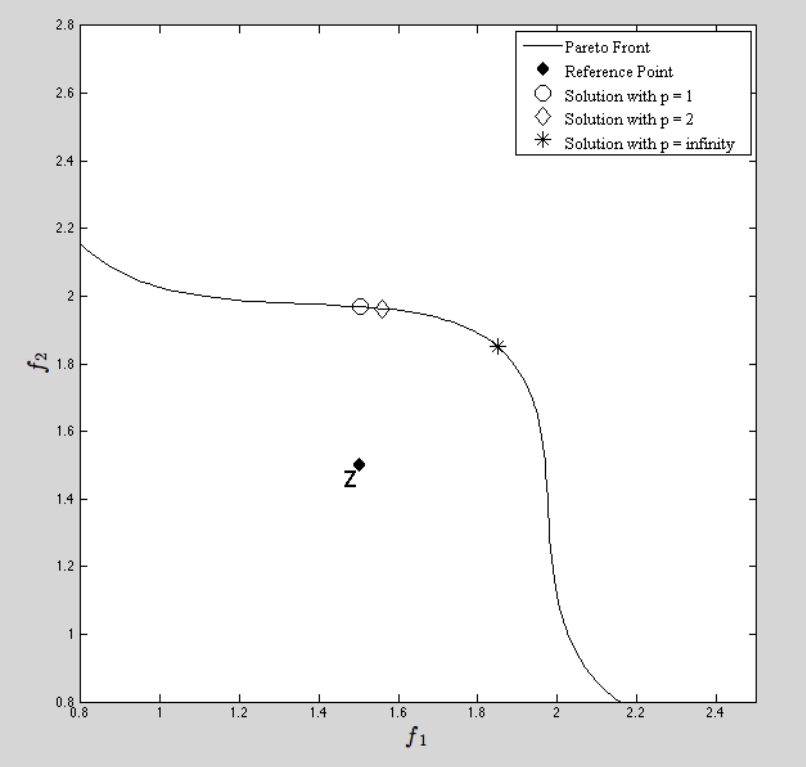


Figure 2.4: The weighted metrics method with three different values of p.

The following theorems show important properties regarding the choice of p.

Theorem 2. *The solution of* (2.10) *is Pareto optimal if either:*

- *the solution is unique.*
- *all the weighting coefficients are positive.*

Theorem 3. *The solution of* (2.11) *is (weakly) Pareto optimal if* $\omega \in \mathbb{R}^k_+$.

Theorem 4. *Let* $\mathbf{x}^* \in S$ *be Pareto optimal. Then there exists* $\omega \in \mathbb{R}^k_+$ *such that* $\mathbf{x}^*$ *is a solution of* (2.11).

The proofs can be found in [5]. The first theorem implies that the solution to (2.10) is a Pareto point, but this does not imply that all Pareto points can be found by it. When considering $p = \infty$, all Pareto solutions can be found, however, weakly Pareto solutions might be found during the process too. Hence, additional computation is needed to identify the weak solutions.

Directed Search Method

Schütze et al. proposed in 2011 an algorithm for unconstrained MOPs called the Directed Search (DS) Method [10]. The idea of DS is to find a direction $\nu \in \mathbb{R}^n$ that from a point $x_0 \in \mathbb{R}^n$ can steer the search to a certain direction $d \in \mathbb{R}^k$ defined in objective space. Formally stated, this is:

$$\lim_{t \to 0} \frac{f_i(x_0 + t\nu) - f_i(x_0)}{t} = d_i, \quad i = 1, \ldots, k. \tag{2.13}$$

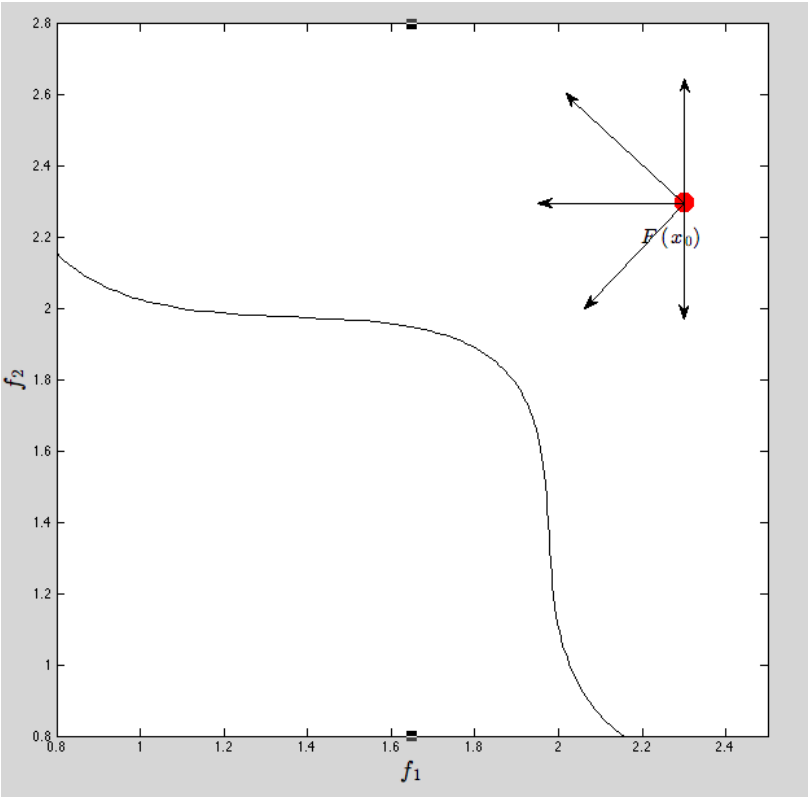


Figure 2.5: A set of possible directions to steer in objective space starting from x_0.

Stated in matrix vector notation, Equation (2.13) is equivalent to:

$$J(x_0)\nu = d, \tag{2.14}$$

where $J(x)$ denotes the Jacobian at x_0. Notice that (2.14) only has a unique solution iff $J(x)$ is square and has full rank. If the Jacobian is not square (which is usually the case for a MOP since typically the number of objectives is much less than the number of variables) the represented linear system of equations is under-determined and has infinitely many solutions. Among the set of solutions, one suggesting choice is to use the pseudo-inverse $J(x_0)^+$ since it has the smallest Euclidean norm among all possible solutions. In consequence, one can expect the greatest improvement in direction d with a fixed and small step size t.

Regarding the choice of d, the authors make use of a direction tailored to find

dominating solutions, that is, each new solution found in an iteration of DS dominates the previous one. For this, a descent direction $d \leq_p 0, d \neq 0 \in \mathbb{R}^k$ is used. However, since for a step size the corresponding movement in direction ν can lead the new iterate away from the desired direction d, a corrector step is used to bring the solutions back to the desired line $F(x_0) + t^* d$. This process is equivalent to the numerical solution of the following initial value problem:

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x}(t) &= \nu_+(x(t)), \quad t > 0. \end{aligned} \tag{2.15}$$

DS can only keep shooting in direction d until a boundary (and hopefully also Pareto) point is reached. Such points are characterized by having a Jacobian matrix which is rank deficient or equivalently that the gradients are linear dependent. That is, for a boundary point x_b there exist constants $c_1, \ldots, c_k$ with not all c_i's $= 0$ such that:

$$c_1 \nabla f_1(x_b) + \cdots + c_k \nabla f_k(x_b) = 0. \tag{2.16}$$

A boundary point can be tested for optimality by solving the zero finding problem associated to (2.6) in page 14.

This phase of finding a Pareto point is called the *descent phase* of DS. A stopping criterion for this phase is to check for given tolerance on the condition number of the Jacobian: $\kappa(J(x)) < tol$.

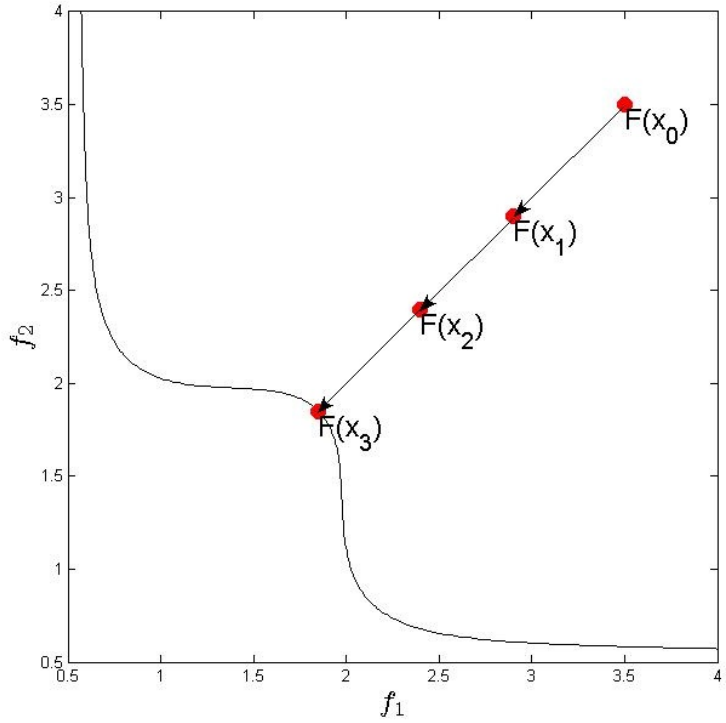


Figure 2.6: Descent phase of DS using direction $d = (-1, -1)^T$. Descent phase stops when a Pareto point is reached.

In order to steer the search along P_F from a given Pareto point x, a new direction d is computed to follow the boundary of the MOP. This second phase is called *continuation phase* of DS due to the continuation-like method used. A continuation method for a curve is basically composed of a predictor step and a corrector step. The predictor step follows the curve in a given direction. Meanwhile, the corrector step is used to correct back to the curve due to the possible errors made in the predictor step.

Suppose we are given a Pareto point x_0. Recall that such points fulfill,

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(\mathbf{x}^*) = 0,$$

where $\alpha_i \geq 0$, $i = 1, \ldots, k$, and $\sum_{i=1}^{k} \alpha_i = 1$ implies that α is orthogonal to the linearized Pareto front at $F(x^*)$ [20]. Hence, a search orthogonal to α will follow the (linearized) Pareto front. This is done as follows. Compute the QR decomposition of α: $\alpha = QR$, where $Q = (q_1, \ldots, q_k) \in \mathbb{R}^{k \times k}$ is an orthogonal matrix and $R = (r_{11}, 0, \ldots, 0)^T \in \mathbb{R}^{k \times 1}$ such that $r_{11} \neq 0$. Since $\alpha = r_{11} q_1$, the column vectors $q_2, \ldots, q_k$ form an orthonormal basis of the hyperplane orthogonal to α. Hence, a search in direction q_i, $i = 2, \ldots, k$, will follow partially the Pareto front. For the special case of $k = 2$, the search in direction q_2 will cover the entire Pareto front.

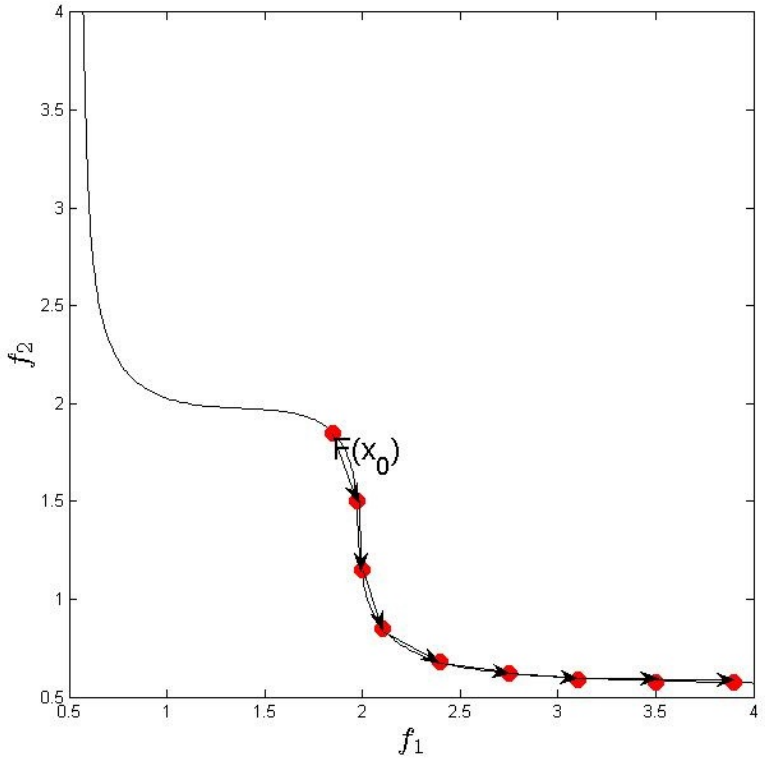


Figure 2.7: Continuation phase in a given direction along the Pareto front.

Discrete Directed Search

In 2012, Lara et al. presented a gradient free version of the DS method called the Discrete Directed Search [26] or DDS. The method assumes a solution $x \in Q$ is given

along with r search directions $\nu_i \in \mathbb{R}^n, i = 1, \ldots, r$. Then, the matrix $\mathcal{F} \in \mathbb{R}^{k \times r}$ is defined as:

$$\mathcal{F} := (\langle \nabla f_i(x), \nu_j \rangle), i = 1, \ldots, k; j = 1, \ldots, r, \tag{2.17}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product. Hence, each entry m_{ij} of $\mathcal{F}$ is equivalent to the directional derivative of the i-th objective in direction ν_j (i.e. $\nabla_{\nu_j} f_i(x)$). The following result is crucial for the DDS:

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

$$J(x)\nu = \mathcal{F}\lambda. \tag{2.18}$$

Proof 1. *In the original work [26] the proof can be found.*

Since we are only looking for a search direction ν satisfying (2.14) we can exchange the problem by first solving

$$\mathcal{F}\lambda = d, \tag{2.19}$$

and then set

$$\nu := \sum_{i=1}^{r} \lambda_i \nu_i. \tag{2.20}$$

In order to avoid computing the entries (derivatives) of $\mathcal{F}$ at some point x_0, one can try to approximate them by finite differences assuming r points $x_i, i = 1, \ldots, r$, in the neighborhood of x_0 together with their corresponding function values $F(x_i)$. Define

$$\begin{aligned} t_j &= ||x_j - x_0||_2, \\ \nu_j &= \frac{x_j - x_0}{t_j}, j = 1, \ldots, r, \end{aligned} \tag{2.21}$$

and set m_{ij} as

$$\begin{aligned} m_{ij} = \langle \nabla f_i(x), \nu_j \rangle &= \lim_{t \to 0} \frac{f_i(x_0 + t\nu_j) - f_i(x_0)}{t} \\ &\approx \frac{f_i(x_0 + t\nu_j) - f_i(x_0)}{||x_j - x_0||_2} \quad i = 1, \ldots, k, j = 1, \ldots, r. \end{aligned} \tag{2.22}$$

Finally compute the direction ν as

$$\nu^{(r)} = \sum_{i=1}^{r} \lambda_i \nu_i, \quad \text{where} \quad \lambda = \mathcal{F}^+(x)d. \tag{2.23}$$

The Reference Point Method Interactive Approach

The use of RPs was introduced and first defended over scalarization methods (such as the WS method mentioned above) by Wierzbicki in 1979 [9]. His work holds the point of view that the optimization process has to be carried out along with the DM. Moreover, Wierzbicki argues that goals are easy to understand and to be established by DMs. Consequently, he proposes an interactive approach whose purpose is to satisfy aspiration levels regardless of the properties of the MOP. The algorithm of Wierzbicki is practical although very simple. First, a RP is asked from the DM. As the solution of this problem is found, an extended set of RPs similar to the first provided is created and all the solutions are presented to the DM. The DM then analyzes the solutions and refines his/her preferences until he/she is satisfied. The basic framework of the algorithm is as follows:

- Let $Z \in \mathbb{R}^k$ be a RP provided by the DM with Z^* its corresponding solution using Equation (2.10). Then, create k more RPs (and find the corresponding solution) as:

$$Z_i = Z + (Z^* - Z) \cdot \hat{e}_i, \tag{2.24}$$

  where $\hat{e}_i, \quad i = 1, \ldots, k$ are the canonical vectors in $\mathbb{R}^k$.

- Present the $k+1$ solutions to the DM. If he/she is satisfied, stop. Else, ask the DM to choose a new RP or one out of the presented ones and repeat the process.

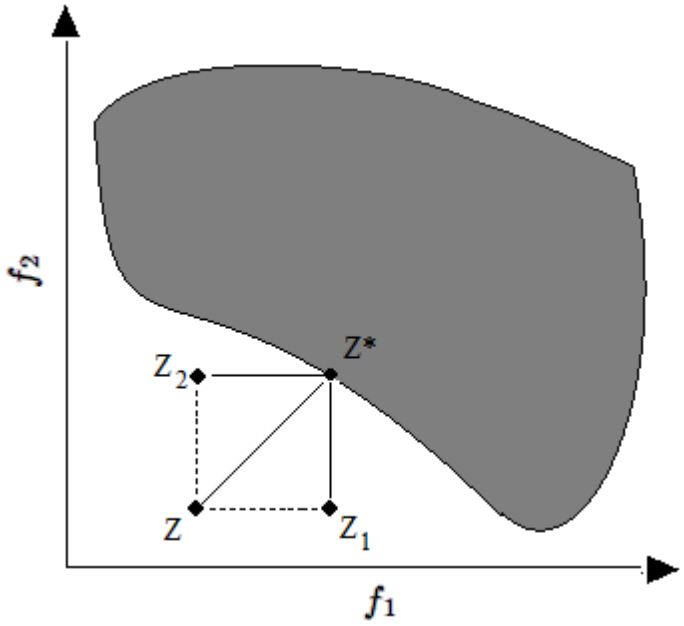


Figure 2.8: Example of an interation of the interactive approach by Wierzbicki. Two new RPs are set and their solution presented to the DM.

2.2.2 Evolutionary Algorithms

Evolutionary Algorithms (EA) approach optimization problems based on the principles of modern evolution. The evolution (optimization) process relies on the selection of individuals of a population (candidate solutions) in a generation (iteration) based on their fitness to create new, better, solutions. The fitness value represents the usefulness of the individual for solving the problem and is not necessarily associated to the function value(s). The next generation of individuals (called the offspring population) is created by combining the best-fitness individuals and mutating them. This process is then repeated taking the offspring population to create new solutions until

a stopping criterion is met.

EAs have been widely used to solve optimization problems as they have shown an extraordinary performance (see for example, [27]). Because of this, they have also been adapted to the context of MOO [3] as multi objective Evolutionary Algorithms (MOEA).

Unlike the mathematical scalarization methods listed before, MOEAs do not make assumptions on the model. In turn, they can not guarantee optimality of the solutions found. In addition, they have a slow convergence rate, as they test over and over (in a structured manner) the variable space looking for better solutions. In the following, we will present some of the most important EAs and MOEAs.

Genetic Algorithms

Genetic algorithms (GA) [28] were first proposed by John Holland in the sixties. In the original work, solutions are represented by a binary string called chromosome, where each entry of the chromosome (zero or one) is called an allele. The GA emphasizes the creation of offspring from solutions with higher fitness value via its crossover operator. A second operator, called mutation, is used to keep the population of chromosomes from stagnation. The basic algorithm of the GA is presented in Algorithm 1.

Algorithm 1 Generic Genetic Algorithm

Require: NP: population size, C_p, M_p : cross-over and mutation probability.

1: Create a random initial population P_G. Set $G \leftarrow 0$
2: Compute the fitness of individual $x_i \in P_G \quad i = 1, \ldots, NP$
3: **while** stopping criterion not met **do**
4: Select individuals for cross-over based on fitness.
5: Apply cross-over with probability C_p and create new population Q_G.
6: Apply mutation with probability M_p to individuals in Q_G.
7: Select individuals for the next population P_{G+1}. Set $G \leftarrow G + 1$.
8: **end while**

It is a common practice in GA to select only the best N_P individuals (elitism) from the parent P and offspring Q combined population. Several different versions of GAs have been suggested which vary in terms of the representation, selection mechanism, crossover and mutation operator.

Differential Evolution

Differential Evolution is a relatively recent algorithm dating from 1997 designed by Rainer Storn and Kenneth Price (DE,[27]) as a global optimizer for continuous SOPs. Unlike most EAs, DE does not have a biological inspiration. Nevertheless, it has proven efficacy and robustness in solving several benchmarks (e.g. [27]). Many versions of DE have been proposed, mainly classified as *DE/x/y/z*. Under this notation, *DE* means Differential Evolution, x indicates the strategy used to select individual included in the mutation operator, y is an integer representing the number of pairs of solutions chosen for mutation and z stands for the recombination strategy used. The most widely used variant is *DE/rand/1/bin*: the individuals selected for mutation are chosen at random, one pair of solutions is chosen for this purpose and a binomial recombination is used.

DE has only two specific parameters: CR and F. The former, controls the influence of the parent individual in the offspring individual and is restricted to values between 0 and 1. The latter scales the influence of the pair of solutions in the mutation, the recommended values are between 0 and 2. The main procedure is presented in Algorithm 2.

Algorithm 2 Differential Evolution rand/1/bin.

Require: NP: population size, MAX_{GEN}: maximum number of iterations, CR and F are user-defined parameters of DE.

1: Create a random initial population $x_{i,G} \quad i = 1, \ldots, NP$. Set $G \leftarrow 0$.

Evaluate $f(x_{i,G}) \quad i = 1, \ldots, NP$.
for $G = 1$ **to** $MAXGEN$ **do**
  for $i = 1$ **to** NP **do**
    Select randomly $r_1 \neq r_2 \neq r_3$:
    j_{rand} = randint(1,n).
    for $i = 1$ **to** n **do**
      if $rand_j[0, 1) < CR$ **or** $j = j_{rand}$ **then**
        $u_{i,j,G+1} = x_{r3,j,G} + F(x_{r1,j,G} - x_{r2,j,G})$
      else
        $u_{i,j,G+1} = x_{i,j,G}$
      end if
    end for
    if $f(u_{i,G+1}) \leq f(x_{i,G})$ **then**
      $x_{i,G+1} = u_{i,G+1}$
    else
      $x_{i,G+1} = x_{i,G}$
    end if
  end for
end for

NSGA-II

Deb proposed in 2002 an improved multi objective GA of his Non-dominated Sorting Genetic Algorithm [29] or NSGA-II. The algorithm is characterized by the selection mechanism based on a sorting of the population into different levels of non-dominated fronts, hence its name. In the first stage, NSGA-II creates a random population and produces an offspring population (using the Simulated Binary Crossover (SBX) [30]). Then, both populations are combined and sorted using the procedure presented in Algorithm 3.

Algorithm 3 Non-dominated sorting procedure in t-th generation.

Require: P: population

for each individual $p \in P$ **do**
  Set $S_p = 0$ and $\eta_p = 0$ ▷ S_p stores the indices of the solutions dominated by p
  Set $\eta_p = 0$ ▷ η_p is a counter of the number of solutions which dominate p
  for each $q \in P$ **do**
    if $p \prec q$ **then**
      $S_p = S_p \cup \{q\}$
    else if $q \prec p$ **then**
      $\eta_p = \eta_p + 1$
    end if
  end for
  if $\eta_p = 0$ **then**
    $p_{rank} = 1$ ▷ Non-dominated rank of p
    $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$ ▷ First front of non-dominated individuals
  end if
  $i = 1$
  while $\mathcal{F}_i \neq \emptyset$ **do**
    $Q = \emptyset$
    for each $p \in \mathcal{F}_i$ **do**
      for each $q \in S_p$ **do**
        $\eta_q = \eta_q - 1$
        if $\eta_q = 0$ **then**
          $q_{rank} = i + 1$, $Q = Q \cup \{q\}$
        end if
      end for
    end for
    $i = i + 1$, $\mathcal{F}_i = Q$
  end while
end for

Individuals are selected to the next parent population from the sorted combined population starting from the first non-dominated front until the population size is exceeded. If the last front can not be completely included but only partially, a second property called crowding distance, is associated to each individual. This distance measures how isolated a solution is from the rest of the population, being the most isolated ones preferred over crowded solutions. The proposed crowding distance procedure works only in two objectives and will not be presented here.

MOEA-D

The multi objective Evolutionary Algorithm based on Decomposition or MOEA-D [31] was presented in 2007 by Qingfu Zhang and Hui Li from the University of Essex. The main idea of MOEA-D is to solve the MOP by using scalarization methods like the WS or WM methods presented before. Since the theoretical properties of the WM method with the weighted infinity norm guarantee a (weakly) Pareto solution it has been most widely used in practice. MOEA-D considers neighbor solutions for a subproblem with weighting matrix W_i for crossover. These neighbor solutions consist of the solutions with similar weighting matrix W_j. The population is then composed of the best solution found so far for each subproblem. The generic algorithm of MOEA-D is presented in Algorithm 4.

Algorithm 4 MOEA-D generic algorithm

Require: N: number of subproblems considered, T : number of neighbors to be considered

1: Set $EA = \emptyset$ ▷ The set EA is composed of all non-dominated solution found so far
2: **for** each subproblem $i = 1, \ldots, N$ **do**
3: Compute the T closest solutions using a distance metric and store them in B_i.
4: **end for**
5: Initialize and evaluate a population with size N

Compute the ideal point so far as Z_i, where each component is the best value found so far within the population's objective values.
while stopping criterion not met **do**
  for each subproblem $i = 1, \ldots, N$ **do**
    Take two random solutions from B_i and create a new solution x for subproblem i with genetic operators.
    Apply a heuristic to x to produce a better solution x'.
    for each objective $j = 1, \ldots, k$ **do**
      if $F(x')_j < Z_j$ **then**
        Update the value of Z_j with $F(x')_j$.
      end if
    end for
    for each neighbor solution $j = 1, \ldots, T$ **do**
      if $WM(x', W_j) < WM(x_j, W_j)$ **then** ▷ $WM(\Delta, W_j)$ is the subproblem using weighting matrix W_j,x_j is the best value for subproblem $WM(\Delta, W_j)$.
        Update the value of x_j with x'.
      end if
    end for
    Remove from EA all solutions dominated by x'.
    if x' is not dominated by a member of EA **then**
      Set $EA = EA \cup \{x'\}$
    end if
  end for
end while

It is worth mentioning that MOEA-D is just a framework to solve MOPs where any heuristic could be taken within to obtain better solutions. Also, since the computation of the real ideal point is a costful operation, it is taken rather from the minimum values

found so far from the population.

2.3 Reference Point Problems

Originally, the reference point approach was presented as an interactive method in 1979 by Wierzbicki [9]. In his work, he argues about the benefits of using RPs instead of weight coefficients commonly used in scalarization methods. He gives several qualitative and theoretical reasons for using RPs. However, we will not discuss any RP interactive approach in this work. The reader is invited to read [5, 9] for a thorough investigation of this topic.

No preference and a posteriori methods which use RPs are also found in literature specially in mathematical programming techniques. They will be briefly discussed in this section. We will emphasize the study of the a priori approach, i.e, we will assume pre-defined RPs supplied by the DM.

Formally a reference point problem (RPP) is composed of a MOP along with a set of RPs. A RP, $Z \in \mathbb{R}^k$, is a point defined in objective space and serves as a pre-established goal or aspiration level.

The task in a RPP usually consists of minimizing the distance of the feasible search space and the RPs via a distance metric.

Definition 13 (Metric)**.** *A metric is a function* $D : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *such that for all* $x, y, z \in \mathbb{R}^n$*,* D *satisfies:*

$$\begin{aligned} D(x,y) &\geq 0 \\ D(x,y) = 0 \quad &\textit{iff} \quad x = y \\ D(x,y) &= D(y,x) \\ D(x,z) &\leq D(x,y) + D(y,z). \end{aligned}$$

The most commonly used metric is the p-norm as presented before in (11). For the following definitions consider a MOP of k objectives as in (2.3) and let $Z \in \mathbb{R}^k$ be a

given RP. Then, the p-norm RPP becomes:

$$\min_{\mathbf{x}\in Q} ||F(\mathbf{x}) - Z||_p. \tag{2.25}$$

Equation (2.25) has interesting features which will be studied in the following subsection.

Depending on the position the RP has in the objective space, it can be classified into special cases and will have particular properties.

Let $Z \in \mathbb{R}^k$ be a RP and $F(Q)$ be the domain of F.

Definition 14 (Ideal and Utopian Reference Point)**.** *The ideal RP Z^* is composed of the individual minimum of each objective, i.e.*

$$Z_i^* = \min_x f_i(x) \quad \forall i = 1, ..., k.$$

Z^{**} is called the utopian *RP if it is partially less than every Pareto point:*

$$Z_i^{**} = Z_i^* - \epsilon \quad \forall i = 1, ..., k.$$

Proposition 2. *The solution of (2.25) using the ideal RP is a Pareto point.*

Proof 2. *Assume the solution x^* of (2.25) is not Pareto optimal. Then x^* is dominated by at least one point. Let $x^{**} \prec x^*$, i.e. $F(x^{**}) <_p F(x^*)$. Since $Z^* <_p F(x) \quad \forall x \in Q$, $||F(\mathbf{x}^{**}) - Z^*||_p < ||F(\mathbf{x}^*) - Z^*||_p$. Hence, the solution to (2.25) is non-dominated.*

The same proof is valid when using the utopian RP.

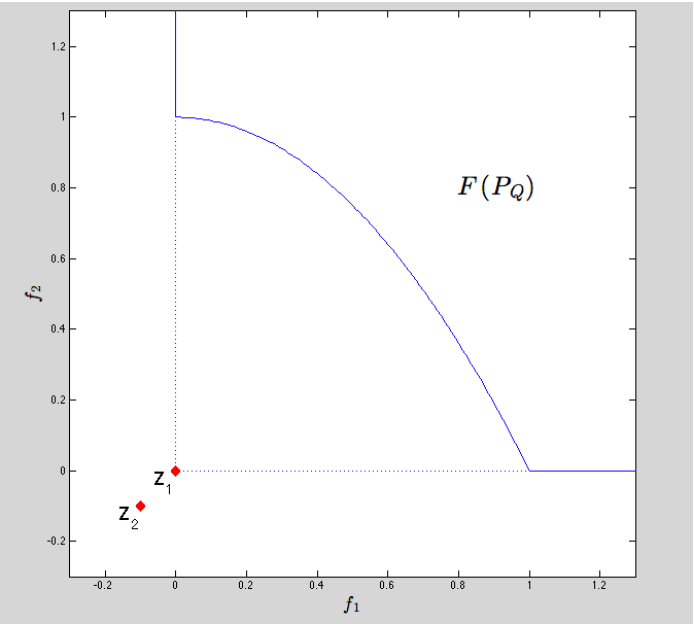


Figure 2.9: Ideal (Z_1) and utopian (Z_2) RPs of a concave MOP with two objectives. The black line indicates the Pareto front.

Definition 15 (Feasible Reference Point)**.** *Z is called* feasible,non-utopian *or* attainable *if $Z \in Q_k$. For a feasible RP the following statement is true:*

$$\exists x^* \in Q \ s.t. \ ||F(\mathbf{x}^*) - Z||_p = 0. \tag{2.26}$$

If (2.26) is not satisfied, Z is called *infeasible*. Two properties from RP's feasibility are to be noticed:

1. Non-feasibility does not necessarily imply optimality as the RP can be partially more than a point x^* yet not be contained in Q_k. That is, $\exists x^* \in Q : F(\mathbf{x}^*) <_p Z$ such that (2.26) is not satisfied.

2. A feasible RP is only optimal if it belongs to the Pareto front. In any other case, there will exist a better solution according to the dominance relation than the feasible RP. However, notice that as expressed by (2.26) for a non-optimal RP, the actual solution of the RPP is given at x^* and not at a Pareto point.

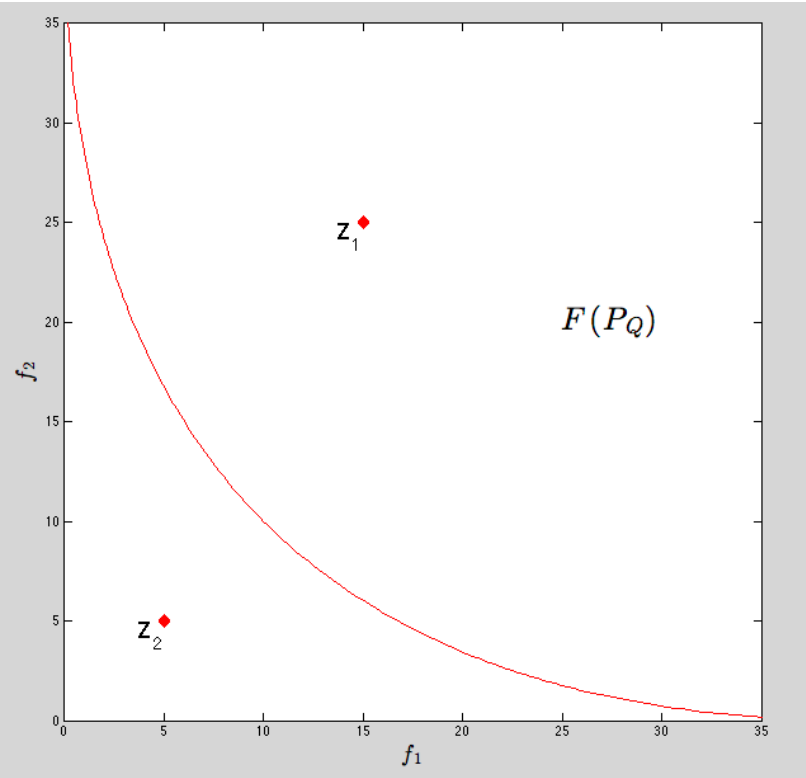


Figure 2.10: Example of a feasible RP Z_1 and an infeasible RP Z_2. For Z_1 there exists a point $x \in Q$ s.t. $F(x) = Z_1$.

2.3.1 Reference Point Based multi objective Optimization Evolutionary Algorithms

The work that exists in literature on RPs is presented in this section. Two remarkable features of RP-based MOEAs are to be noticed. Ideally, a RP-based MOEA should be able to:

- Work with several RPs in one single run.
- Given a RP for which the closest solution is not a Pareto point, it should be able to find solutions on the Pareto front close the such RP.

These algorithms usually take more than one criteria (distance towards a RP) to obtain optimal solutions since in the sense of the classical RP problem (2.25) for feasible RPs, the problem attains its optimal value at the RP and not in a point on the Pareto front.

RNSGA-II

The RNSGA-II [32] algorithm was proposed by Deb. et al in 2006. RNSGA-II is an a priori version of the popular NSGA-II algorithm presented in the previous section which can deal with user preferences in the form of RPs. The framework of the NSGA-II algorithm is kept with only one difference, namely, the crowding distance is replaced by the procedure presented in Algorithm 5.

Algorithm 5 Modified crowding distance in RNSGA-II.

Require: P: population , Z: set of RPs , ϵ: desired spread around Z_i

1: **for** each RP $Z_i \in Z$ **do**
2: Compute the distance of each member in the population to Z_i and store it in an ordered ascending list.
3: **end for**
4: **for** each $p \in P$ **do**
5: Assign the minimum rank p has in the sorted lists as the modified crowding distance.
6: **end for**
7: Cluster solutions with normalized difference in objective values of ϵ or less.
8: **for** each cluster **do**
9: Keep the assigned crowding distance of a random solution and change the crowding distance of all others to a large value.
10: **end for**

RNSGA-II was tested on the bi-objective benchmark problems ZDT1, ZDT2, ZDT3 and the three, five and ten objective DTLZ2.

RMEAD

In 2012, Mohammadi et al. proposed a RP algorithm based on decomposition called RMEAD [33]. Similarly to MOEA-D, RMEAD is an algorithm which borrows the idea of decomposition of the MOP into simpler SOP subproblems, however, the subproblems are restricted to those which give solutions close to the supplied RPs. Namely, RMEAD takes two decomposition methods: the WS and WM with Tchebycheff norm. The algorithm was tested and compared against RNSGA-II on two and three objective concave and convex problems giving better results when using the WM method. RMEAD was outperformed when using WS as decomposition since for non convex problems the formulation does not guarantee a Pareto optimal solution.

The RMEAD algorithm is presented in Algorithm 6.

Algorithm 6 RMEAD algorithm.

Require: P_s: population size , $radius$: spread of solutions around a given RP , Z: set of RPs, lb: lower bounds, ub: upper bounds

1: Initialize a population with P_s individuals within the boundaries lb and ub.
2: Compute a set of P_s initial weight vectors IW.
3: Use a heuristic to find the weight vectors giving the closest solutions to the RPs.
4: Evaluate the population using the given MOP.
5: **for** each RP $Z_i \in Z$ **do**
6: Find the closest individual and its associated weight vector BW_i in the population to Z_i.
7: Use BW_i to initialize P_s new weight vectors W_i in the neighborhood of size $radius$.
8: **end for**
9: Initialize a population with $P_s \times |Z|$ individuals.
10: Set $step = radius/P_s$.
11: **while** stop criteria not met **do**
12: Evolve the population using a given heuristic

Evaluate the population using the set of weight vectors W_i.
for each RP $Z_i \in Z$ **do**
  Set $best_i$ as the weight of the closest solution to Z_i.
  Set $worst_i$ as the weight of the farthest solution to Z_i.
  Compute an update direction as $d = best_i - worst_i$
  Update the weight vector W_i in direction d by an amount *step*.
end for
end while

2.4 Performance Indicators

All methods presented so far need to be assessed in order to obtain certain information about their outcome. The means for retrieving such information are called performance indicators. In practice, it is always desirable to attain an approximation that (a) covers the entire Pareto front uniformly (which accounts for spread) and (b) lies exactly over the Pareto front (which accounts for convergence). There exists several performance indicators for this task, nevertheless, none of them are perfect on their own. We will present in this section the ones which will be the most relevant in the context of RPPs.

In the following, we will define the output of a method, a finite approximation of the Pareto set, as an archive.

Definition 16 (Archive)**.** *An archive A of size l contains l elements in $\mathbb{R}^n$ which are mutually non dominated, i.e.*

$$\begin{aligned} A = \{a_1, \ldots, a_l\} \quad a_i \in \mathbb{R}^n \quad i = 1, \ldots, l \\ : a_i \not\prec a_j \quad \forall i = 1, \ldots, l : \forall j = 1, \ldots, l, j \neq i. \end{aligned} \tag{2.27}$$

Because of the use of distances in RPPs, we will begin this section with the definition of special metrics.

Definition 17 (Distance from a point to a set)**.** *The distance between a point $b \in \mathbb{R}^n$ and a set $A \subset \mathbb{R}^n$ is defined as:*

$$dist(b, A) := \inf_{a \in A} ||b - a||, \tag{2.28}$$

where $||\cdot||$ is a norm as in Equation (11).

Definition 18 (Semi-distance from two sets)**.** *The semi-distance between two sets A, $B \subset \mathbb{R}^n$ is defined as*

$$dist(B, A) = \sup_{b \in B} dist(b, A). \tag{2.29}$$

Note that $dist(A, B)$ is not symmetric, i.e., it does not have to hold $dist(B, A) \neq dist(A, B)$ for all sets.

Finally, the Hausdorff distance [34], is defined as:

Definition 19 (Hausdorff distance)**.** *Let A,$B \subset \mathbb{R}^n$, then the Hausdorff distance between A and B is defined as*

$$d_H(A, B) = \max(dist(A, B), dist(B, A)). \tag{2.30}$$

2.4.1 Generational Distance

The Generational Distance (GD) indicator measures the (averaged) distance from an archive A to the true Pareto front $F(P_Q)$. Veldhuizen and Lamont introduced the GD indicator in [35]:

Definition 20.

$$GD(F(A), F(P_Q)) := \frac{1}{|A|}(\sum_{i=1}^{|A|} d_{a \in F(A)}(a, F(P_Q))^2)^{1/2}. \tag{2.31}$$

The GD indicator takes the distance from each member of the archive to the closest point in the Pareto front. This property suggests that GD measures convergence to the Pareto front (a GD value of zero means all solutions of A lie on the Pareto front).

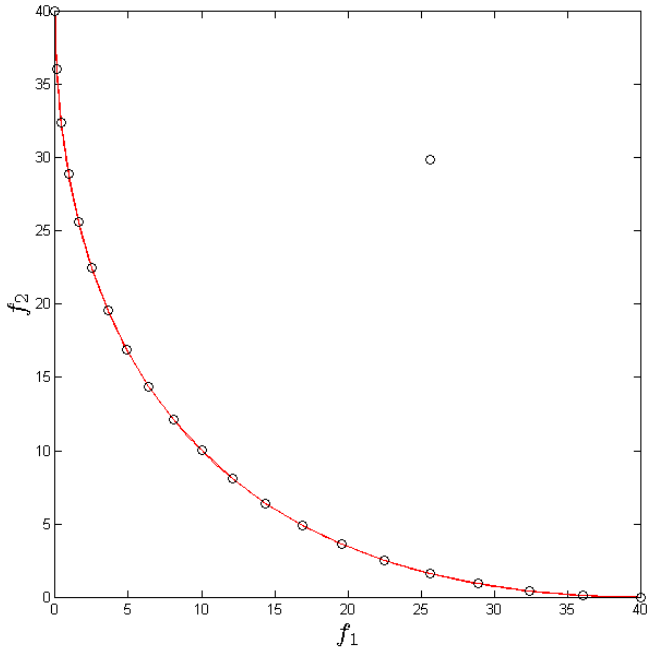


Figure 2.11: A discretization of the Pareto front (red line) is used to compute GD indicator. Single solutions have a huge impact in the GD value.

However, GD is not well suited for measuring spread of solutions. For example, an archive with a single solution which is Pareto optimal would get the best value of the GD indicator, although it doesn't cover $F(P_Q)$. In addition, requiring the true Pareto front is a big disadvantage of GD, since it is not always available in real world problems or even in academic examples.

2.4.2 Inverted Generational Distance

A variation of the GD indicator is called the Inverted Generational Distance (IGD). First presented in [36] by Cruz and Coello, IGD can be seen as complementary to GD.

Definition 21 (Inverted Generational Distance)**.** *Let* $F_P = \{y_1, \ldots, y_m\}$ *be a finite*

size discretization of the Pareto front. IGD is defined as:

$$IGD(F(A), F_P) := \frac{1}{m}(\sum_{i=1}^{|F_P|} d_{p \in F_p}(p, F(A))^2)^{1/2}. \tag{2.32}$$

Hence, the IGD value of an archive denotes 'how well' the archive covers the given discretization of the Pareto front. In this sense, IGD is a good indicator for the spread of solutions. IGD is not flawless. Picture the following scenario: let an archive have more points than F_P and for each point in F_P there is a point in $F(A)$ where the distance between them is zero. Then, all remaining points are not taken into account by the IGD indicator and can lie wherever in the objective space.

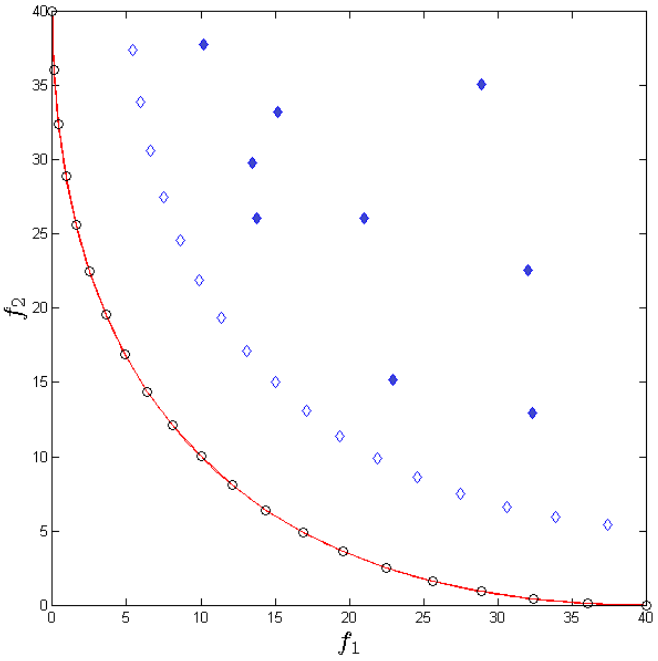


Figure 2.12: A small discretization of the Pareto front (black circles) is used to compute IGD indicator. Only the closest solutions in the population (empty diamonds) are considered meanwhile solutions far away from F_P are neglected.

2.4.3 Averaged Hausdorff Distance Δ_p

The Δ_p indicator proposed by Schütze et al. [37] can be viewed as an averaged version of the Hausdorff distance. This indicator slightly changes and combines the GD and IGD indicators in an attempt to prevent their individual disadvantages:

Definition 22 (Δ_p). *Let F_P be a subset of the Pareto front.*

$$GD_p(A, F_P) := \left(\frac{1}{|A|} \sum_{i=1}^{|A|} dist(a, F(P_Q))^p \right)^{1/p} \tag{2.33}$$

$$IGD_p(A, F_P) := \left(\frac{1}{|F_P|} \sum_{i=1}^{|F_P|} dist(p, F(A))_i^p \right)^{1/p} \tag{2.34}$$

$$\Delta_p := \max(GD_p(A), IGD_p(A)). \tag{2.35}$$

For $p = \infty$ it is $\Delta_p = d_H$. For $p < \infty$ the distances in Equations (2.33) and (2.34) are averaged. Thus, Δ_p can be viewed as an averaged Hausdorff distance.

2.4.4 UPCF

In 2013, Mohammadi et al. proposed a RP metric called user-preference metric based on a composite front or UPCF. UPCF was designed to avoid requiring the Pareto front beforehand by creating a 'reference Pareto front' from the non-dominated solutions of more than one algorithm. The procedure is as follows:

1. Combine all non-dominated solutions from the algorithms to be compared. This set of optimal solutions form the so called composite front.

2. For each RP, identify the closest solution in the composite front. A parameter r (specified by the user) defines the spread of solutions sought in the composite front. Only solutions within Euclidean distance r to the closest point will be considered for further computations.

3. Using the preferred region on the composite front as the reference Pareto front, a 'classical' indicator as IGD is applied on each population of the algorithms.

UPCF was meant to compare more than one algorithm as can be seen from Step 1, since for only one archive all non-dominated solutions belong to the same algorithm. Hence, any metric used would have a perfect value. Nevertheless, this metric was used with the IGD and Hyper volume to compare RMEAD and RNSGA-II.

3 | Directed Search for Reference Point Problems

We have discussed so far the generic structure of the RPP along with properties regarding the position of the RP. In this chapter, we will turn our attention to the Directed Search method presented in the previous chapter for solving RPPs such as (2.25) which we call Reference Point - DS or RDS.

The theoretical basis of the RDS is presented in Sections 3.1.1 and 3.1.2 which includes the descent and continuation phases, respectively, analog to the 'classical' DS. Further, for a feasible RP and in contrast to the classical formulation of the RPP, we present an alternative for finding better solutions according to the dominance relation. A novel feature consisting of neighborhood exploration of the solution of a RPMOP is discussed in Section 3.2. Finally, a brief discussion of the novel RDS is provided in Section 3.3.

3.1 RDS

3.1.1 Descent phase

In the previous section we have presented the highlights of the DS where we have emphasized the 'steering' property via a direction $d \in \mathbb{R}^k$. In order to compute such

a direction in the context of RPPs, recall the generic formulation of the RPP

$$\min_{\mathbf{x}\in\mathbb{R}^n} D(F(\mathbf{x}), Z), \tag{3.1}$$

where $D : \mathbb{R}^n \to \mathbb{R}$ is a chosen metric such as the p-norm, $F : \mathbb{R}^n \to \mathbb{R}^k$ is a k objective unconstrained MOP, and $Z \in \mathbb{R}^k$ is a given RP in objective space.

Now, given an initial point x_0 and a RP Z, the greedy direction d_G in objective space is certainly given by

$$d_G = Z - F(x_0). \tag{3.2}$$

Therefore, d_G can be used by the DS approach. That is, an application of DS is equivalent to the numerical realization of the following initial value problem (IVP):

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x}(t) &= J(x(t))^+(Z - F(x(t))), \quad t > 0. \end{aligned} \tag{3.3}$$

In order to understand the end point of the solution curve (3.3) we need to identify when $\dot{x}(t) = 0$.

Proposition 3. *Given a RP Z, the end point of (3.3) is a critical point of (3.1).*

Proof 3. *We know that an end point to (3.3) is such that $\dot{x}(t) = 0$. This is the case if one of the three cases is satisfied:*

1. $J(x(t))^+ = 0$.
2. $(Z - F(x(t))) = 0$.
3. $J(x(t))^+(Z - F(x(t))) = 0$.

- *If the given RP is feasible then there exists $x^* \in \mathbb{R}^n : F(x^*) = Z$. Hence, the solution curve of (3.3) ends when the second case is met.*

- *If the given RP is infeasible, consider the following auxiliary function*

$$g(x) = D(Z, F(x)) = \frac{1}{2}||Z - F(x)||_2^2. \tag{3.4}$$

We know that a critical point of this function can be found by setting the derivative to zero:

$$\begin{aligned} \nabla D(Z, F(x)) &= \left(\frac{1}{2}(2)||Z - F(x)||_2 \frac{(Z - F(x))^T}{||Z - F(x)||_2} J(x)\right)^T \\ &= J(x)^T(Z - F(x)) = 0. \end{aligned} \tag{3.5}$$

Hence, we can assume that the third condition for an end point of the solution curve of (3.3) is reached when the minimum of the RPP (3.1) is obtained. □

Equation (3.5) gives insight of another appealing reason to use RDS in RPPs, particularly when a feasible RP is used. Namely, that we can expect (local) quadratic convergence as the following discussion shows. If we consider (3.2) as a root finding problem, an iteration of the Gauss-Newton method (for which locally quadratical convergence is know, see [38]) is given by:

$$\begin{aligned} x_{i+1} &= x_i - J_g(x_i)^+ g(x_i) = x_i - J(x_i)^+(F(x_i) - Z) \\ &= x_i + J(x_i)^+(Z - F(x_i)), \end{aligned} \tag{3.6}$$

where $J_g(x)$ denotes the Jacobian of (3.4) at x.

On the other hand, an iteration performed via DS yields

$$x_{i+1} = x_i + t_i J(x_i)^+(Z - F(x_i)), \tag{3.7}$$

where $t_i \in \mathbb{R}_+$ is the chosen step size. Comparing (3.6) and (3.7) we see that both iterations coincide for $t_i = 1$. Such a step size can be used even when a RP is not feasible in order to reach Pareto (boundary) solutions as soon as possible.

It is worth noticing that, unlike the 'classical' DS where the direction $d \in \mathbb{R}^k$ dictates a straight line in direction d, (requiring consequently a corrector step to the line $F(x) + td$ the direction used in (3.3) is updated at each point $x(t)$.

3.1.2 Continuation phase

In order to see how to proceed in case the given RP is not feasible, we have to understand the geometry of the solution curves of IVP (3.3). For this, compare to Figure 3.1.

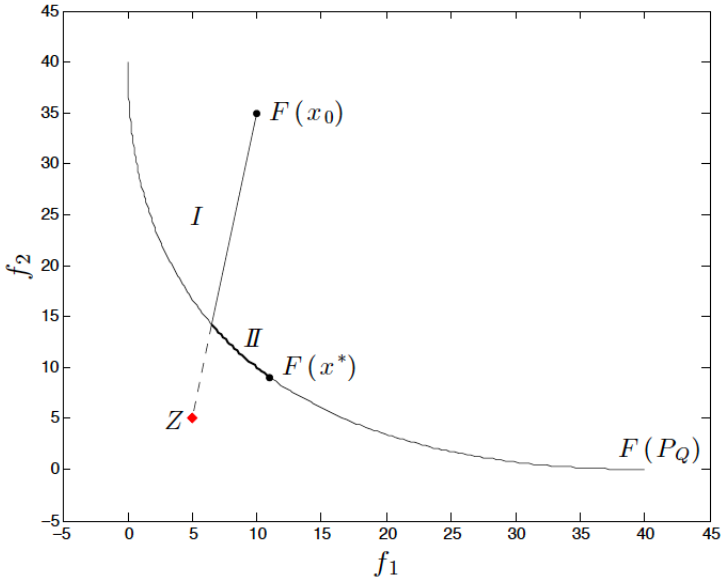


Figure 3.1: Typical image of a solution curve of (3.3).

We can divide such solution curves into two parts:

- In part I, a movement in d-direction is performed that accounts for the descent phase of RDS.
- Once a boundary point x of the MOP is reached, the movement is steered along the linearized Pareto front at $F(x)$. This movement along the Pareto front constitutes part II of the solution curve.

Unfortunately, the ordinary differential equation in (3.3) is stiff in part II which means that its numerical treatment gets complicated. To see the stiffness, let x be a boundary point. Then there exists a direction d such that the equation $J(x)\nu = d$ has no solution. This is equivalent to $\text{rank}(J(x)) < k$ which means that the condition number of $J(x)$ is infinite.

In order to overcome this stiffness, we can perform a linearization to steer the search directly as follows: let x be a boundary point with weight $\alpha \in \mathbb{R}^k$ such that:

$$\sum_{i=1}^{k} \alpha_i \nabla f_i(x) = 0. \tag{3.8}$$

Further, let a QR factorization of α be given,

$$\alpha = QR = (q_1, \ldots, q_k)R. \tag{3.9}$$

Then the column vectors of $Q_2 = (q_2, \ldots, q_k) \in \mathbb{R}^{k\times(k-1)}$ form an orthonormal basis of the linearized Pareto front at $F(x)$. Given the greedy direction d_G, the projection onto the tangent space is thus given by

$$d_{new} = Q_2(\underbrace{Q_2^T Q_2}_{=I})^{-1} Q_2^T d_G = Q_2 Q_2^T d_G. \tag{3.10}$$

Denote by $P(x) := Q_2(x)Q_2(x)^T$ the projection described above. Then we can solve the following IVP for points x_0, where $F(x_0)$ is on the boundary of the image:

$$\begin{aligned} x(0) &= x_0 \in \mathbb{R}^n \\ \dot{x}(t) &= J(x(t))^+ P(x(t))(Z - F(x(t))), \quad t > 0. \end{aligned} \tag{3.11}$$

The switch between (3.3) and (3.11) can be handled via monitoring the condition number of $J(x)$:

- If $J(x) < tol$ for a certain tolerance value we choose to follow the flow in (3.3).
- Otherwise we take (3.11)

The flow given by (3.11) can certainly lead to a new point away from the Pareto front when traversing it. Hence, a corrector step might be needed to bring the new solution back to the desired curve.

For a corrector direction, notice that for an iterate which is away from the Pareto front it is the again case that $J(x) < tol$. That is, we can use again the descent phase of RDS using the greedy direction d_G.

An observation from (3.5) can be used as a stopping criterion: recall that a critical point of the p-norm RPP is given by

$$\nabla D(Z, F(x)) = J(x)^T(Z - F(x)) = 0.$$

Now, assume a Pareto point x^*. We know by the KKT conditions for the unconstrained case (2.6) that there exists an associated weight $\alpha \in \mathbb{R}^k$ where the sum of gradients multiplied by each component of α is zero.

Hence, a solution to (3.1) satisfies the following two conditions:

- $\text{rank}(J(x)) < k \quad \leftrightarrow \quad \text{cond}(J(x)) = \infty.$
- α and $(Z - F(x))$ are collinear.

3.1.3 Step size control

Possible step size controls for the DS method have been discussed in [39, 40]. Although they were shown to be efficient for the classical DS, we can improve them for the problem at hand since we can take advantage of the fact that the RPP is in principle a SOP.

Motivated by the discussion above, we can set a step size equal to one expecting to achieve (local) quadratic convergence. However, if it is the case that the new iterate is farther away from the RP Z, we can attempt an Armijo backtrack as follows:

Let x_0 be an initial point with single objective value $f(x_0)$ where the SOP's objective function f is the RPP as defined in (3.4). A line search starting at x_0 is defined as $f_\nu(0) = f(x_0)$. Further, assume RDS yields a new iterate such that $f(x_1 = x_0 + t\nu) = f_\nu(t) > f(x_0)$. If we consider that the direction ν found by RDS is a also descent direction of (3.4) then it holds $f'_v(0) = \langle \nabla f(x), \nu^+ \rangle < 0$ (by looking at (3.5) we can obtain the derivative of f without additional cost).

Then, approximate f_v by a quadratic polynomial using the interpolation conditions

$$\begin{aligned} p(t) &= at^2 + bt + c. \\ p'(t) &= 2at + b = 0. \end{aligned} \tag{3.12}$$

By the considerations above, we know that p is a strongly convex function and has a unique minimizer. The solution to (3.13) is given by

$$\begin{aligned} t_{opt} &= \frac{-b}{2a} \\ &= \frac{-t^2 < \nabla f(x_0), \nu >}{2(f(x_1) - f(x_0) - t\langle \nabla f(x_0), \nu \rangle)}. \end{aligned} \tag{3.13}$$

If $f_v(t_{opt}) > f_v(0)$ we can repeat the process.

3.1.4 Box constraints

In the following, we consider box constrained MOPs of the form

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} F(x) \\ &\text{s.t. } l \leq x \leq u, \end{aligned} \tag{3.14}$$

where $l, u \in \mathbb{R}^n$ are the lower and upper limits, respectively.

Among the various methods for handling box constraints, the most effective one is probably the gradient projection method [41]. Each iteration of the method consists of two steps. In the first one, we perform a search in the descent (e.g. steepest descent) direction for the current iterate and in the second one the search direction is bended so that the iterate remains feasible.

That is if for a search direction and a step size a new iterate does not violate any of the box constraints, it can be accepted with no further considerations. On the other hand, if any of the inequality constraints is active ($x = l$ or $x = u$) the inequality constraint can be considered as an equality constraint of the following form

$$h_i(x) = \pm x_i + a_i = 0 \quad i = 1, \ldots, p. \tag{3.15}$$

In this case a movement orthogonal to the gradient of the active inequality constraints is sought, i.e.

$$\langle \nabla h_i(x), \nu \rangle = 0 \text{ for all } i = 1, \ldots, p. \tag{3.16}$$

This property can be easily fulfilled. The derivative of an active box constraint has the following form

$$\nabla h_i(x) = (0, \ldots, 0, \pm 1, 0, \ldots, 0)^T \quad i = 1, \ldots, p. \tag{3.17}$$

Thus, directions with $\nu_i = 0$ comply with (3.16).

3.1.5 Non linear constraints

In [39], Salinas proposed an extension for handling non linear constraints in the DS method which can be easily adapted to the RDS. Analog to the gradient projection

method, the idea is to handle the problem as if it is unconstrained until inequality constraints become active. In his work, the problem of a search direction is treated as a general least squares problem of the following form

$$\begin{aligned} \min_{x} \; & ||Ax - b||_2^2 \\ \text{s.t. } & Cx = d, \end{aligned} \tag{3.18}$$

where $A \in \mathbb{R}^{n\times k}, b \in \mathbb{R}^n, C \in \mathbb{R}^{m\times k}$ and $d \in \mathbb{R}^m$. In the context of DS, the problem becomes

$$\begin{aligned} \min_{\nu} \; & ||J(x)\nu - d||_2^2 \\ \text{s.t. } & H(x) = 0, \end{aligned} \tag{3.19}$$

where $H(x) \in \mathbb{R}^{q\times n}$ is the Jacobian of the q active equality and inequality constraints. The solution to (3.19) can be stated in closed form as

$$\begin{bmatrix} 2J(x)^T J(x) & H(x)^T \\ H(x) & 0 \end{bmatrix} \begin{bmatrix} \nu \\ \lambda^* \end{bmatrix} = \begin{bmatrix} J(x)^T d \\ 0 \end{bmatrix},$$

and solving for $(\nu, \lambda^*)^T$

$$\begin{bmatrix} \nu \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 2J(x)^T J(x) & H(x)^T \\ H(x) & 0 \end{bmatrix}^+ \begin{bmatrix} J(x)^T d \\ 0 \end{bmatrix}. \tag{3.20}$$

3.2 Neighborhood exploration

In the sense of (3.1), the problem is considered solved when a solution to the RPP is found. However, may times the purpose of DMs when establishing RPs is to explore optimal solutions in the Pareto front *similar* to the RP supplied, that is, to obtain

an overview of the region surrounding a RP. As presented in Chapter 2, EAs already include the means to accomplish this task by relying on the population's individuals to cover the sought regions. However, there does not exist such a specific tool in mathematical programming techniques. Here we will present a simple adaption of the RDS for 'exploring' the neighborhood of the solution to the RPP in the form of an algorithm.

3.2.1 Considerations

Let x^* be a Pareto point and the solution to the RPP found by RDS. Assume the Jacobian of $x^*, J(x^*)$, is given along with the associated convex weight α. Then, an exploration of the surrounding optimal solutions of x^* can be performed with no additional Jacobian computations for a small neighborhood.

Recall that RDS's direction in variable space ν is computed using only the Jacobian of a starting point (in this case, the solution of the RPP) which is then used in a line search. Now, as it has been discussed in [40], a sufficiently small step size t_x in variable space Q is equivalent to a step size t_y in objective space $\mathbb{R}^k$, i.e., $t_x \approx t_y$ for $t_x \to 0$. These two observations suggest that further Pareto points can be found by choosing first a suitable set of directions $D = \{d_1, \ldots, d_F\} \subset \mathbb{R}^k$ and then a set of step sizes $T = \{t_1, \ldots, t_N\} \subset \mathbb{R}$.

Such directions need to point along the Pareto front in order to guarantee the optimality of the neighboring solutions. For the special case of $k = 2$ such directions are restricted to either 'left up' or 'right down' in objective space. However, for $k > 2$ the set of directions is infinite. Here we propose to take the coordinate directions $\hat{e}_i \in \mathbb{R}^k \quad , i = 1, \ldots, l$, in positive and negative directions: $d_{i,1} = \hat{e}_i, d_{i,2} = -\hat{e}_i$ and to project them to the Pareto front using the projection operator defined in the second phase of RDS. All that remains is to provide a set of desired neighborhood sizes to explore in order to cover a certain extent of the Pareto front. The pseudo code of the neighborhood exploration can be found in Algorithm 7.

Algorithm 7 RDS Neighborhood Exploration

Require: x^*: solution to RPP, $J(x^*)$: Jacobian of MOP at x^* , α: convex weight, $N \subset \mathbb{R}$ neighborhood sizes.

Compute the projection operator $P(x^*)$ of the linearized Pareto front at x^* as in (3.10).

for each neighborhood size $n \in N$ **do**

for each coordinate axis $\hat{e}_i \quad i = 1, \ldots, k$ **do**

Project $\hat{e}_i$ using $P(x^*)$ and obtain $\dot{e}_i$.

Shoot in direction $d = \dot{e}_o - F(x^*)$ using step size $t = n$

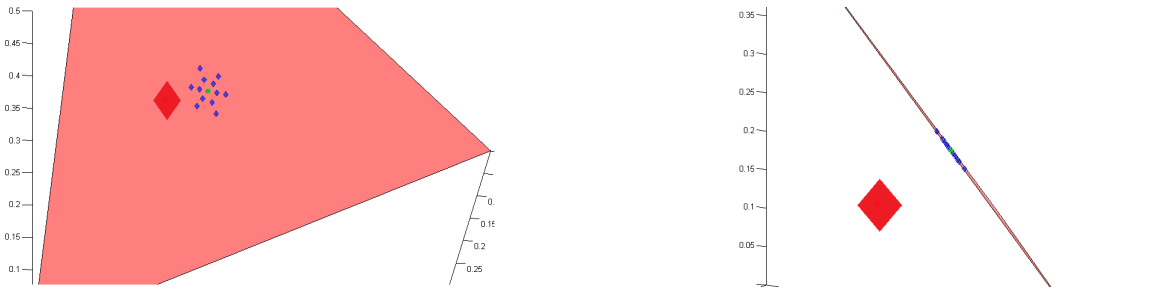
Project $-\hat{e}_i$ using $P(x^*)$ and obtain $-\dot{e}_i$.

Shoot in direction $d = -\dot{e}_i - F(x^*)$ using step size $t = n$

end for

end for

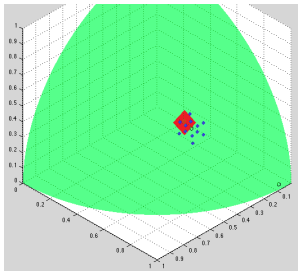
We will now present two examples of the neighborhood exploration on three objective functions. Namely these are the DTLZ1(linear) and DTLZ2(concave) problems. For both examples we have used a RP which is infeasible and computed the solution to the associated RPP which is a Pareto point with RDS. From there on we have used it as starting point for exploration. The red diamond represents the RP, the green circle is the solution $F(x^*)$ to the RPP, the blue diamonds are the neighboring solutions for two desired neighborhoods around the RP and still within the Pareto front.



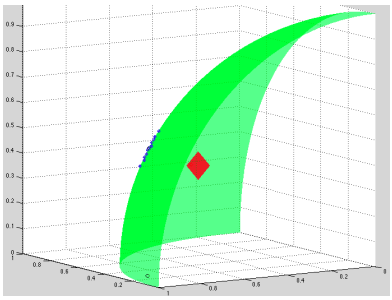
(a) Neighborhood exploration spread of solutions

(b) Neighborhood exploration optimality of solutions

Figure 3.2: Neighborhood exploration on DTLZ1.



(a) Neighborhood Exploration Spread



(b) Neighborhood Exploration Convergence

Figure 3.3: Neighborhood exploration on DTLZ2.

3.3 Discussion

In this chapter we presented a modification of DS, called RDS, to tackle RPPs. RDS has a big advantage over several other mathematical techniques, namely, that it can be made gradient free by sampling solutions in the neighborhood. This becomes specially useful when considering set based algorithms such as MOEAs. Further, almost all work done for DS can be used for RDS such as the constraint handling. Also, a novel method for exploring neighboring solutions from the original solution of a RPP without computing further derivatives was proposed.

4 | RDS within MOEAs

In this chapter we will present a hybridization of RNSGA-II and the RDS algorithms presented in the previous chapters. This memetic algorithm takes the best features from both approaches: the global overview of the problem given by the evolutionary algorithm and the accurate and focused search carried by the mathematical programming technique along with its convergence properties.

The resulting algorithm, called RDS-RNSGA-II, is first introduced for unconstrained and box constrained problems in Section 4.1 and for constrained problems in Section 4.2.3 where one comparison of RDS-RNSGA-II against RNSGA-II is shown. A modified version of the IGD indicator discussed in Chapter 2 is presented in Section 4.3.

4.1 Unconstrained and Box Constrained RPPs

In this section, we will introduce the base framework for the hybridization of RNSGA-II and RDS. First, we will address the general considerations of the memetic strategy which will be used in the memetic RPMOEA regardless of the model. Next, we will present the local search algorithm for the totally unconstrained case. The adaption for box constrained problems are presented last.

4.1.1 General considerations

When designing the memetic RDS-RNSGA-II algorithm, many aspects were taken into account in order to get a balance between the LS and the EA. The generic parameters suggested in [42]) are:

- The number of individuals to which LS will be applied.
- The maximal iteration number (depth) of the LS applied to an individual.
- Frequency of the same.

Next to these generic parameters there are some more specific ones related to RPPs. Firstly, in order to guarantee balanced convergence for each RPP, we decided to take RPs for improvement one at a time in a round robin fashion. That is, given a set of RPs $Z_1, \ldots, Z_l$, Z_1 is first taken for a RPP and marked as the current working RP. This RPP is optimized by RDS for a given budget, then Z_2 is taken as the current working RP and so on until Z_l is reached. If some budget is still available, the process is repeated in the same order.

The second issue to tackle is which solution to chose for improvement. This is the most important aspect in the memetic algorithm. Considering the non dominated sorting of NSGA-II, a LS algorithm could try to improve individuals on the first non dominated front and backwards from then on. This approach, however, is not useful in particular when the number of objectives increases since most of the individuals will belong to this front. Hence, a second and more promising alternative is to improve solutions which are already close to RPs whether they are dominated or not.

However, this selection scheme introduces a problem when a RP is found to be feasible during the evolution since there will exist better solutions according to the domination relation: a new iterate, closer to the current working RP found by local search would be discarded by RNSGA-II in favor of dominating solutions, wasting computational

effort in the process. Recall that the first goal of RNSGA-II is to obtain non dominated individuals and afterwards solutions close to the supplied RPs.

Therefore, a feasibility check of the RPs (presented in Algorithm 8) is performed always after a generation ends.

Algorithm 8 Feasibility check of RP

Require: Population P, supplied RPs $Z_1, ..., Z_l$, feasible RPs found so far FRP

Ensure: Feasible RPs so far FRP

```
for each Z_i ∈ Z do
    for each p_i ∈ O do
        Set F_i as the function value of individual p_i.
        if ||F_i − Z_i|| < tol or F_i <_p Z_i then
            Add Z_i to FRP
        end if
    end for
end for
```

In RDS-RNSGA-II, individuals are sorted according to a fitness function that takes into account the feasibility of the RP. Then, the best individual is improved. The fitness functions for each case is described below:

- If the RP is *not* feasible, the distance towards the RP becomes the fitness function. This way, closest solutions are preferred.
- If the RP *is* feasible, the distance towards the RP plus the non dominated rank times a constant becomes the fitness function. In this sense, dominating solutions are preferred and within the same front, closest solutions are taken.

In order to avoid premature convergence due to super elite individuals, the trail of solutions improved by the LS are stored and replace the worst solutions in the population.

Algorithm 9 shows the whole process of selection and replacement of RDS-RNSGA-II described so far.

Algorithm 9 Local Search Engine

Require: Population P, RPs $Z_1, ..., Z_l$, feasible RPs so far FRP, iterations $XDSI$, solutions to improve per RP $XDSPZ$

Set Sr as 0. ▷ Success rate of the RDS
for $i = 1$ to $XDSPZ$ **do**
  for each Z_i **do**
    if Z_i is feasible **then**
      Set the fitness function $F_f(I)$ as the distance towards Z_i plus a dominated penalty.
    else
      Set the fitness function $F_f(I)$ as the distance towards Z_i.
    end if
    Sort P using $F_f(I)$.
    Take the best solution and apply "Local Search" for $XDSI$ iterations.
    Store the trail of solutions obtained in a set TS.
    if $TS \neq \emptyset$ **then**
      Replace the worst solutions by each solution in TS one by one.
      Set Sr as $Sr + 1$.
    end if
    Check for feasibility of Z_i.
  end for
end for
Set Sr as $\frac{Sr}{|Z| \times XDSPZ}$.

A success rate variable Sr is computed in order to further apply or postpone the LS.

4.1.2 RDS within a MOEA

It is worth noticing that all the considerations discussed above are completely independent of the choice of the LS algorithm including the RDS. Hence, they are useful for both the gradient based and the gradient free version of the RDS algorithm. Both algorithms are used further or not depending on the success rate they have once applied. The reason for this is that in the first stages of the evolution, the EA's population is less likely to have neighborhood information available, being the opposite in the last stages, where the population is expected to converge towards the optimal areas.

This behavior suggests that at the beginning of the evolution, RDS will yield better solutions than the gradient free version and should be used to reach the Pareto front as quick as possible, meanwhile the RDDS can gain the most when the population members become closer and more search directions can be included to approximate the gradient.

In Algorithm 9, the "Local Search" procedure refers to the RDS and RDDS. We will now present in Algorithm 10 RDS within the memetic strategy.

Algorithm 10 RDS within a MOEA

Require: Initial point x_0 along with $F(x_0)$, RP Z, flag of Z's feasibility, iterations $XDSI$

1: Compute the Jacobian $J(x_0)$ at x_0.
2: Compute $\alpha = \text{argmin}_{\alpha \in \mathbb{R}^k} ||\nabla f_i(x_0)\alpha_i||$. Set $iter \leftarrow 0$.
3: Set x_H and F_H to the empty set. Add x_0 and $F(x_0)$ to x_H and F_H respectively. ▷ x_H and F_H will store the trials.
4: **while** $iter < XDSI$ **do**
5: Compute the greedy direction $d_G = Z - F(x_0)$.
6: **if** $\kappa\ (J(x_0)) < tol$ **then** ▷ x_0 lies on the Pareto front
7: Compute the projection operator as in (3.11).

8: Project d_G and obtain d.
9: **else**
10: **if** Z is feasible **then** ▷ Looking for dominating solutions
11: Set d as $-(1,\ldots,1)^T$
12: **else** ▷ Looking for closer solutions
13: Set d as d_G.
14: **end if**
15: **end if**
16: Normalize d as $\frac{d}{||d||_2}$
17: Compute direction $\nu = J(x_0)^+ d$
18: Modify ν to comply with box constraints.
19: Compute a suitable step size t.
20: **while** $t > tol_2$ **do**
21: Compute $p = x_0 + \nu t$ and $F(p)$.
22: **if** Z is feasible **then**
23: **if** $D(Z, F(p)) < D(Z, F(x_0)$ **then**
24: Accept p.
25: **else**
26: Backtrack using $J(x_0)$ and reducing step size.
27: **end if**
28: **else**
29: **if** $p \prec x$ or $(D(Z, F(p)) < D(Z, F(x_0))$ and $x_0 \not\prec p)$ **then**
30: Accept p.
31: **else**
32: Reduce the step size by some factor.
33: **end if**
34: **end if**

35: **if** p accepted **then**
36: Exchange x_0 by p and $F(x_0)$ for $F(p)$.
37: Add x_0 and $F(x_0)$ ton x_H and F_H respectively.
38: **end if**
39: **end while**
40: Update the Jacobian $J(x_0)$ and the associated convex weight α.
41: Set $iter \leftarrow iter + 1$.
42: **end while**

Two features of this version of RDS are to be noticed when the flag of feasibility is active.

1. The direction $d_D = -(1, \ldots, 1)^T$ is used to find a curve of dominating solutions in order to aid the RNSGA-II converge faster to the Pareto front. Furthermore, if a solution already lies on the Pareto front, the projection operator as defined in the previous chapter can be used to project the RP to the Pareto front and compute a better direction in order to bring a solution closer to the real solution.

2. The criteria to accept a new iterate p takes into account the case when the initial solution x_0 is not a Pareto point ($p \prec x_0$) and when it is a Pareto point ($(D(Z, F(p)) < D(Z, F(x_0))$ and $x_0 \not\prec p$)).

Correspondingly, the RDDS is modified to work as local search engine and isp-resented in Algorithm 11. This algorithm works with one individual at the time. The best individuals are chosen to construct the matrix V and correspondingly $\mathcal{F}$. If the number of neighbors is greater or equal to a value T_p, RDDS is expected to have improvement. This value has to hold a relation with the number of variables n (a more thorough discussion on the choice of neighbors can be found in [39]).

Algorithm 11 RDDS within a MOEA

Require: Initial point x_0 along with function value $F(x_0)$, RP Z, iterations $XDSI$, population P, neighborhood size N_h, test points required T_p

Search for individuals in P within neighborhood N_h of x_0. Store in P_N.
Set V and $\mathcal{F}$ as the empty matrices.
Set a counter c to 0.
for each inividual $i \in P_N$ **do**
    Set $m_{i,j}$ as $\frac{F_i - F(x_0)}{||x_i - x_0||}$.
    Set $\nu_{i,j}$ as $\frac{x_i - x_0}{||x_i - x_0||}$. Add $\nu_{i,j}$ to V as a column vector.
    if $\kappa(V) > tol$ **then**
        Remove $\nu_{i,j}$ from V.
    else
        Add $m_{i,j}$ to $\mathcal{F}$ as a column.
        Increment the counter c.
    end if
end for
if $c \geq T_p$ **then**
    if Z is feasible **then** ▷ Looking for dominating solutions
        Set d as $-(1, \ldots, 1)^T$
    else ▷ Looking for closer solutions
        Set d as the greedy direction $d_G = Z - F(x_0)$.
    end if
    Compute $\lambda = \mathcal{F}^{-}1d$.
    Set $\nu = V\lambda$ and normalize it as $\nu = \frac{\nu}{||\nu||}$
    Modify ν to comply with box constraints.
    Compute a suitable step size t.
end if

Set *iter* to 0.
while $t > tol_2$ or $iter > XDSI$ **do**
    Compute $p = x_0 + \nu t$.
    Compute $F(p)$.
    if $p \prec x_0$ or $(D(Z, F(p)) < D(Z, F(x_0))$ and $x_0 \not\prec p)$ **then**
        Accept p.
    else
        Reduce the step size by some factor.
    end if
    if p accepted **then**
        Exchange x by p and $F(x_0)$ for $F(p)$.
    end if
    Set *iter* as $iter + 1$.
end while

4.1.3 Example

For the following example consider the bi-objective problem

$$f_1(x) = ||x - a_1||_2^2, \quad f_2(x) = ||x - a_2||_2^2, \tag{4.1}$$

where $a_1 = (1, ..., 1)^T \in \mathbb{R}^{100}$, $a_2 = -a_1$, $Q = \mathbb{R}^{100}$ and
$Z = \{(20, 200)^T, (100, 50)^T, (250, 10)^T, (150, 150)^T\}$.

In Figure 4.1, blue circles represent individuals of a RNSGA-II's population. Circles filled with red are selected for LS (RDS and RDDS) since they are the closest in objective space with respect to the given RPs.

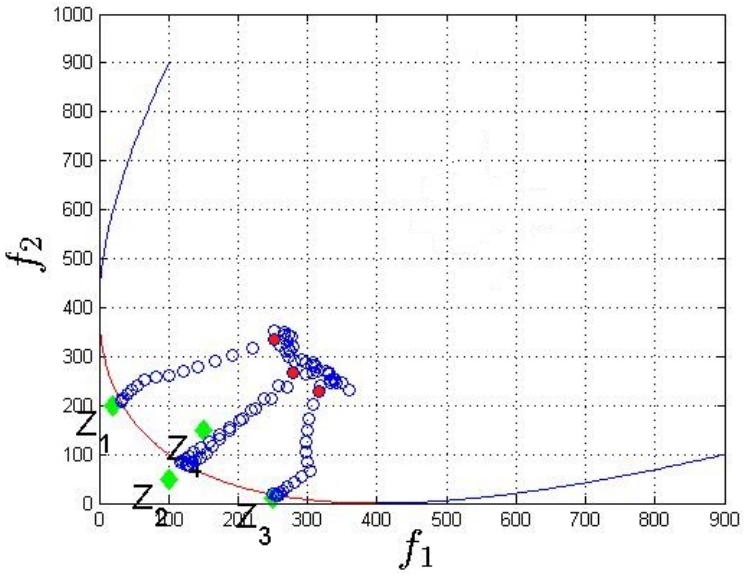


Figure 4.1: Local search operator on RDS-RNSGA-II's population for problem (4.1).

It can be seen from Figure 4.1 that a great improvement can be expected in the first stages of the evolution. The trail of solutions is stored and replace the farthest solutions accordingly in order to prevent super elite individuals. Even in the presence of a feasible RP, RDS can accelerate convergence by searching in the dominating zones of the MOP.

4.2 Constrained RPPs

In this section, we will present extended versions of the local search engine selection mechanism, the RDS and RDDS.

4.2.1 Constraint handling RNSGA-II

RNSGA-II does not include a mechanism for handling constraints per se as it was tested only on unconstrained or box constrained problems. In turn, the generic constraint-domination relation proposed by Deb in [29] can be easily included to

tackle such problems. The constraint handling RNSGA-II remains the same except for the replacement of the domination relation for the constraint-domination in Algorithm 3. This new relation requires the constraint violation value defined as:

Definition 23 (Constraint Violation)**.** *The constraint violation of a solution x is defined as:*

$$C_V(x) = \sum_{i=1}^{p} \max(0, g_i(x)) + \sum_{i=1}^{q} |h_i(x)|. \tag{4.2}$$

Hence, *feasible* solutions have a constraint violation value of zero and *infeasible* solutions have a constraint violation value greater than zero.

Definition 24 (Constraint-Domination)**.** *A solution $x \in \mathbb{R}$ is said to constraint-dominate a solution $y \in \mathbb{R}$ if either of the following conditions are true*

- *x and y are infeasible and $C_V(x) < C_V(y)$.*
- *x is feasible and y is infeasible.*
- *x and y are feasible and $x \prec y$.*

That is, (i) for infeasible solutions, the one with the less constraint violation wins, (ii) a feasible solutions always win over an infeasible one and (iii) for feasible solutions, the one dominating wins. Therefore, the algorithm remains the same as in the unconstrained case in absence of constraints.

RNSGA-II can be said to optimize two different functions during the evolution process. When only infeasible solutions are present, the algorithm becomes a SOP optimizer that directs the search towards less constraint violating region and once feasible solutions are found the original RPP takes over as the main objective.

4.2.2 Further considerations

When constraints have to be considered in a memetic algorithm the selection of individuals is, once again, a crucial decision. In this case, in order to agree on a LS algorithm within RNSGA-II, it has to minimize the constraint violation value of the best individual and once feasible solutions are available, retake the RPP. Nevertheless, solutions already close to RPs with small constraint violation can yield less constraint violating or even feasible solutions depending on the landscape of the MOP. Motivated by this we reformulate the fitness function for selection of individuals to further include the constraint violation of a solution.

- If the RP is *not* feasible, the distance towards the RP plus the constraint violation value becomes the fitness function.

- If the RP *is* feasible, the distance towards the RP plus the constraint violation value plus the non dominated rank times a constant becomes the fitness function.

If the objective space and the constraints are normalized in a proper way (see for example [3]) a balance can be considered between the search of feasible solutions and the distances for the RPP.

4.2.3 Constrained RDS within a MOEA

Motivated by the discussion above, RDS does not allow a new iterate to have more constraint violating value than the original starting point. In addition, if any constraints are active, the proper system of equations is modified to comply with such constraints. The constrained RDS is presented in Algorithm 12.

Algorithm 12 Constraint Handling RDS

Require: Initial point x_0 along with function and constraint value $F(x_0), G(x_0), H(x_0)$, RP Z, flag of Z's feasibility, iterations XDSI

Compute the Jacobian $J(x_0)$ and the derivative of the constraints $J_G(x_0), J_H(x_0)$ at x_0

Solve (2.8) for active equality and inequality constraints.

Set $iter \leftarrow 0$. ▷ These sets will store the trials.

Set x_H and F_H to the empty set.

Add x_0 and $F(x_0)$ to x_H and F_H respectively.

while $iter <$ XDSI **do**

    Compute the greedy direction $d_G = Z - F(x_0)$.

    if $\kappa\ (J(x_0)) < tol$ **then** ▷ x_0 lies on the Pareto front

        Compute the projection operator as in (3.11).

        Project d_G and obtain d.

    else

        if Z is feasible **then** ▷ Looking for dominating solutions

            Set d as $-(1, \ldots, 1)^T$

        else ▷ Looking for closer solutions

            Set d as d_G.

        end if

    end if

    Normalize d as $\frac{d}{||d||_2}$

    if there are active constraints **then**

        Compute ν as in (3.20)

    else

        Compute $\nu = J(x_0)^+ d$

    end if

    Modify ν to comply with box constraints.

    Compute a suitable step size t.

26: **while** $t > tol_2$ **do**
27: Compute $p = x_0 + \nu t$ and $F(p)$.
28: **if** $C_V(p) > C_V(x_0)$ **then**
29: Reduce step size.
30: Continue
31: **end if**
32: **if** Z is feasible **then**
33: **if** $D(Z, F(p)) < D(Z, F(x_0)$ **then**
34: Accept p
35: **else**
36: Backtrack using $J(x_0)$ and reducing step size.
37: **end if**
38: **else**
39: **if** $p \prec x$ or $(D(Z, F(p)) < D(Z, F(x_0))$ and $x_0 \not\prec p)$ **then**
40: Accept p.
41: **else**
42: Reduce the step size by some factor.
43: **end if**
44: **end if**
45: **if** p accepted **then**
46: Exchange $x_0, F(x_0), G(x_0), H(x_0)$ for the corresponding values of p.
47: Add x_0 and $F(x_0)$ to x_H and F_H respectively.
48: **end if**
49: **end while**
50: Update the Jacobian $J(x_0)$, the derivative of the constraints $J_G(x_0), J_H(x_0)$.
51: Solve (2.8) for active equality and inequality constraints.
52: Set $iter \leftarrow iter + 1$.
53: **end while**

4.3 A modified version of the IGD indicator for RPPs

As discussed in Chapter 2 (in page 40), performance indicators are tailored to assess the final outcome of a MOEA. However, in a memetic strategy it is also important to evaluate the convergence of the algorithms in previous stages of the optimization process. We decided not to take the RP indicator UPCF since it is meaningless before the final output is available. Furthermore, the advantages of the memetic strategy (in terms of convergence) are diminished when taking only the final archive.

Recall the classical RPP for point-wise iterative methods

$$\min_{x \in Q} D(Z, F(x)), \tag{4.3}$$

where $Z \in \mathbb{R}^k$ is the given RP and D is a chosen metric. Since a MOEA is dealing with entire sets of candidate solutions (populations or archives), it is advantageous to re-state the problem as

$$\min_{\substack{A \subset Q \\ |A|=N}} \text{dist}(Z, F(A)), \tag{4.4}$$

where A is an archive of magnitude N, and *dist* measures the distance of two sets as in Equation (2.28) and (2.29).

The advantages of (4.4) over (4.3) in our context are that (i) archives can be considered to be 'good' even if they contain elements $a_i \in A$ that are far away from Z (those elements will not be selected by the DM anyway), and (ii) this concept can be extended to the case $Z = Z_1 \cup \ldots \cup Z_l$ contains multiple RPs Z_i, $i = 1, \ldots, l$.

We have chosen to take the IGD indicator ([43]) applied to RPPs which can be viewed as an averaged version of the distance measurement (4.4):

$$\text{IGD}_Z(F(A), Z) = \frac{1}{|Z|} \sum_{i=1}^{|Z|} \min_{j=1}^{|A|} \text{dist}(Z_i, F(a_j)). \tag{IGD_Z}$$

Hereby, $A \subset Q$ is the given population or archive and dist the chosen distance metric. Hence, IGD_Z averages the distance of the closest member of the archive to a RP. Needless to say, the optimal value of IGD_Z is zero.

However, for constrained problems, the archive A has to filter its members in order to have only feasible solutions: although infeasible solutions can be the closest to a respective RP, they are not interesting for DMs and will likely be discarded by a MOEA, resulting in a non monotonically decreasing function which does not reflect convergence towards the solution. If only infeasible solutions are present in a generation, the archive will be composed of the less violating solution.

5 | Numerical Results

In this chapter we will present numerical results on unconstrained, box constrained, and constrained problems where we concentrate on models with small number of objectives (up to five). The novel memetic algorithm RDS-RNSGA-II is compared against its base algorithm RNSGA-II. All results are averaged over 30 independent runs. The best, median and worst IGD_Zvalues are computed and presented in a table for each problem at three different stages of the evolution: early, middle and later stage where significant improvement is achieved in the early and middle stages for the majority of the test problems. The last row are the results of the t-test (p-values) using a 95% confidence interval.

5.1 Parameter setting

In the following problems, RDS-RNSGA-II and RNSGA-II share the evolutionary algorithm's parameters: population size, generations, crossover and mutation probability and ϵ spread. Table 5.1 presents the corresponding value for each problem. For RDS-RNSGA-II the parameters shown in Table 5.2 were used for RDS and RDDS. Since the parameters are changed adaptively during the evolution, we have set in addition to the initial values, lower and upper limits to each of the variables. Finally, all problems are normalized in objective space by dividing each objective by the nadir point minus the ideal point.

	P	G	C_P	M_P	ϵ
CONV	100	200	0.9	0.01	1E-3
ZDT1	100	200	0.9	0.03	1E-3
ZDT2	100	200	0.9	0.03	1E-3
ZDT3	100	200	0.9	0.03	1E-3
ZDT4	100	200	0.9	0.1	1E-3
DTLZ2 $k = 3$	100	200	0.9	0.03	1E-3
DTLZ2 $k = 5$	100	200	0.9	0.03	1E-3
DTLZ3	100	500	0.9	0.14	1E-3
C1-DTLZ1	100	250	0.9	0.14	1E-5
C2-DTLZ2	100	150	0.9	0.083	1E-5
C2-CONVEX DTLZ2	100	150	0.9	0.083	1E-5
Welded Beam	80	100	0.9	0.25	1E-5
Car Side Impact	80	100	0.9	0.14	1E-5
Water Problem	100	100	0.9	0.33	1E-5

Table 5.1: RNSGA-II parameter setting for each test problem P population size, G generations, C_P crossover probability, M_P mutation probability and ϵ spread.

	I_C	F			D			SPZ		
		Max	Ini	Min	Max	Ini	Min	Max	Ini	Min
RDS	1 (G)	11 (G)	0.1 (B)	0.15 (B)	2 (G)	2 (G)	4 (G)	1		
RDDS	0.05 (G)	6 (G)	0.05 (B)	0.15 (B)	1			5	5	10

Table 5.2: Initial, maximum and minimum parameters of RDS and RDDS: I_C initial call, F frequency of LS, D depth of the same and SPZ number of solutions to improve for each RP. Parameters are updated based on the success rate of the LS. Parameters are stated in relation to a specific generation (G) or a percent of the budget of generations (B).

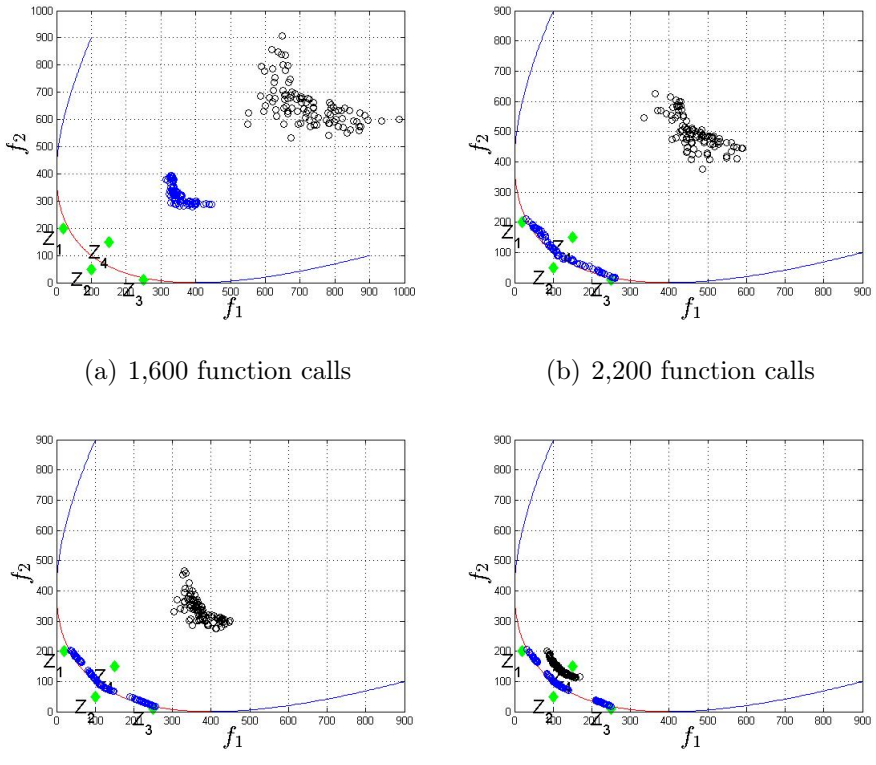
5.2 Unconstrained models

5.2.1 CONV

First, we consider the bi-objective problem CONV ([44])

$$f_1(x) = ||x - a_1||_2^2, \quad f_2(x) = ||x - a_2||_2^2, \tag{5.1}$$

where $a_1 = (1, ..., 1)^T \in \mathbb{R}^{100}$ and $a_2 = -a_1$, and where $Q = \mathbb{R}^{100}$ is the domain. We chose the four RPs $Z = \{(20, 200)^T, (100, 50)^T (250, 10)^T (150, 150)^T\}$. Since CONV is a convex problem large improvements are expected via the help of RDS. This is indeed the case as can be seen in Figure 5.1 as well as in Table 5.3, where the IGD_Z values for all the examples are significantly better in all three stages.



(a) 1,600 function calls

(b) 2,200 function calls

(c) 3,000 function calls

(d) 10,000 function calls

Figure 5.1: Numerical results of RNGSA-II (black) and RDS-RNGSA-II (blue) on CONV.

	Early (600)	Middle (2,000)	Later (5,000)
RDS-RNSGA-II	**0.906710** **1.101572** (0.094642) **1.254400**	**0.003985** **0.011901** (0.004613) **0.020219**	**0.000496** **0.002284** (0.001145) **0.004600**
RNSGA-II	2.582100 2.899363 (0.121790) 3.154800	1.002100 1.146750 (0.073128) 1.300100	0.259280 0.338415 (0.039805) 0.452310
RDS-RNSGA-II vs RNSGA-II	4.441786e-35	3.385631e-36	9.442286e-29

Table 5.3: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

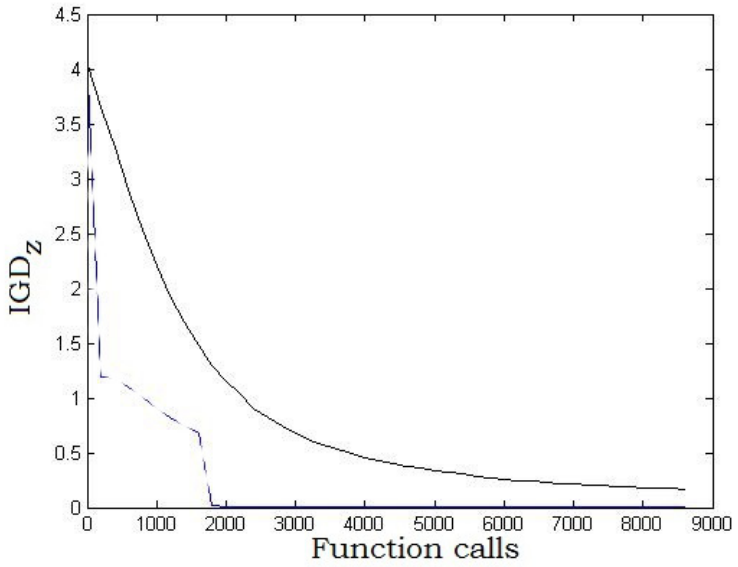


Figure 5.2: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on CONV.

5.2.2 ZDT

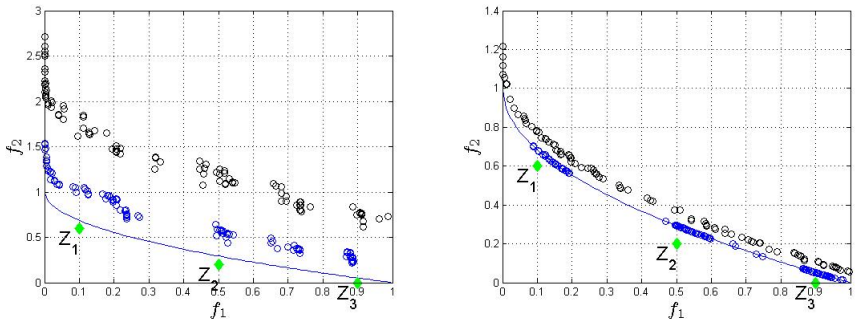
Now we consider the bi-objective box constrained functions ZDT1 up to ZDT4 functions from the ZDT benchmark suite ([45]).

ZDT1

The ZDT1 function is defined as

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)(1 - \sqrt{\frac{f_1(x)}{g(x)}}) \\ g(x) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i = 1, \ldots, n. \end{aligned} \tag{5.2}$$

We used $n = 30$ and $Z = \{(0.1, 0.6)^T, (0.5, 0.2)^T, (0.9, 0)^T\}$. Compared to CONV, RDS-RNSGA-II needs more function evaluations on ZDT1 to obtain a covering of the optimal solutions near the RPs, but comes quite close after already 6,000 function calls (compare to Figure 5.3 and Table 5.4).



(a) 1,800 function calls (b) 6,000 function calls

Figure 5.3: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on ZDT1.

	Early (400)	Middle (1,800)	Later (6,000)
RDS-RNSGA-II	**0.314820** **0.505073** (0.081484) **0.633980**	**0.067492** **0.141993** (0.056467) **0.271670**	**0.000253** **0.001451** (0.000837) **0.003725**
RNSGA-II	1.335100 1.540096 (0.116520) 1.742100	0.429280 0.490541 (0.055729) 0.636430	0.030279 0.044529 (0.009092) 0.067333
RDS-RNSGA-II vs RNSGA-II	4.198558e-21	2.737065e-16	3.617890e-18

Table 5.4: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages.The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

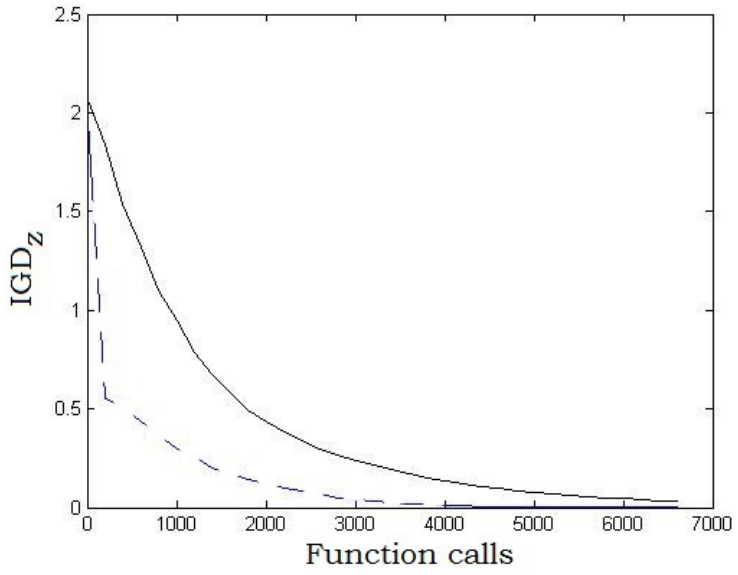


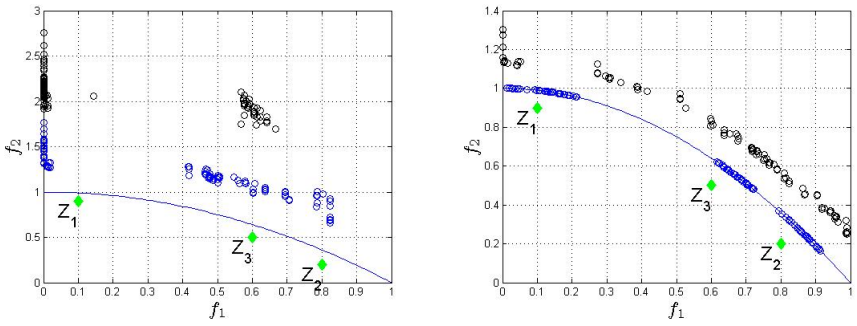
Figure 5.4: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on ZDT1.

ZDT2

ZDT2 represents a challenge for evolutionary algorithms since typically in first stages of the search, all solutions concentrate on the left top area of the objective space where weak Pareto points exist. The mathematical definition of ZDT2 is as follows

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left(1-\left(\frac{f_1(x)}{g(x)}\right)^2\right) \\ g(x) &= 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i=1,\ldots,n. \end{aligned} \tag{5.3}$$

Here,we can see in Figure 5.5 that RDS-RNSGA-II achieves a focus on the desired areas here given by $Z = \{(0.1, 0.9)^T, (0.8, 0.2)^T, (0.6, 0.5)^T\}$ after a budget of 6,200 function calls. Table 5.5 confirms that the performance of the memetic algorithm is significantly better in the best, median and worst cases for the checkpoints considered.



(a) 2,000 function calls (b) 6,200 function calls

Figure 5.5: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on ZDT2.

	Early (2,000)	Middle (3,600)	Later (6,200)
RDS-RNSGA-II	**0.138220** **0.282305** (0.114797) **0.581880**	**0.008232** **0.028267** (0.015262) **0.065586**	**0.000145** **0.000724** (0.000371) **0.001714**
RNSGA-II	0.760730 1.058190 (0.195260) 1.487200	0.294620 0.553861 (0.226002) 0.995980	0.056378 0.199139 (0.212441) 0.683100
RDS-RNSGA-II vs RNSGA-II	1.127396e-14	2.110516e-11	9.662720e-05

Table 5.5: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

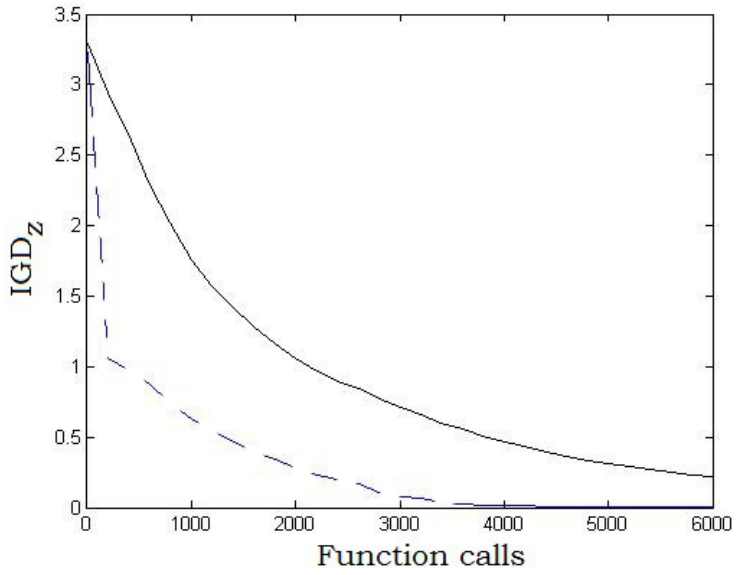


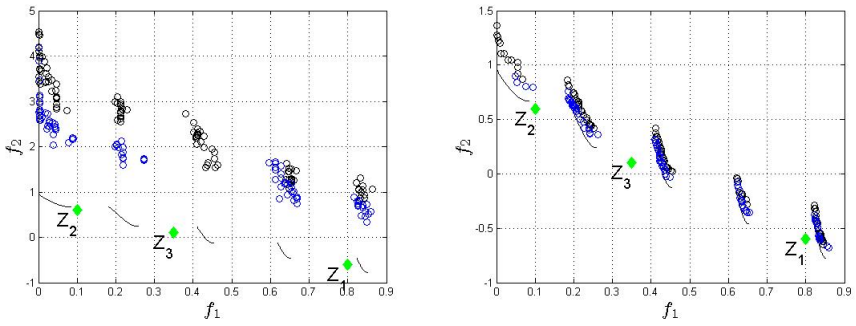
Figure 5.6: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on ZDT2.

ZDT3

ZDT3 has a more complex geometry than the previous MOPs, where not all boundary solutions are optimal. ZDT3 is defined as

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left(1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)}\sin\left(10\pi f_1(x)\right)\right) \\ g(x) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i = 1, \ldots, n. \end{aligned} \tag{5.4}$$

Here, we have chosen $Z = \{(0.8, -0.6)^T, (0.1, 0.6)^T, (0.35, 0.1)^T\}$. In the early stages of the evolution, the memetic strategy is outperforming RNSGA-II (compare to Figure 5.7 and Table 5.6). Nevertheless, at a budget of 5,400 function calls solutions both algorithms are very close together.



(a) 800 function calls (b) 5,400 function calls

Figure 5.7: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on ZDT3.

	Early (800)	Middle (2,600)	Later (5,400)
RDS-RNSGA-II	**0.322600** **0.494672** (0.112499) **0.712720**	**0.054966** **0.115268** (0.035003) **0.176390**	**0.003117** **0.027388** (0.018857) **0.078693**
RNSGA-II	0.561860 0.655991 (0.045888) 0.772640	0.118280 0.163294 (0.031888) 0.217180	0.027756 0.046208 (0.015386) 0.111150
RDS-RNSGA-II vs RNSGA-II	9.076490e-07	3.614816e-05	1.254285e-03

Table 5.6: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

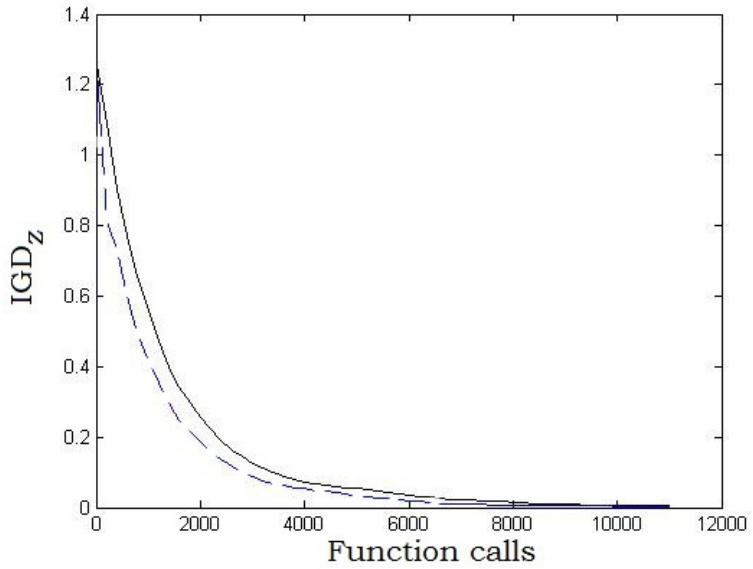


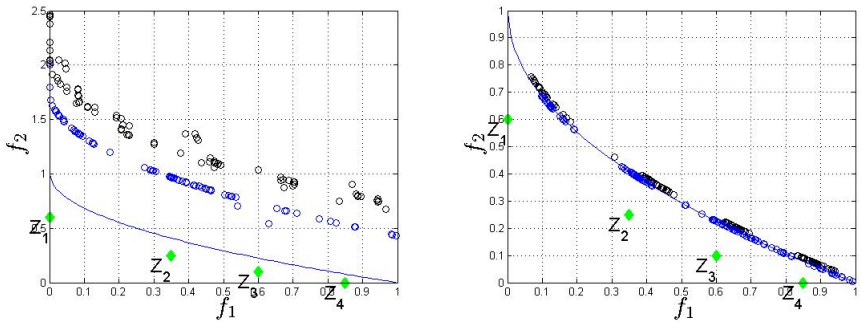
Figure 5.8: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on ZDT3.

ZDT4

ZDT4 is a convex multimodal problem where the challenge is to overcome 21 local Pareto fronts when using $n = 10$. The problem definition is as follows

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left(1 - \sqrt{\frac{f_1(x)}{g(x)}}\right) \\ g(x) &= 1 + 10(n-1) + sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)) \\ \text{s.t.} \quad & 0 \leq x_1 \leq 1 \\ \text{and} \quad & 0 \leq x_i \leq 5 \quad i = 2, \ldots, n. \end{aligned} \tag{5.5}$$

The search space ranges from -5 to 5 for the nine last variables and 0 to 1 for the first one. We have chosen the RPs $Z = \{(0, 0.6)^T (0.35, 0.25)^T (0.6, 0.1)^T (0.85, 0)^T\}$. The multimodal nature of ZDT4 makes it more difficult to reach the Pareto front in comparison to the previous problems. However, as can be seen from Figure 5.9 and Table 5.7 there is a better distribution and convergence of solutions even in the local fronts.



(a) 8,400 function calls (b) 18,200 function calls

Figure 5.9: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on ZDT3.

	Early (800)	Middle (8,400)	Later (18,200)
RDS-RNSGA-II	**8.220500** **27.265883** (10.165459) **56.936000**	**3.780700** **8.507060** (3.037916) **15.567000**	**0.237630** **0.692766** (0.371938) **1.673900**
RNSGA-II	33.492000 48.271767 (6.739564) 61.591000	4.933700 11.554433 (3.592855) 20.110000	0.571870 1.509929 (0.586590) 3.092100
RDS-RNSGA-II vs RNSGA-II	7.005070e-11	9.653172e-04	2.846882e-06

Table 5.7: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

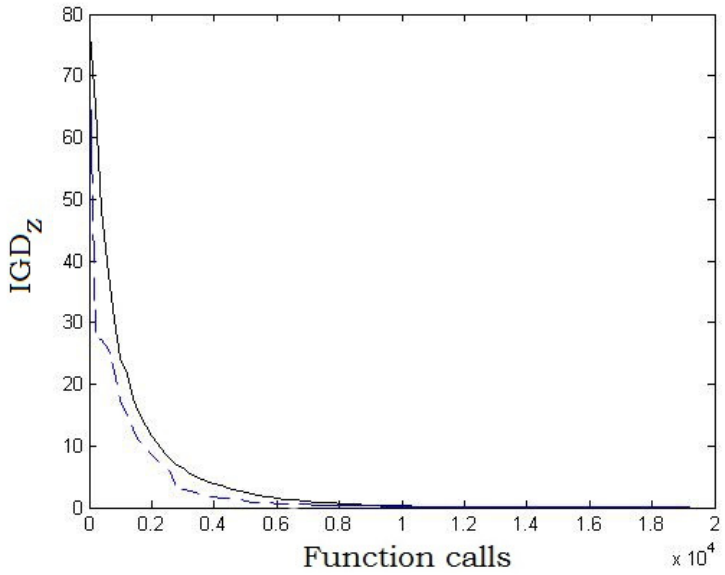


Figure 5.10: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on ZDT4.

5.2.3 DTLZ

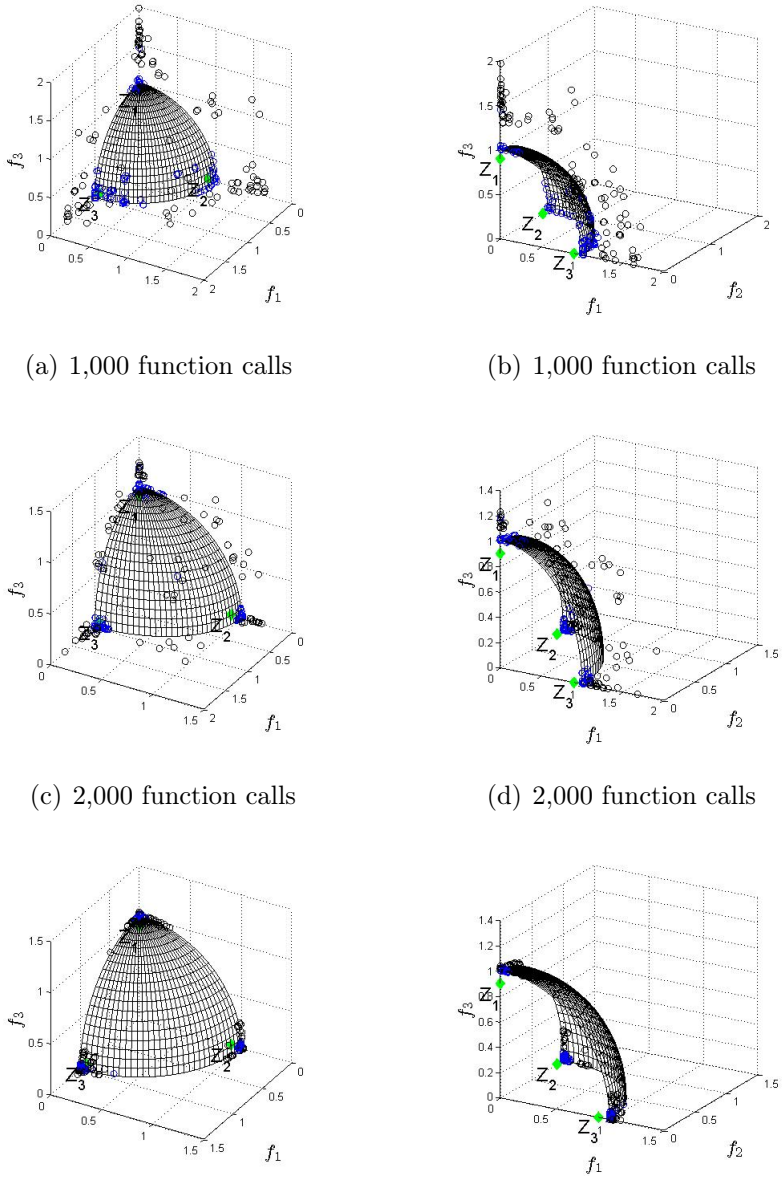
The DTLZ benchmark suite functions are scalable in the number of variables and objectives and each one poses different challenges to MOEAs and memetic algorithms. In this study we decided to take the three and five objective DTLZ2 presented in the original RNSGA-II paper and the three objective DTLZ3 problem.

Three Objective DTLZ2

In its general form, DTLZ2 is defined as

$$\begin{aligned} f_1(x) &= \cos(\frac{\pi}{2}x_1)\ldots\cos(\frac{\pi}{2}x_{k-1})(1+g(x)) \\ f_2(x) &= \cos(\frac{\pi}{2}x_1)\ldots\sin(\frac{\pi}{2}x_{k-1})(1+g(x)) \\ &\ldots \\ f_k(x) &= \sin(\frac{\pi}{2}x_1)(1+g(x)) \\ g(x) &= \sum_{i=k}^{n}(x_i-0.5)^2 \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i=1,\ldots,n. \end{aligned} \tag{5.6}$$

For the first RPP we have chosen $k = 3$,$n = 20$ and $Z = \{(0,0,0.9)^T(0,0.9,0)^T(0.9,0,0)^T\}$. It can be seen from Figure 5.11 that great improvement is reached even in the early stage. Table 5.10 shows that the performance of RDS-RNSGA-II is significantly better in the early, middle and later stage.



(a) 1,000 function calls

(b) 1,000 function calls

(c) 2,000 function calls

(d) 2,000 function calls

(e) 5,000 function calls

(f) 5,000 function calls

Figure 5.11: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on DTLZ2.

	Early (600)	Middle (1,000)	Later (3,000)
RDS-RNSGA-II	**0.009449** **0.021091** (0.007473) **0.042114**	**0.001590** **0.004786** (0.001778) **0.007548**	**0.000282** **0.001445** (0.000733) **0.003115**
RNSGA-II	0.480290 0.583997 (0.057965) 0.736570	0.051691 0.075625 (0.012442) 0.097156	0.006714 0.011200 (0.002145) 0.014696
RDS-RNSGA-II vs RNSGA-II	2.020206e-25	6.097794e-20	3.835055e-17

Table 5.8: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

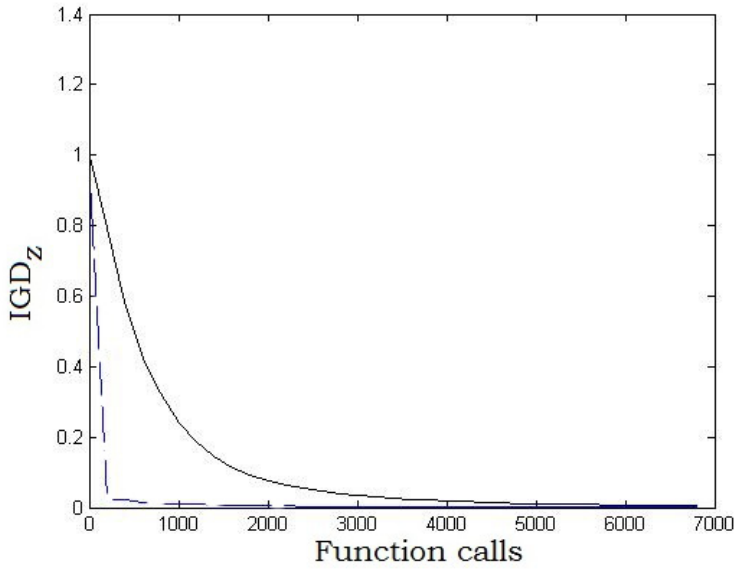


Figure 5.12: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on DTLZ2 with $k = 3$.

Five Objective DTLZ2

For the next RPP, we choose to take DTLZ2 with five objectives and 20 decision variables with RPs $Z = \{(0.25, 0.25, 0.25, 0.25, 0.25)^T\}$. Table 5.9 shows that a better performance is achieved by the memetic algorithm RDS-RNSGA-II even in the early stage of 600 function calls. Figure 5.13 shows how RDS-RNSGA-II's local search brings the IGD_Z closer to the best value. RPs This problem shows that RDS-RNSGA-II keeps outperforming the traditional MOEA regardless of the increase in the number of objectives. The t-test shows that there is indeed a statistical difference between both algorithms.

	Early (600)	Middle (1,000)	Later (3,000)
RDS-RNSGA-II	**0.451640** **0.521481** (0.058174) **0.634230**	**0.441670** **0.449111** (0.007101) **0.469450**	**0.440990** **0.441105** (0.000109) **0.441490**
RNSGA-II	0.765460 0.928997 (0.083361) 1.089100	0.473330 0.492402 (0.013223) 0.524710	0.442150 0.443092 (0.000635) 0.444720
RDS-RNSGA-II vs RNSGA-II	4.582916e-16	2.965448e-13	1.190214e-13

Table 5.9: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

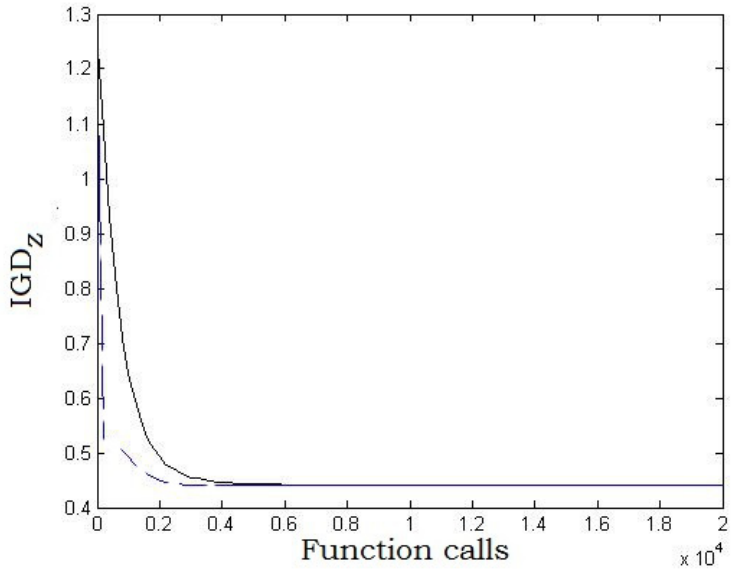


Figure 5.13: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on DTLZ2 for $k = 5$.

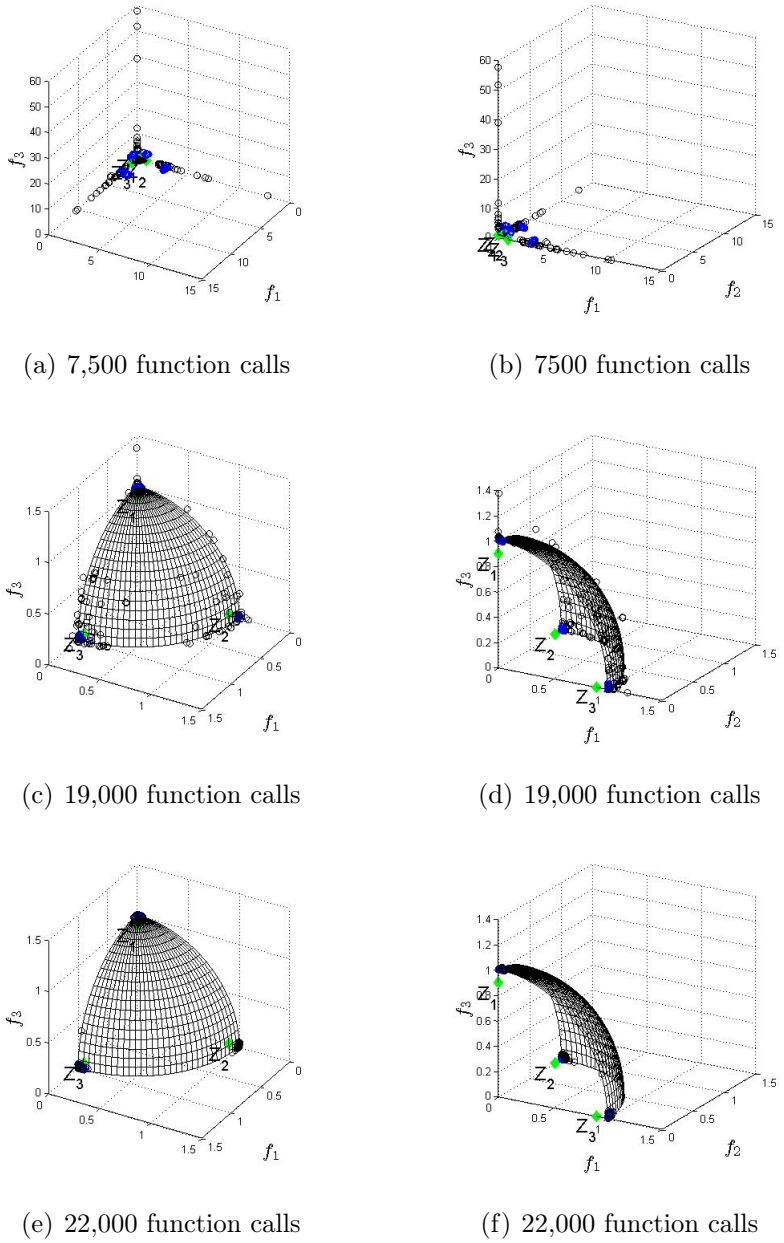
DTLZ3

DTLZ3 is a highly multimodal problem with $3^n - 1$ local Pareto fronts. DTLZ3 is defined as

$$\begin{aligned} f_1(x) &= \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)(1+g(x)) \\ f_2(x) &= \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)(1+g(x)) \\ f_3(x) &= \sin\left(\frac{\pi}{2}x_1\right)(1+g(x)) \\ g(x) &= 100(10+\sum_{i=3}^{n}(x_i-0.5)^2-\cos\left(20\pi(x_i-0.5)\right)) \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i = 1,\ldots,n, \end{aligned} \tag{5.7}$$

where we chose $n = 7$ and $Z = \{(0,0,0.9)^T(0,0.9,0)^T(0.9,0,0)^T\}$. In this problem we can see from Table 5.10 that the performance of RDS-RNSGA-II is only better in

the early or middle stage, although the t-test hints that both algorithms performance is not truly different. This behavior was somehow expected as the LS operator can get stuck in any of the many local Pareto fronts.



(a) 7,500 function calls

(b) 7500 function calls

(c) 19,000 function calls

(d) 19,000 function calls

(e) 22,000 function calls

(f) 22,000 function calls

Figure 5.14: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on DTLZ3.

	Early(5,000)	Middle (10,000)	Later(19,000)
RDS-RNSGA-II	4.062700 **11.356508** (5.231146) **20.234000**	0.126080 **2.123615** (1.603333) **6.393600**	**0.001074** 0.742888 (0.963273) 3.635100
RNSGA-II	**2.435800** 13.495268 (5.021763) 27.018000	**0.117000** 2.130263 (1.603003) 6.417900	0.007968 **0.492401** (0.627942) **2.768300**
RDS-RNSGA-II vs RNSGA-II	1.434311e-01	6.676504e-01	2.978407e-01

Table 5.10: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

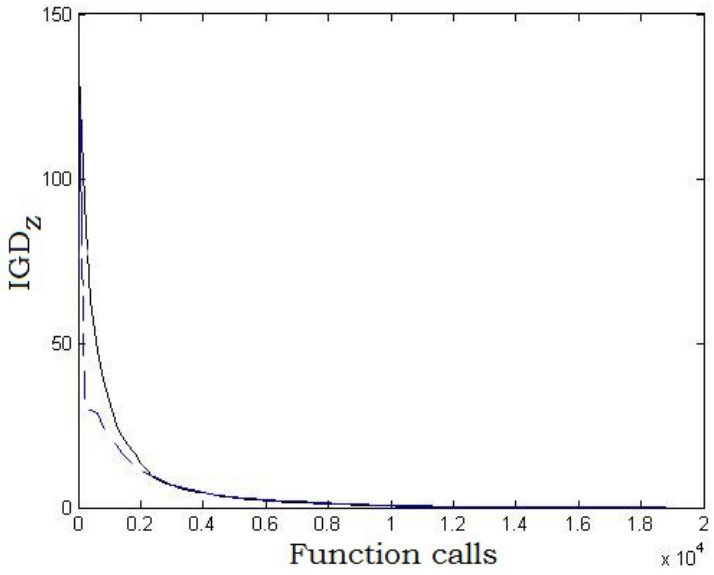


Figure 5.15: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on DTLZ3 for $k = 3$.

5.3 Constrained models

5.3.1 C-DTLZ

For the constrained version of RDS-RNSGA-II we decided to take three test problems from [18]. The chosen C-DTLZ problems introduce one non-linear inequality constraint that modifies the search space in a specific manner.

C1-DTLZ1

Problem C1-DTLZ1 is based on the multimodal DTLZ1 problem where the majority of the search space is made infeasible. DTLZ1 has $11^n - 1$ local Pareto fronts and is defined as

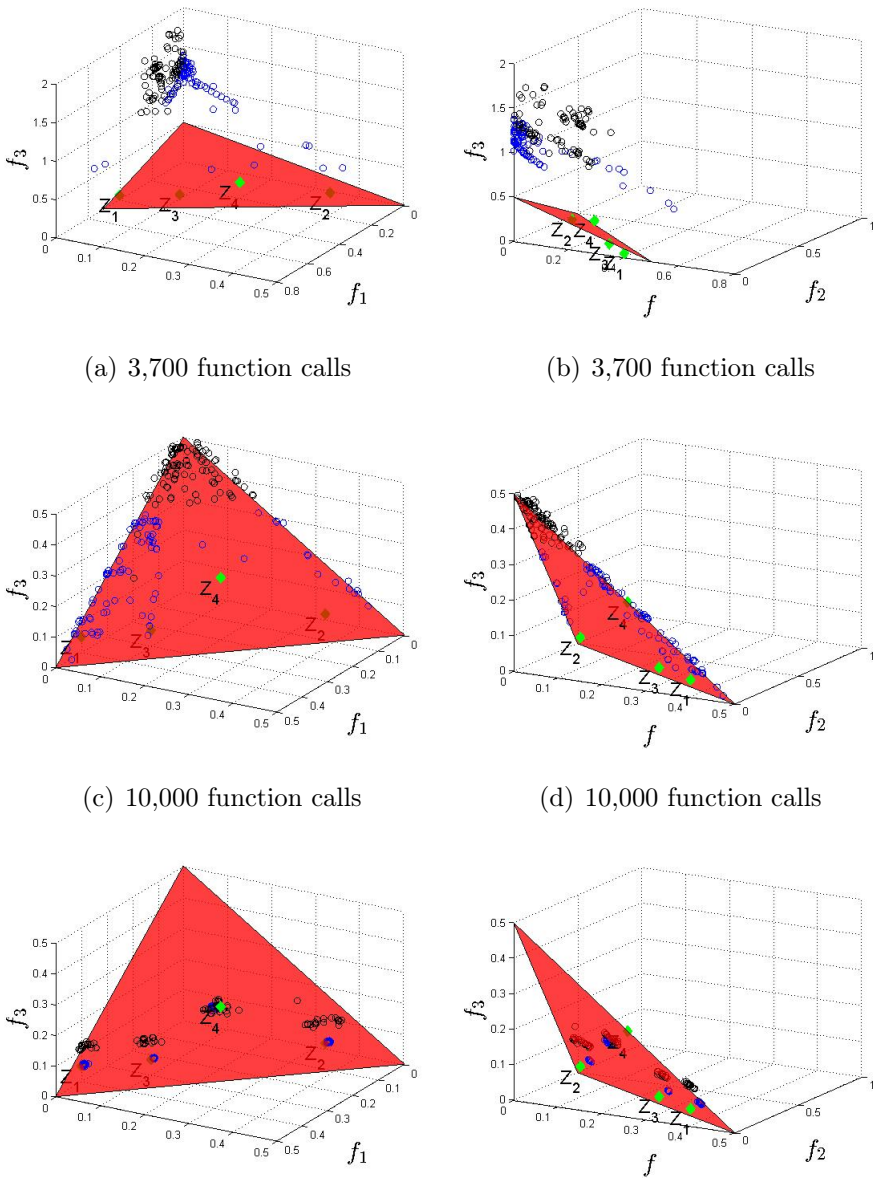
$$\begin{aligned} f_1(x) &= \frac{1}{2}x_1x_2(1+g(x)) \\ f_2(x) &= \frac{1}{2}x_1(1-x_2)(1+g(x)) \\ f_3(x) &= \frac{1}{2}(1-x_1)(1+g(x)) \\ g(x) &= 100(10+\sum_{i=3}^{n}(x_i-0.5)^2 - \cos\left(20\pi(x_i-0.5)\right)) \\ \text{s.t.} \quad & 0 \leq x_i \leq 1 \quad i = 1,\ldots,n. \end{aligned} \tag{5.8}$$

The C1-DTLZ1 inequality constraint is defined as

$$g_1(x) = \frac{f_k(x)}{0.6} + \sum_{i=1}^{k-1}\frac{f_i(x)}{0.5} - 1, \tag{5.9}$$

where we chose $n = 7$ and $Z = \{(0.4, 0, 0.05)^T(0.05, 0.35, 0.05)^T(0.3, 0.1, 0.05)^T(0.2, 0.2, 0.2)^T\}$. Figure 5.16 and Table 5.11 show that for the memetic algorithm a big improvement can be expected even when the population lies on the infeasible region (early stage). However, the LS operator gains the most when solutions lie close to the Pareto front

(later stage). Table 5.11 confirm that the IGD_Z values are better in these two stages for the memetic algorithm.



(a) 3,700 function calls

(b) 3,700 function calls

(c) 10,000 function calls

(d) 10,000 function calls

(e) 15,000 function calls

(f) 15,000 function calls

Figure 5.16: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on C1-DTLZ1.

	Early (2,000)	Middle (5,000)	Later (10,000)
RDS-RNSGA-II	**4.471100** **12.549900** (6.227609) **23.485000**	1.018600 **3.417825** (1.597864) **6.621500**	**0.001753** **0.051518** (0.139535) **0.637570**
RNSGA-II	13.465000 27.557950 (8.708799) 43.109000	**0.746190** 3.980884 (2.048932) 8.420100	0.007376 0.494893 (0.468109) 1.806900
RDS-RNSGA-II vs RNSGA-II	1.049737e-05	3.244156e-01	8.795591e-04

Table 5.11: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

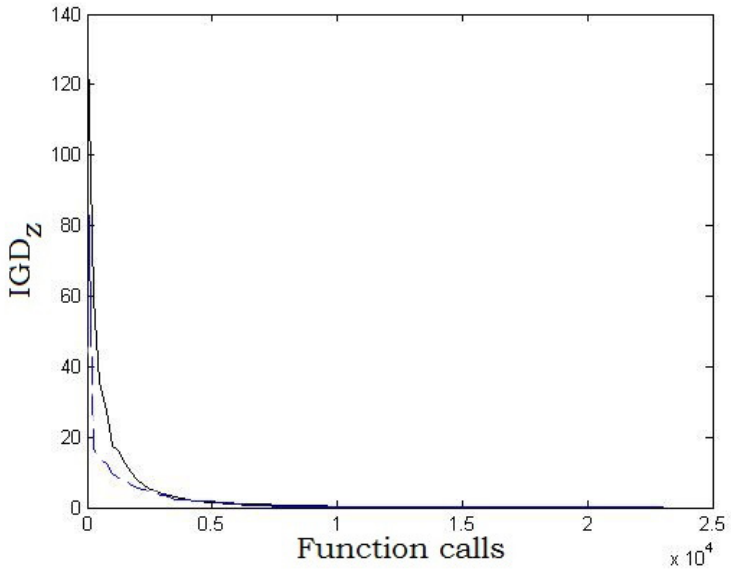


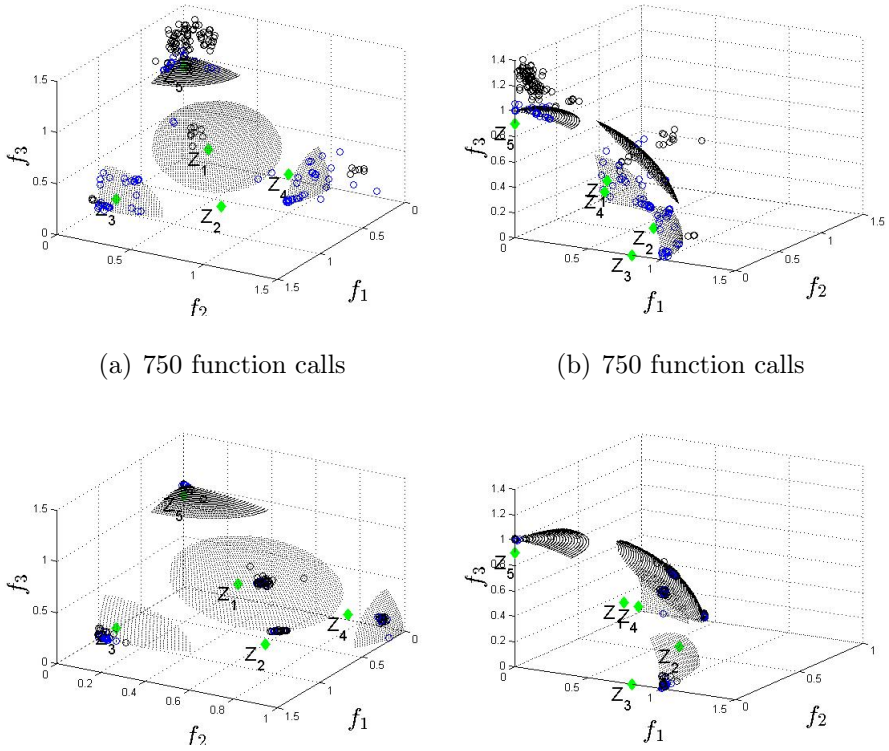
Figure 5.17: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on C1DTLZ1.

C2-DTLZ2

C2-DTLZ2 extends the original DTLZ2 problem by adding one inequality constraint. The feasible objective space lies inside $k+1$ hyper spheres of radius $r = 0.4$ where the inequality constraint is defined as

$$g_1(x) = -\max\left[\max_{i=1}^{k} -\left[(f_i(x)-1)^2 + \sum_{j=1,j\neq i}^{k} f_j(x)^2 - r^2\right], -\left[\sum_{i=1}^{k}(f_i(x) - \frac{1}{\sqrt{k}}^2 - r^2\right]\right]. \tag{5.10}$$

We took $n = 12$ and $Z = \{(0.4, 0.4, 0.4)^T (0.6, 0.6, 0)^T (0.8, 0, 0)^T (0.15; 0.8; 0.15)^T (0, 0, 0.9)^T\}$ for our RPP (see Figure 5.18). Table 5.12 shows that RDS-RNSGA-II clearly outperforms RNSGA-II in all three stages.



(a) 750 function calls (b) 750 function calls

(c) 4,500 function calls (d) 4,500 function calls

Figure 5.18: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on C2-DTLZ2.

	Early (750)	Middle (2,600)	Later (5,000)
RDS-RNSGA-II	**0.043942** **0.114567** (0.082092) 0.433460	**0.002885** **0.013328** (0.011808) **0.041206**	**0.000944** **0.001684** (0.000561) **0.002941**
RNSGA-II	0.139890 0.256823 (0.068890) **0.385260**	0.030398 0.086089 (0.048306) 0.182830	0.003215 0.013915 (0.022673) 0.104700
RDS-RNSGA-II vs RNSGA-II	2.014172e-05	5.263643e-06	2.587012e-02

Table 5.12: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

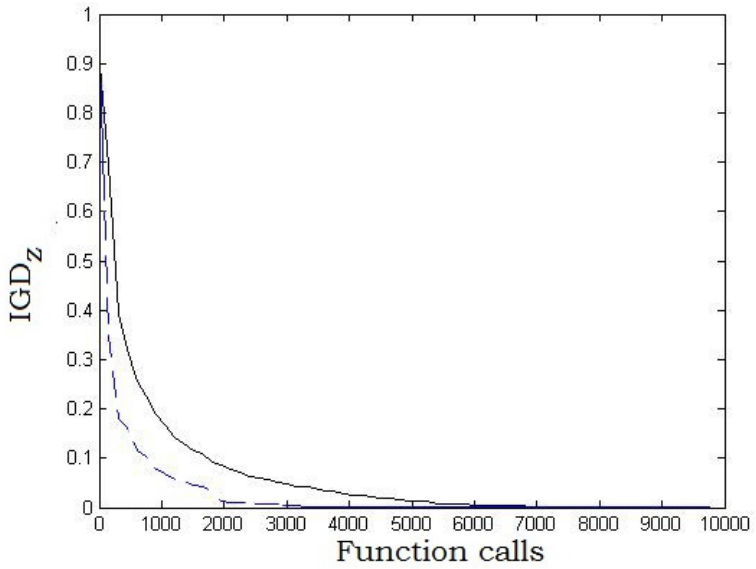


Figure 5.19: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on C2DTLZ2.

C2-CONVEX DTLZ2

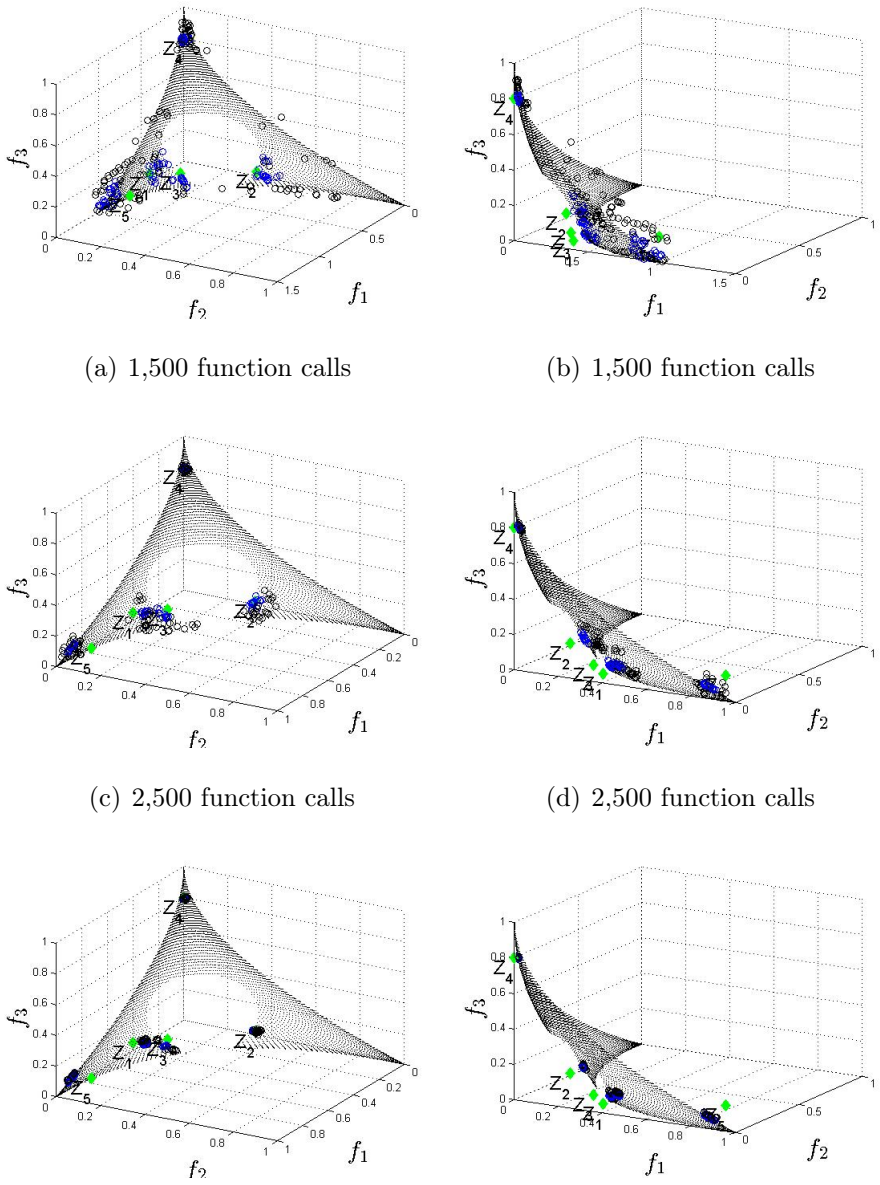
Problem C2-CONVEX DTLZ2 relies on a convex reformulation of DTLZ2. Here, each objective is modified as follows

$$\begin{aligned} f_i(x) &= f_i(x)^4 \quad i = 1, \ldots, k-1 \\ f_k(x) &= f_k(x)^2. \end{aligned} \tag{5.11}$$

C2-CONVEX DTLZ2 has a "hole" of infeasibility in the objective space introduced by a hyper cylinder of radius $r = 0.225$ when $k = 3$ defined by the following inequality constraint

$$\begin{aligned} g_1(x) &= r^2 - \sum_{i=1}^{k} (f_i(x) - \lambda)^2 \\ \lambda &= \frac{1}{k} \sum_{i=1}^{k} f_i(x). \end{aligned} \tag{5.12}$$

The objective space is graphically represented in Figure 5.20 in different stages of RDS-RNSGA-II and RNSGA-II evolution. Table 5.13 shows that such constraints can be easily handled by the memetic algorithm with a significant improvement in early, middle or later stage, where the p-values show that there is a clear advantage of the memetic algorithm over the MOEA.



(a) 1,500 function calls

(b) 1,500 function calls

(c) 2,500 function calls

(d) 2,500 function calls

(e) 4,000 function calls

(f) 4,000 function calls

Figure 5.20: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on C2-CONVEX DTLZ2.

	Early (800)	Middle (1,500)	Later (4,000)
RDS-RNSGA-II	**0.023379** **0.049699** (0.024897) **0.118890**	**0.002878** **0.005857** (0.002477) **0.011516**	**0.000749** **0.001445** (0.000532) **0.003154**
RNSGA-II	0.115840 0.205667 (0.049033) 0.308500	0.025375 0.039196 (0.010318) 0.065317	0.002721 0.004061 (0.000942) 0.005898
RDS-RNSGA-II vs RNSGA-II	7.377884e-12	2.772782e-11	6.848316e-10

Table 5.13: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

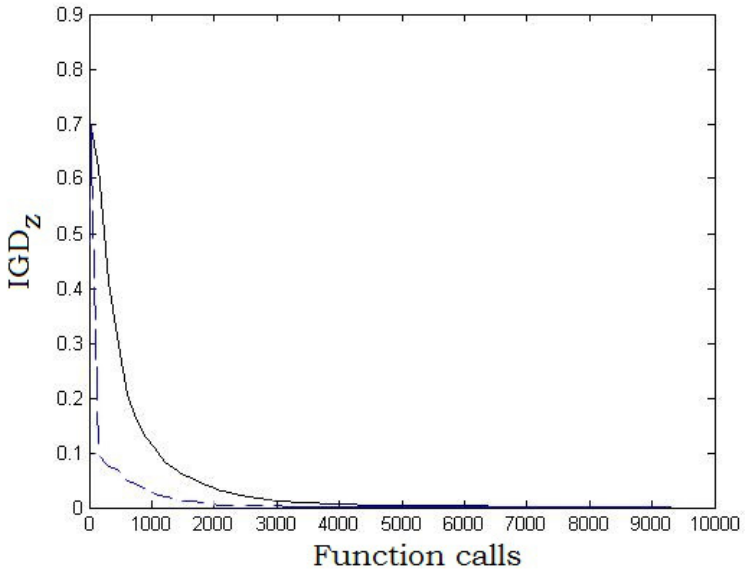


Figure 5.21: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on C2-CONVEX DTLZ2.

5.4 Three problems from practice

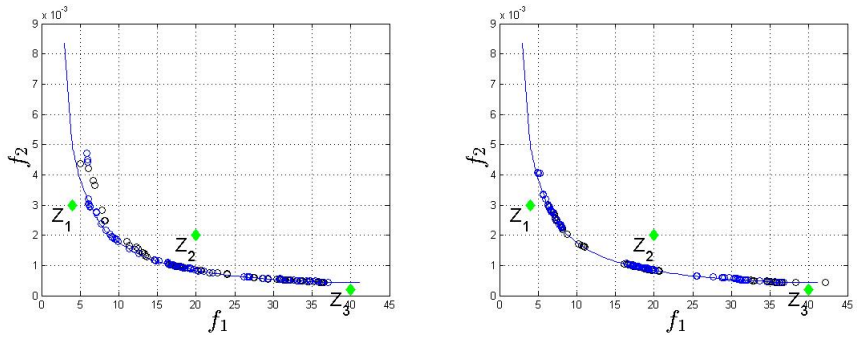
Now we will consider three engineering problems with two, three and five objectives. These are the Welded Beam design [32], Car Side Impact for crash worthiness [18] and Water problem [46], respectively. As for the previous problems, we will concentrate on the speed of convergence for the memetic strategy.

5.4.1 Welded Beam

The Welded Beam problem is a two objective, four variable problem with four non linear constraints. The objectives are the cost of fabrication and the deflection of the welded beam

$$
\begin{aligned}
f_1(x) &= 1.1047x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \\
f_2(x) &= \frac{2.1952}{x_3^2x_4} \\
g_1(x) &= 13600 - \tau \\
g_2(x) &= 30000 - \sigma \\
g_3(x) &= x_4 - x_1 \\
g_4(x) &= P - 6000 \\
\tau &= \sqrt{\tau'^2 + \tau''^2 + x_2\tau'\tau''/\sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}} \\
\tau' &= \frac{6000}{\sqrt{2x_1x_2}} \\
\tau'' &= \frac{6000(14 + 0.5x_2\sqrt{0.25(x_2^2 + (x_1 + x_3)^2)}}{2(0.707x_1x_2(x_2^2/12 + 0.25(x_1 + x_3)^2))} \\
\sigma &= \frac{504000}{x_3^2x_4} \\
P &= 64746.022(1 - 0.0282346x_3)x_3x_4^2 \\
\text{s.t.} \quad & 0.125 \leq x_1, x_4 \leq 5 \quad 0.1 \leq x_2, x_3 \leq 10.
\end{aligned}
\tag{5.13}
$$

Figures 5.22 and 5.23 show that both algorithms have an expected similar performance with good values in the early stage already, since the complexity and scale of objectives and constraints can be easily handled by RNSGA-II. That is indeed the case as the statistical significance test shows in Table 5.14.



(a) 1,600 function calls (b) 2,550 function calls

Figure 5.22: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on the Welded Beam problem.

	Early (950)	Middle (1,600)	Later (2,550)
RDS-RNSGA-II	0.006904 0.023025 (0.009056) 0.043220	**0.004942** **0.015464** (0.006019) 0.028164	**0.003065** **0.011100** (0.006286) 0.027509
RNSGA-II	**0.044925** **0.022974** (0.010723) **0.007681**	0.038428 0.017111 (0.009857) **0.002882**	0.032565 0.011150 (0.008095) **0.001395**
RDS-RNSGA-II vs RNSGA-II	9.834391e-01	4.097445e-01	9.892144e-01

Table 5.14: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

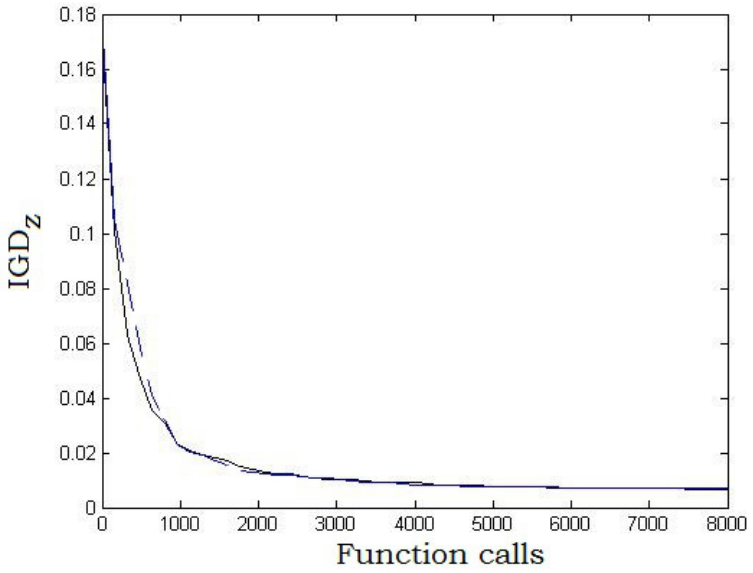


Figure 5.23: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on bi objective Welded Beam problem.

5.4.2 Car Side Impact

The Car Side Impact problem is a three objective MOP where the objectives are (i) to minimize the weight of car, (ii) the pubic force experienced by a passenger and (iii) the average velocity of the V-Pillar responsible for withstanding the impact load. The problem has three objectives, seven design variables and is constrained by ten inequality constraints. The problem is given by

$$f_1(x) = 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 0.00001x_6 + 2.73x_7$$

$$f_2(x) = 4.72 - 0.5x_4 - 0.19x_2x_3$$

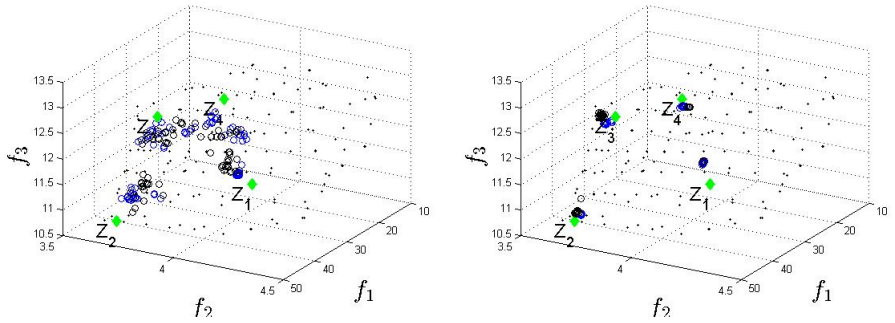
$$f_3(x) = 0.5(10.58 - 0.674x_1x_2 - 0.67275x_2 + 16.45 - 0.489x_3x_7 - 0.843x_5x_6)$$

$$g_1(x) = -1 + 1.16 - 0.3717x_2x_4 - 0.0092928x_3$$

$$g_2(x) = -0.32 + 0.261 - 0.0159x_1x_2 - 0.06486x_1 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0154464x_6$$

$$
\begin{aligned}
g_3(x) &= -0.32 + 0.214 + 0.00817x_5 - 0.045195x_1 - 0.0135168x_1 + 0.03099x_2x_6 - 0.018x_2x_7 \\
&\quad + 0.007176x_3 + 0.023232x_3 - 0.00364x_5x_6 - 0.018x_2^2 \\
g_4(x) &= -0.32 + 0.74 - 0.61x_2 - 0.031296x_3 - 0.031872x_7 + 0.227x_2^2 \\
g_5(x) &= -32 + 28.98 + 3.818x_3 - 4.2x_1x_2 + 1.27296x_6 - 2.68065x_7 \\
g_6(x) &= -32 + 33.86 + 2.95x_3 - 5.057x_1x_2 - 3.795x_2 - 3.4431x_7 + 1.45728 \\
g_7(x) &= -32 + 46.36 - 9.9x_2 - 4.4505x_1 \\
g_8(x) &= -4 + 4.72 - 0.5x_4 - 0.19x_2x_3 \\
g_9(x) &= -9.9 + 10.58 - 0.674x_1x_2 - 0.67275x_2 \\
g_{10}(x) &= -15.7 + 16.45 - 0.489x_3x_7 - 0.843x_5x_6 \\
\text{s.t.} \quad & 0.5 \leq x_1, x_3, x_4 \leq 1.5 \quad 0.45 \leq x_2 \leq 1.35 \quad 0.875 \leq x_5 \leq 2.625 \quad 0.4 \leq x_6, x_7 \leq 1.2
\end{aligned}
\tag{5.14}
$$

Figure 5.25 and Table 5.15 show that RDS-RNSGA-II has better best and median IGD_Z values in all three stages. Although the p value is not as big as in the academic problems, there is indeed a difference in the performance of the memetic strategy.



(a) 480 function calls (b) 4,500 function calls

Figure 5.24: Numerical results of R-NGSA-II (black) and RDS-RNGSA-II (blue) on the Car Side Impact Problem.

	Early (480)	Middle (1,600)	Later (4,500)
RDS-RNSGA-II	**0.042154** **0.072363** (0.013766) 0.098701	**0.030761** **0.040208** (0.006474) 0.055860	**0.029540** **0.032780** (0.002489) 0.040981
RNSGA-II	0.112420 0.086927 (0.012254) **0.063910**	0.059682 0.045197 (0.005877) **0.035668**	0.039558 0.033920 (0.002005) **0.030374**
RDS-RNSGA-II vs RNSGA-II	1.725967e-05	1.808759e-03	7.714379e-02

Table 5.15: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

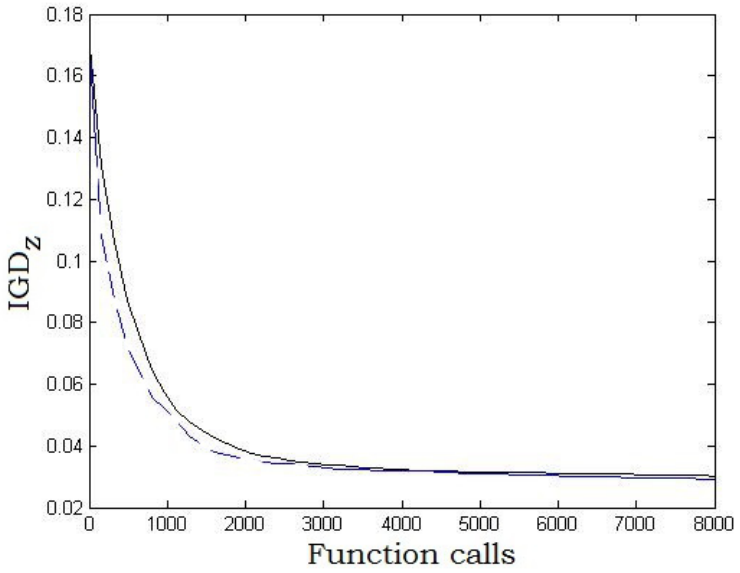


Figure 5.25: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on three objective Car Side Impact problem.

5.4.3 Water Problem

The Water Problem is a three variable, five objective problem taken from [46]. The mathematical formulation of the problem is as follows

$$
\begin{aligned}
f_1(x) &= 106780.37(x_2 + x_3) + 61704.67 \\
f_2(x) &= 3000x_1 \\
f_3(x) &= 305700 * 2289\frac{x_2}{(0.06 * 2289)^{0.65}} \\
f_4(x) &= 250 * 2289e^{-39.75x_2+9.9x_3+2.74} \\
f_5(x) &= 25(\frac{1.39}{x_1x_2} + 4940x_3 - 80) \\
g_1(x) &= -1 + \frac{0.00139}{x_1x_2} + 4.94x_3 - 0.08 \\
g_2(x) &= -1 + \frac{0.000306}{x_1x_2} + 1.082x_3 - 0.0986 \\
g_3(x) &= -50000 + \frac{12.307}{x_1x_2} + 49408.24x_3 + 4051.02 \\
g_4(x) &= -16000 + \frac{2.098}{x_1x_2} + 8046.33x_3 - 696.71 \\
g_5(x) &= -10000 + \frac{2.138}{x_1x_2} + 7883.39x_3 - 705.04 \\
g_6(x) &= -2000 + \frac{0.417}{x_1x_2} + 1721.26x_3 - 136.54 \\
g_7(x) &= -550 + \frac{0.164}{x_1x_2} + 631.13x_3 - 54.84 \\
\text{s.t.} \quad & 0.01 \leq x_1 \leq 0.45 \quad 0.01 \leq x_2, x_3 \leq 0.1.
\end{aligned}
\tag{5.15}
$$

Table 5.16 shows that in the later stage the memetic strategy has the best IGD_Z value in the best, median and worst case which translates into higher accuracy. This was not the case in the early stage since the original MOEA had a better average performance. However, as in the Welded Beam problem, the statistical significance determined that both algorithms have a similar performance.

	Early (400)	Middle (1,400)	Later (3,000)
RDS-RNSGA-II	**0.006328** 0.024149 (0.014080) 0.063493	**0.000204** **0.005129** (0.003049) 0.011112	**0.000204** **0.002693** (0.001930) **0.006714**
RNSGA-II	0.055148 **0.021734** (0.012671) **0.055148**	0.010949 0.005327 (0.002654) **0.010949**	0.006720 0.003221 (0.001472) 0.006720
RDS-RNSGA-II vs RNSGA-II	5.275097e-01	7.738408e-01	2.121774e-01

Table 5.16: Best, median and worst IGD_Z for RDS-RNSGA-II and RNSGA-II at three different stages. The last row are the results for statistical significance based on the t-test using a 95% confidence interval.

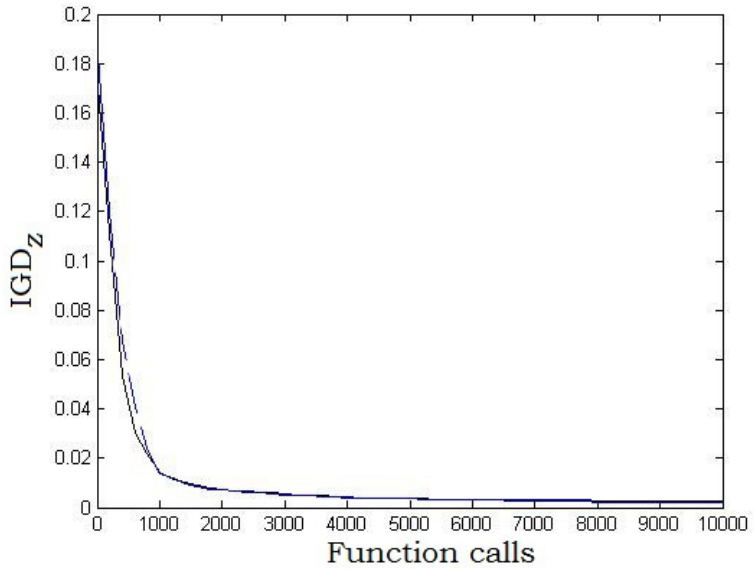


Figure 5.26: IGD_Z against function calls RNGSA-II (straight line) and RDS-RNGSA-II (dashed line) on five objective Water Problem.

6 | Conclusions and Future Work

In this work we have discussed and developed an a priori approach for solving MOPs using RPs. Firstly, we developed a version of DS for tackling such problems called RDS. Among the highlights of RDS are the following:

- It can shoot towards the RP from any direction of the objective space in the greedy direction. The direction is inherently updated at each iteration of RDS.
- It can reach a feasible RP with local quadratic convergence.
- Furthermore, given a feasible RP, RDS is able to find better solutions by "shooting" in a dominating direction towards the Pareto front.
- For infeasible RPs, in order to reach the closest solution to the RP, a continuation phase along the Pareto front was proposed.
- A new step size strategy was proposed for the given context. Classical DS step size strategies can also be implemented.

A neighborhood exploration procedure was also discussed at this point. This new algorithm, called Neighborhood exploration, was able to find new local solutions in the Pareto front close to a given Pareto point (for example, the solution to a RPP) in a structured manner. A major advantage of the proposed algorithm is its ability to find different layers of neighborhoods, each one defined by user specified step size, making it ideal for an a posteriori treatment of the RPP. Furthermore, only the Jacobian matrix for the initial Pareto point is needed, making it a very efficient algorithm.

The comprehensive RDS was extended to handle box constraints with a gradient projection method which proved to be effective by Salinas et al., likewise, RDS was made gradient free by taking into account the considerations of the DDS.

A first attempt to hybridize both the gradient based and the gradient free RDS with RNSGA-II for box constrained problems was proposed and tested against its original counterpart on several unconstrained and box constrained problems where a significant improvement was achieved. For this memetic algorithm, we proposed a framework for selecting, improving and replacing solutions within a population which proved to be efficient on the many benchmark functions tested. Worth noticing, is that this hybrid version of RDS was designed to be independent of the base algorithm, in this case, RNSGA-II. Hence, it can be used by any other evolutionary RP-based algorithm. We investigated possible performance indicators for assessing the speed and convergence of our algorithm (that is, before the final archive is ready) and took a variant of the IGD metric. This version, which we called IGD_Z, measures convergence towards the solution of the RPP given by each RP.

The RDS was able to handle non-linear constraints by using once again the gradient projection method. In addition, we have included constraint handling mechanisms for the RDS as local search engine in order to agree with RNSGA-II's constraint handling strategy. Proper modifications were made for improving always feasible over infeasible solutions and for archiving only feasible solutions for IGD_Z to take.

RDS-RNSGA-II was able to outperform significantly RNSGA-II in all three kinds of academic problems (unconstrained, box constrained and constrained) at different stages of their evolution (early, middle and later) in terms of IGD_Z. Interestingly enough, pictures of each problem show a better spread of solutions around each RP, however, this property could not be measured before the final archive is ready.

Finally, we took three constrained engineering problems with $2, 3$ and 5 objectives. For these three problems, the performance of RDS-RNSGA-II and RNSGA-II was not significantly different. For such complex models, the considerations for the memetic

algorithm on constrained problems need to be improved.

Hence, we can conclude that the local search engine proposed in combination with DS previous work make RDS a competitive local search engine for RPPs, backed up by the experiments presented in the previous chapter. Furthermore, the hybrid algorithm is able to yield better and more accurate solutions with a statistical significance in most test cases.

During the development of this thesis:

- The work was presented in the international conference *EVOLVE 2014 - A Bridge Between Probability, Set-Oriented Numerics, and Evolutionary Computation*, held in Beijing, China from July 1st to the 4th. The title of the work is *A Memetic Variant of RNSGA-II for Reference Point Problems* and its authors are Jesús Alejandro Hernández Mejía, Oliver Schütze and Kalyanmoy Deb. The considerations of RDS were presented in this work along with a first approach to its hybridization with RNSGA-II for unconstrained and box constrained academic problems. The RDS-RNSGA-II was compared to RNSGA-II in terms of IGD_Z outperforming it.

6.1 Future work

To improve the existing work there exist many options such as:

- The inclusion of a "Pareto explorer" such as the Neighborhood exploration presented in Chapter 3. Given a Pareto point, for instance, the closest to a supplied RP, further optimal solutions can be obtained in the neighborhood that can be interesting for DMs. Although we presented very nice images of this neighborhood exploration feature for three dimensional problems, this calls for an extended metric of IGD_Z that can measure the desired spread of solutions within the Pareto front.

- Different constraint handling techniques for the local search engine. The numerical results presented in the previous chapter show that RDS' performance is much better on unconstrained (box constrained) problems.
- Further experiments for highly multimodal problems such as DTLZ3.
- Comparison with other RP based MOEAs such as RMEAD.
- To address RPPs by achievement scalarizing functions (ASF) other than the one taken in this work.
- To test the memetic strategy on several many objective problems since it is expected that in this kind of problems, RDS-RNSGA-II can overwhelmingly outperform RP based MOEAs, as it was the case for DTLZ2. Benchmark and real world functions with different properties should be considered for this purpose.
- To further hybridize RDS and RDDS in order to reduce the use of gradients in the memetic strategy.

Bibliography

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

[2] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 2006.

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

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

[5] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1999.

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

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

[8] J.P. Ignizio. *Goal programming and extensions*. Lexington Books, 1976.

[9] Andrzej P. Wierzbicki. The use of reference objectives in multiobjective optimization. In Günter Fandel and Tomas Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 468–486. Springer Berlin Heidelberg, 1980.

[10] O. Schütze, A. Lara, and C. A. Coello Coello. The directed search method for unconstrained multi-objective optimization problems. In *Proceedings of the EVOLVE – A Bridge Between Probability, Set Oriented Numerics, and Evolutionary Computation*, 2011.

[11] Karthik Sindhya, Kalyanmoy Deb, and Kaisa Miettinen. A local search based evolutionary multi-objective optimization approach for fast and accurate convergence. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature – PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 815–824. Springer Berlin Heidelberg, 2008.

[12] Jih-Yiing Lin and Ying ping Chen. When and what kind of memetic algorithms perform well. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[13] S. Bechikh, L. Ben Said, and K. Ghedira. Estimating nadir point in multiobjective optimization using mobile reference points. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–9, July 2010.

[14] Karthik Sindhya, Kalyanmoy Deb, and Kaisa Miettinen. A local search based evolutionary multi-objective optimization approach for fast and accurate convergence. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature, PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 815–824. Springer Berlin Heidelberg, 2008.

[15] Karthik Sindhya, AnaBelen Ruiz, and Kaisa Miettinen. A preference based interactive evolutionary algorithm for multi-objective optimization: PIE. In RicardoH.C. Takahashi, Kalyanmoy Deb, ElizabethF. Wanner, and Salvatore Greco, editors, *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 212–225. Springer Berlin Heidelberg, 2011.

[16] A. Lara, G. Sanchez, Carlos A. Coello Coello, and O. Schütze. HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 14(1):112–132, Feb 2010.

[17] K. Deb and H. Jain. An Evolutionary Many-objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems with Box Constraints. *Evolutionary Computation, IEEE Transactions on*, 18(4):577–601, Aug 2014.

[18] H. Jain and K. Deb. An Evolutionary Many-objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. *Evolutionary Computation, IEEE Transactions on*, 18(4):602–622, Aug 2014.

[19] V. Pareto. Cours d'economie politique, volume I and II. f. rouge, lausanne. 1896.

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

[21] H. Kuhn and A. Tucker. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, page 481–492. University of California Press, Berkeley and Los Angeles, 1951.

[22] W. E. Karush. *Minima of functions of several variables with inequalities as side conditions*. PhD thesis, Dept. Math., Univ. Chicago, 1939.

[23] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[24] L.A. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8:59–60, 1963.

[25] J.L. Cochrane and M. Zeleny. *Multiple Criteria Decision Making*. University of South Carolina Press, Columbia, 1973.

[26] A. Lara, S. Alvarado, S. Salomon, G. Avigad, C. A. Coello Coello, and O. Schütze. The gradient free directed search method as local search within multi-objective evolutionary algorithms. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation (EVOLVE II)*, pages 153–168, 2013.

[27] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[28] John H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, July 1962.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, Apr 2002.

[30] Ram Bhusan Agrawal and Kalyanmoy Deb. Simulated binary crossover for continuous search space. Technical report, 1994.

[31] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, Dec 2007.

[32] Kalyanmoy Deb, J. Sundar, Udaya Bhaskara Rao N, and Shamik Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms.

In *International Journal of Computational Intelligence Research*, pages 635–642. Springer-Verlag, 2006.

[33] A Mohammadi, M.N. Omidvar, and Xiaodong Li. Reference point based multi-objective optimization through decomposition. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[34] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[35] D. A. Van Veldhuizen and G. B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.

[36] Carlos A. Coello Coello and Nareli Cruz Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190, 2005.

[37] O. Schütze, X. Esquivel, A. Lara, and Carlos A. Coello Coello. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 16(4):504–522, Aug 2012.

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

[39] Eduardo Salinas Márquez. *A New Gradient Free Continuation Method for the Treatment of Constrained Multiobjective Optimization Problems*. PhD thesis, Computer Science Department, CINVESTAV, 2013.

[40] Erick Ignacio Mejía Estrada. *The Directed Search Method for Constrained Multi-Objective Optimization Problems*. PhD thesis, Computer Science Department, CINVESTAV, 2012.

[41] J.B. Rosen. The gradient projection method for nonlinear programming part i linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8:181–217, 1960.

[42] Hisao Ishibuchi, Tadashi Yoshida, and Tadahiko Murata. Balance between genetic search and local search in hybrid evolutionary multi-criterion optimization algorithms. *IEEE Trans. on Evolutionary Computation*, 7:204–223, 2002.

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

[44] M. Köppen and K. Yoshida. Substitute distance assignment in NSGA-II for handling many-objective optimization problems. In S. Obayashi et al., editor, *Evolutionary Multi-Criterion Optimization, 4th International Conference (EMO 2007)*, pages 727–741, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[45] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

[46] T Ray, K Tai, and C Seow. An evolutionary algorithm for multiobjective optimization. *Eng. Optim*, 33(3):399–424, 2001.