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

Development of techniques to improve computational efficiency in multi-objective evolutionary algorithms

Submitted by:

Luis Vicente Santana Quintero

as the fulfillment of the
requirement for the degree of

Doctor of Science

Specialization in:
Electrical Engineering

Option:
Computer Science

Advisor:
Dr. Carlos A. Coello Coello

México, D. F. November 2008

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

Desarrollo de técnicas para mejorar la eficiencia computacional de algoritmos evolutivos multiobjetivo

Tesis que presenta:

Luis Vicente Santana Quintero

Para obtener el grado de

Doctor en Ciencias

En la especialidad de
Ingeniería Eléctrica

Opción:
Computación

Director de la Tesis:
Dr. Carlos A. Coello Coello

México, D. F. Noviembre 2008

Dedicada a mis padres. . .

People choose the paths that gain
them the greatest reward for the least
amount of effort. That's a law of nature.

Las personas eligen los caminos que les
represente la mayor recompensa por la menor
cantidad de esfuerzo. Esa es la ley de la vida.

Agradecimientos

Gracias a mis padres (Luis Vicente y Guadalupe), hermanas (Talía y Aglae) y seres queridos por su apoyo incondicional durante el desarrollo de esta tesis.

Gracias a mi asesor, el Dr. Carlos Coello Coello por bien aconsejarme en todo momento, además de transmitirme valiosos conocimientos que seguro me serán útiles por el resto de mi vida.

Gracias a ti, Karina por tu cariño y apoyo, eres una persona muy valiosa para mi, y espero tenerte a mi lado por mucho tiempo para apoyarnos en todo.

Agradezco a mis compañeros, por hacer de mi estancia en el CINVESTAV un lugar más ameno y alegre, tanto... que llegase a disfrutarlo. En orden alfabético: Adriana, Antonio, Alfredo, Aracely, don Benja, Efrén, Gabriela, Gregorio, Israel, Mario, Mario Augusto, Margarita, Mireya, Noel, Ricardo, entre otros.

Agradezco a todo el personal del departamento de Computación: Sofia, Felipa y Flor, por apoyarme siempre en cuanto a la logística de documentos, inscripciones, boletas, congresos, etc.

Gracias al CINVESTAV por su apoyo al permitirme cursar todos los cursos necesarios para lograr este trabajo de tesis y facilitarme las instalaciones en las que se desarrolló el mismo.

Gracias al CONACyT por la beca proporcionada durante mi estancia en el programa de doctorado en el CINVESTAV. Este trabajo de tesis se derivó del proyecto CONACyT titulado "Técnicas Avanzadas de Optimización Evolutiva Multiobjetivo" (Ref. 45683-Y) cuyo responsable es el Dr. Carlos A. Coello Coello.

Abstract

In this thesis, we present different techniques that aim to improve the efficiency of the multi-objective evolutionary algorithms. By efficiency, we refer to reducing the number of fitness function evaluations performed by a multi-objective evolutionary algorithm when solving problems of moderate dimensionality (up to 30 decision variables). When solving a multi-objective problem, it is very important to do a good *exploration* at the beginning of the search and find quickly promising solutions. After that, an *exploitation* of those good solutions is needed to accelerate the convergence and finally get to the true Pareto front of the multi-objective problem. Additionally, a mechanism to keep the nondominated solutions is normally used to retain a *well-distributed* set of solutions along the Pareto front. We present here mechanisms that tackle each of these problems: exploration, exploitation and distribution.

First, we propose a new multi-objective evolutionary algorithm (MOEA) based on differential evolution and rough sets theory to show how the hybridization of a fast multi-objective evolutionary algorithm and a local search method based on the use of rough sets, is an efficient alternative to obtain a robust algorithm able to solve difficult unconstrained and constrained multi-objective optimization problems. The proposed approach adopts an external archive in order to retain the nondominated solutions found during the evolutionary process. Additionally, the approach also incorporates the concept of $pa\epsilon$-dominance to get a good distribution of the solutions retained. The main idea of the approach is to use differential evolution (DE) as our main search engine, trying to translate its good convergence properties exhibited in single-objective optimization to the multi-objective case. Rough sets theory is adopted in a second stage of the search in order to improve the spread of the nondominated solutions that have been found so far. Our hybrid approach is validated using standard test functions and performance measures commonly adopted in the specialized literature. Our results are compared with respect to two approaches that are representative of the state-of-the-art in the area: NSGA-II, and SPEA2.

Secondly, we propose an approach in which a multi-objective particle swarm optimizer (MOPSO) is coupled to a surrogate method in order to explore the search space in an efficient manner. In order to determine which is the most appropriate surrogate method to be adopted, a small comparative study among three surrogate methods is conducted: an artificial neural network (ANN), a radial basis function (RBF) and a support vector machine (SVM). The best performer in this comparative study was the support vector machine, which was, therefore, adopted for our approach. Also, we perform a comparative study among different leader selection schemes in a MOPSO, in order to determine the most appropriate approach to be adopted for solving the sort of problems of our interest. However, our results indicated that the spread of solutions achieved by our surrogate-based MOEA was poor. Thus, we decided to introduce a second phase to the algorithm in which it is hybridized with the rough sets in order to improve the spread of solutions and reach the true Pareto front. In this case, the rough sets act as a local search approach which is able to generate solutions in the neighborhood of the nondominated solutions previously generated. We show that our proposed hybrid approach only requires 2,000 fitness function evaluations in order to solve test problems with up to 30 decision variables.

Finally, we propose a mechanism that can be seen as a variant of ϵ-dominance, which we call Pareto-adaptive ϵ-dominance ($pa\epsilon$-dominance). Our proposed approach overcomes the main limitations of ϵ-dominance: the loss of several nondominated solutions from the hypergrid adopted in the archive because of the way in which solutions are selected within each box.

Resumen

En esta tesis, se presentan diferentes técnicas que tienen como objetivo el mejorar la eficiencia de los algoritmos evolutivos para problemas de optimización multi-objetivo. Por eficiencia, nos referimos a reducir el número de evaluaciones de la función objetivo realizadas por el algoritmo evolutivo multi-objetivo al resolver problemas de dimensionalidad moderada (de hasta 30 variables de decisión). Cuando tratamos de resolver un problema multi-objetivo, es muy importante hacer una buena exploración al inicio y encontrar rápidamente soluciones prometedoras. Después, se necesita una explotación de las buenas soluciones para acelerar la convergencia y, finalmente, encontrar el frente de Pareto verdadero del problema. Además, se necesita un mecanismo para retener a las soluciones no dominadas con una buena distribución a lo largo del frente de Pareto. En esta tesis, lidiamos con cada uno de estos problemas: la exploración, la explotación y la distribución de las soluciones.

En primer lugar, se propone un nuevo algoritmo evolutivo multi-objetivo basado en evolución diferencial y la teoría de los conjuntos borrosos, para mostrar cómo es que la hibridación de un algoritmo evolutivo con una alta velocidad de convergencia como lo es la evolución diferencial y un optimizador local basado en la teoría de los conjuntos borrosos, pueden ser una alternativa eficiente para construir un algoritmo robusto capaz de resolver problemas multi-objetivo con y sin restricciones. El algoritmo propuesto adopta el uso de un archivo externo con el fin de retener a las soluciones no dominadas encontradas durante el proceso evolutivo. Además, se utiliza el concepto de dominancia-ϵ Pareto-adaptativa para obtener una buena distribución de las mejores soluciones. La idea principal del algoritmo es la de utilizar la evolución diferencial como el motor de búsqueda, intentando extender las buenas propiedades de convergencia mostradas en optimización global para el caso multi-objetivo. Después, la teoría de los conjuntos borrosos se adopta en una segunda etapa, con el fin de generar más soluciones no dominadas en la vecindad de aquellas soluciones no dominadas que produjo la primera etapa. Nuestro algoritmo híbrido es validado utilizando diferentes funciones de prueba estándar y medidas de desempeño que son utilizadas en la literatura especializada. Nuestros resultados finales son comparados con respecto a otros dos algoritmos que son representativas del estado del arte: NSGA-II, y SPEA2.

En segundo lugar, proponemos un algoritmo multi-objetivo usando el algoritmo basado en cúmulos de partículas para problemas multi-objetivo (MOPSO, por sus siglas en inglés), acoplado a un método de aproximación para explorar el espacio de búsqueda de manera eficiente. Con el fin de determinar el mejor método de aproximación, se llevo a cabo un pequeño estudio comparativo entre tres métodos: una red neuronal artificial (ANN), una red con funciones de base (RBF), y una máquina de vectores de soporte (SVM). El que mejor desempeño obtuvo de este estudio fueron las máquinas de vectores de soporte, que fue por tanto, la opción adoptada en nuestro algoritmo final. Además, se realizó un estudio comparativo entre los diferentes esquemas de selección del líder en el algoritmo MOPSO, con tal de encontrar el tipo de selección más adecuada. Sin embargo, los resultados de este algoritmo indican que las soluciones alcanzadas en un inicio no fueron lo eficientemente buenas en cuanto a su distribución, aunque sí respecto a su cercanía al frente de Pareto verdadero. En virtud de ello, decidimos agregar una segunda fase al algoritmo hibridizándolo con los conjuntos borrosos, que actúan como optimizadores locales con el objetivo de mejorar esa distribución de soluciones y producir una mejor aproximiación del frente de Pareto verdadero. Se demuestra que nuestro algoritmo híbrido sólo requiere de 2000 evaluaciones de la función de aptitud para resolver problemas de prueba con hasta 30 variables.

Por último, proponemos un mecanismo que puede ser visto como una variante de la dominancia-ϵ, al cual llamamos dominancia-ϵ Pareto-adaptativa (paϵ-dominancia). Nuestra propuesta intenta superar la principal limitación de la dominancia-ϵ, es decir, la pérdida de soluciones no dominadas que se encuentran en los extremos de la hiper-malla construida por el archivo externo, debido a la forma en que ésta selecciona a las soluciones dentro de cada caja.

Contents

List of Figures

List of Tables

1 Introduction

In many disciplines, optimization problems have two or more objectives, which are normally in conflict with one another, and that we wish to optimize simultaneously. These problems are called "multi-objective", and their solution involves the design of different algorithms than when dealing with single-objective optimization problems. Multi-objective problems, in fact, normally give rise not to one, but to a set of solutions (called Pareto optimal set) which, in the absence of any further information, are all equally good.

Evolutionary algorithms have been very popular for solving multi-objective optimization problems [26, 35], mainly because of their ease of use, and their wide applicability. However, multi-objective evolutionary algorithms (MOEAs) tend to consume a great amount of resources in evaluating the objective function many times, in order to achieve a reasonably good approximation of the Pareto front, even when dealing with benchmark problems of low dimensionality. This is a major concern when attempting to use MOEAs for real-world applications, since we can normally afford only a fairly limited number of fitness function evaluations in such cases.

Despite these concerns, little efforts have been reported in the literature to reduce the computational cost of MOEAs, and several of them only focus on algorithmic complexity (see for example [79]), in which little improvement can be done because of the theoretical bounds related to nondominance checking [93]. It has been until relatively recently, that researchers have developed techniques to achieve a reduction of fitness function evaluations by exploiting knowledge acquired during the search [86]. This knowledge can, for instance, be used to adapt the recombination and mutation operators in order to sample offspring in promising areas of the search space (as is done through the use of cultural algorithms [132]). Knowledge of past evaluations can also be used to build an empirical model that approximates the fitness function to optimize, approximation model which is much less expensive to evaluate than the real fitness function. This approximation can then be used to predict promising new solutions at a smaller evaluation cost than that of the original problem [86, 81].

In the application of evolutionary algorithms (EAs) to engineering design domains, a large number of objective evaluations are normally required in order to obtain a good approximation of the Pareto optimal set. Moreover, the search space can be complex, having many constraints and a small feasible region. Additionally, determining the fitness of each solution may involve the use of a simulator that takes up a significant amount of CPU time.

The high number of fitness evaluations performed by EAs is often computationally expensive, time-consuming or otherwise problematic in many real-world applications. Especially in the following cases, a computationally efficient approximation of the original fitness function to reduce is necessary:

- if the evaluation of the fitness function is computationally expensive.
- if the fitness function can not be defined in an algebraic form (i.e., when using a simulator).
- if additional physical devices must be used and they require human interaction.
- if parallelism is not allowed.
- if the total number of evaluation of the fitness function is limited due to financial constraints.

For EAs, various approaches have been developed to reduce this cost by exploiting knowledge of the history of evaluated points. This knowledge can, for instance, be used to adapt the recombination and mutation operators in order to sample offspring in promising areas. Knowledge of past evaluations can also be used to build an empirical model that approximates the fitness function to optimize. The approximation is then used to predict promising new solutions at a smaller evaluation cost than that of the original problem.

The remainder of this document is organized as follows. In Chapter 2 we discuss the main concepts from multi-objective optimization necessary to make this document self-contained. We also describe the different MOEAs that have been reported in recent years, as well as a description of the performance measures and test problems that we used to validate our proposed approaches in Chapter 3. In Chapter 4, we describe some approaches that have shown a high convergence rate in single-objective optimization problems: differential evolution and particle swarm optimization. Thus, these approaches are obvious candidates to be adapted as search engines in our multi-objective optimizer. We also discuss the most relevant attempts to extend differential evolution and particle swarm optimization to multi-objective optimization. In Chapter 5 we talk about the incorporation of knowledge into an evolutionary algorithm aiming to reduce the number of function evaluations using function approximation techniques (also known as surrogates). Also, we decided to hybridized these approaches with a local search procedure that helps us to find more solutions and make smooth moves along the Pareto front. For this purpose, we have used a mathematical approach called *rough sets*, and this can be found in Chapter 6. In Chapter 7 we talk about different techniques that use an external archive to retain nondominated solutions and we propose a technique that is a new archiving mechanism (which extends and improves ϵ-dominance) called Pareto-adaptive ϵ-dominance. The results of the different approaches are included in Chapter 8, in which we present the comparison of the techniques dealing with few fitness evaluations and the results obtained by our Pareto-adaptive ϵ-dominance technique to retain nondominated solutions. Finally, in Chapter 9 we provide some final remarks and mention some of the paths for future research.

2

Background

2.1 Optimization

In mathematics, optimization refers to the study of problems in which one seeks to find the minimum or maximum value of a function. Optimization problems appear in many areas: engineering, biology, economics, genetics, operations research, physics, or chemistry, among others [21].

For the purposes of this thesis, an optimization problem will be formally defined as follows:
Find the vector $\vec{x}$ which minimizes (without loss of generality, we assume only minimization problems) the function $f(\vec{x})$ subject to $\vec{x} \in S$, where $S \subseteq \mathbb{R}^n$ is the feasible region which satisfies the m inequality constraints:

$$g_i(\vec{x}) \leq 0; \quad i = 1, \ldots, m$$

and the p equality constraints:

$$h_j(\vec{x}) = 0; \quad j = 1, \ldots, p$$

In the previous definition, the decision variables are the values that we modify in order to solve the problem, and are denoted by the vector

$\vec{x} = (x_1, x_2, \ldots, x_n)$. The function f is usually called an objective function, or fitness function. The feasible solution that minimizes the objective function is called an optimal solution. Generally, when the feasible region of the problem does not present convexity, there exist several local optimal solutions.

In an optimization problem, the types of mathematical relationships between the objective, constraints and the decision variables determine how hard is the problem to solve, the solution methods or algorithms that can be used for the optimization process and the confidence in that the final solution that we produce is truly optimal. If the objective function and all the constraints are convex, we can determine precisely a feasible solution, find the globally optimal solution, and solve the problem up to very large size in decision variable space. If the function is non-convex, the problem is much harder to solve and it becomes uncertain if we will be able to find the feasible region and hence the global optimum.

A large number of algorithms have been proposed [125] for solving non-convex problems, but none of them is capable of guaranteeing that the best solution that they produce is the global optimum. Among them, different methods exist to deal with nonlinear problems [128]: steepest descent, Nelder-Mead, simplex method, conjugate gradient, etc.

2.1.1 Basic definitions

Global minimum: A function $f(\vec{x})$ defined on a feasible set S contains a global optimum $\vec{x^*} \in S$ iff $f(\vec{x^*}) \leq f(\vec{x}) \; \forall \; \vec{x} \in S$

Local minimum: A function $f(\vec{x})$ defined on a feasible set S contains a local optimum $\vec{x}^+ \in S$ iff $f(\vec{x}^+) \leq f(\vec{x}) \; \forall \; \vec{x} \in S$ within an $\epsilon > 0$ distance from $\vec{x}^+$. This means that for all $\vec{x}$ satisfying $|\vec{x} - \vec{x}^+| < \epsilon$, $f(\vec{x}^+) \leq f(\vec{x})$ stands.

Convex Function: A function $f(\vec{x})$ is called convex, if for any two vectors $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^n$ and $\theta \in [0, 1]$:

$$f(\theta\vec{x}_1 + (1-\theta)\vec{x}_2) \leq \theta f(\vec{x}_1) + (1-\theta) f(\vec{x}_2)$$

The next problem is an example of a *nonconvex programming* problem, a special type of nonlinear problem that typically has multiple local optima. Consider the following problem:

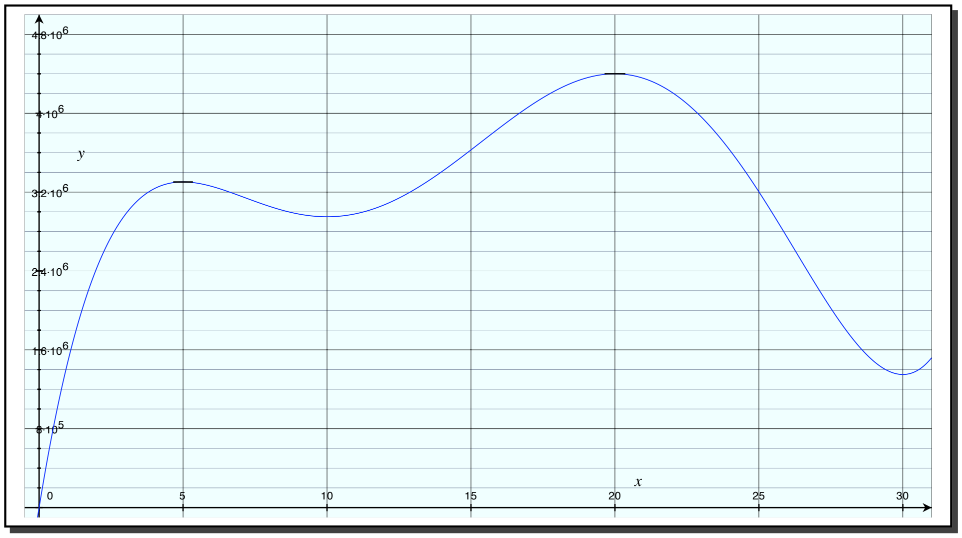


Figure 2.1: A plot of the value of the objective function over the feasible range, $0 \leq x \leq 31$, for the nonlinear programming example.

Maximize:

$$f(x) = 12x^5 - 975x^4 + 28,000x^3 - 345,000x^2 + 1,800,000x$$

subject to:

$$0 \leq x \leq 31$$

Figure 2.1 graphs the objective function $f(x)$ over the feasible values of the single variable x. This plot reveals that the problem has three local optima, one at $x = 5$, another at $x = 20$, and the third at $x = 31$, where the global optimum is at $x = 20$. The objective function $f(x)$ is sufficiently complicated that it would be difficult to determine where the global optimum lies without the benefit of viewing the plot in Figure 2.1.

2.1.2 Karush-Kuhn-Tucker conditions

The Karush-Kuhn-Tucker (KKT) conditions were derived independently by Karush [83] and by Kuhn and Tucker [90]. The KKT conditions help us to recognize an *optimal solution* for a nonlinear programming problem (with differentiable functions). These conditions give us the necessary and sufficient conditions that a solution must satisfy.

Lets assume that $f(x), g_1(x), g_1(x), \ldots, g_m(x)$ are differentiable functions satisfying certain regularity conditions. Then:

$$\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$$

can be an *optimal solution* for the nonlinear programming problem if there exist $u_1, u_2, \ldots, u_m$ and $v_1, v_2, \ldots, v_p$ such that *all* the following KKT conditions are satisfied:

$$\begin{aligned} \frac{\partial f(\vec{x}^*)}{\partial x_j} + \sum_{i=1}^{m} u_i \frac{\partial g_i(\vec{x}^*)}{\partial x_j} + \sum_{j=1}^{p} v_i \frac{\partial g_i(\vec{x}^*)}{\partial x_j} &= 0 \\ g_i(\vec{x}^*) &\leq 0 \qquad \text{for i =}1, 2, \ldots, m \\ h_j(\vec{x}^*) &= 0 \qquad \text{for j =}1, 2, \ldots, p \\ u_i &\geq 0 \qquad \text{for i =}1, 2, \ldots, m \end{aligned} \tag{2.1}$$

However, note that satisfying these conditions does not guarantee that the solution is optimal, certain additional convexity assumptions are needed to obtain this guarantee:

"Assume that $f(\vec{x})$ is a *convex* function and that $g_1(\vec{x}), g_2(\vec{x}), \ldots, g_m(\vec{x})$ are *convex* functions and $h_1(\vec{x}), h_2(\vec{x}), \ldots, h_m(\vec{x})$ be *affine* functions, where all these functions satisfy the regularity conditions. Then $\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is an *optimal solution* if and only if there exists u_i $(i = 1, \ldots, m)$ and v_j $(j = 1, \ldots, p)$ such that all the KKT conditions are satisfied".

2.2 Optimization problems

There exist a large variety of optimization problems, and techniques to handle them. Optimization problems can be subdivided in:

Linear Programming Problems (LP).- are those problems in which the objective and all the constraints are linear functions of the decision variables. A brief example of a linear function is: $f(x) = 2x_1 + 4x_2 + 8x_3$. Since all linear functions are convex, linear programming problems are intrinsically easier to solve than general nonlinear (NLP) problems, which may be non-convex. In a non-convex NLP there may be more than one feasible region and the optimal solution might be found at any point within any such region. In contrast, an LP has at most one feasible region without curves, and the optimal solution will always be found at a corner point on the surface where the constraints intersect. A LP may have multiple (or infinite) number of optimal solutions when two extremes are optimum, then all the segment that connects the extremes contain infinite number of optimal solutions.

Quadratic Programming Problems (QP).- are those in which the objective function is a quadratic function of the decision variables and the constraints are quadratic functions. An example of a QP is: $f(x) = 2x_1^2 + 4x_2^2 + 6x_1x_2$. QP problems have only one feasible region with *flat faces* on its surface, but the optimal solution may be found anywhere within the whole region or on its surface. The quadratic function could be convex (easier to solve) or non-convex which makes it very difficult to solve. However, a faster and more reliable way to solve a QP is to use an extension of the Simplex method [28] or an extension of the Interior Point or Barrier method [147].

Integer Programming Problems (IP).- are those problems where some of the decision variables can only have integer values. Integer variables make an optimization problem non-convex, and therefore more difficult to solve. Memory and solution time may rise exponentially as more integer variables are added to the problem. This kind of problems can be solved by a potentially exhaustive search. The most common used method for solving these problems is called "Branch and Bound" [96]. This method begins by finding the optimal solution of the problem without the integer constraints (using standard linear or nonlinear optimization methods). Using this solution, the Branch and Bound method chooses one variable and creates two new instances of the problem where the value of that variable is more tightly constrained. These new instances are then solved and the process is repeated, until a final solution that satisfies all the integer constraints is found. Other methods that can deal with these problems are Tabu Search [57] and Ant Colony Optimization [43].

Smooth Nonlinear Optimization Problems (NLP).- are those problems in which the objective or at least one of the constraints is a smooth nonlinear function of the decision variables. An example of a smooth nonlinear function is: $f(x) = 3x_1^2 + \log(x_2^4) + \exp(x_3)$. NLP problems and their solution methods require nonlinear functions that are continuous, and require functions that are smooth (derivatives of these functions are continuous). NLP solvers generally exploit the smoothness of the problem functions by computing gradient values at various trial solutions. They usually also exploit second derivative information to follow the curvature as well as the direction of the problem functions.

Nonsmooth Optimization Problems (NSP).- are the most difficult type of problems to solve. Those problems have multiple feasible regions and multiple locally optimal points within each region. Because some of the functions are non-smooth or discontinuous, the derivative or gradient information generally cannot be used to determine the direction in which the function is increasing (or decreasing). In other words, the situation at one possible solution gives very little information about where to look for a better solution. In this type of problems is where Evolutionary Algorithms (see Section 2.3) are very useful.

2.3 Evolutionary algorithms

An evolutionary algorithm (EA) is a search method that is inspired on natural selection and, particularly, the "survival of the fittest" principle that exists in the biological world. EAs are population based search techniques, which means that they operate on a set of solutions (called population) instead of operating on only one solution at a time. At each iteration of an EA a competitive selection mechanism that keeps the fittest solutions is applied. The solutions with the highest fitness values have the highest probability of being recombined with other solutions to mix information and form new solutions, which are expected to be better than their predecessors. This process is repeated until a termination condition is reached.

The pseudo code of an evolutionary algorithm is shown in Algorithm 1.

The main components, procedures, or operators that must be specified in order to define a particular EA are:

Representation (definition of individuals).- To link the real world problems into the EA, we need to use a particular representation of the decision variables. There are two levels of representation used in an

Algorithm 1 Generic scheme of an evolutionary algorithm

```
1: Initialize vectors of the population $P$ randomly
2: Evaluate the cost of each vector (candidate)
3: repeat
4:     Select parents;
5:     Recombine parents;
6:     Mutate the resulting solutions;
7:     Evaluate new candidates;
8:     Select the fittest individuals for the next generation;
9: until Termination condition is satisfied
```

EA: **genotypic** and **phenotypic**. The genotype is the encoding represented by the chromosome and genes[1]. The phenotype is the result of decoding the values of the chromosome into the decision variable space of the problem. The most used representations adopted with EAs are: binary [70], integer [19], real [53], tree [89] and hybrid [61].

Evaluation function (fitness function).- the fitness of an individual is related to the objective function value and it also represents the task to solve in the evolutionary context. The evaluation function assigns a quality measure with respect to other solutions in the population.

Population.- it is very important to maintain a good diversity of solutions in the population. This implies keeping different solutions (or individuals) within the population which are located in different regions of the search space, avoiding that an EA gets trapped in local optima.

Parent selection mechanism.- this mechanism allows the best individuals within the population to become parents of the next generation and guides the search towards solutions with a higher fitness. An individual is selected as a Parent if survives the selection mechanism that can be: (1) *Proportional Selection* [60], in which the individuals are selected based on their fitness, (some variations are: Roulette Wheel [32], Universal Stochastic [8] and Deterministic Sampling [32]). (2) *Tournament Selection* [165], in which direct comparisons among individuals (two or more) are performed and the fittest one is selected as a parent.

[1]A chromosome is a data structure representing an individual in a population, and a gene encodes one decision variable of the problem within the chromosome.

Variation operators, recombination and mutation.- Their function is to modify the way in which the parents are combined to form the offspring. The *crossover* (or recombination operator) uses two or more parents to generate one or two offspring. The main principle of recombination is to produce an offspring that combines the selected parents to form better individuals into the search space. There exists several recombination operators such as one or two point crossover [32], and uniform crossover [153] among others [7]. The *mutation* operator is applied to only one solution and slightly modifies the genetic information of the offspring [106]. The general idea of mutation is allowing a solution to “jump (or abruptly move) from one region to another in the search space.

Survivor selection mechanism (replacement).- This mechanism helps the EA to distinguish among individuals based on their fitness or quality, favoring those with the highest quality.

As the EAs are stochastic, there are no guarantees that the final solution has reached the global optimum by the time the stopping condition has been reached. Thus, termination condition is an important issue in EAs. If the optimization problem has a known optimal solution, the EA should stop when the objective function reaches the desired level of accuracy. If not, then the user should determine the number of generations allowed, the maximum allowed CPU time, the maximum number of fitness evaluations, or some other similar criterion.

2.4 Paradigms

There exist three main paradigms within evolutionary computation: Evolution strategies, evolutionary programming and genetic algorithms. A brief description of each of these paradigms and some other interesting approaches are provided next.

2.4.1 Evolution strategies

Evolution strategies (ES) were proposed by Ingo Rechenberg and Hans-Paul Schwefel [69], and it is normally used for continuous parameter optimization, adopting the mutation operator as the main means of creating offspring. The general scheme is described in Algorithm 2.

The mutation operator is based on a normal (Gaussian) distribution (requiring the mean and the standard deviation as parameters). The mutation parameters are usually changed during a run of the algorithm, allowing a good exploration at the beginning and a better exploitation of solutions by the end of the evolutionary process.

In ES, the parent selection is randomly perform with a uniform distribution from a population of μ individuals. The basic recombination method involves two parents that create a single offspring, called *local recombination*. Later on, a *global recombination* was proposed, in which a set of Q parents are selected to contribute at the moment of generating the offspring. However, the exact number of parents cannot be defined precisely. After creating λ offspring ($\lambda > \mu$) and calculating their fitness, the best μ of them are chosen in a deterministic way (based only on their fitness value). There exist several selection schemes, the first one is called $(1+1)-$ES, in which a single solution generates a single offspring. Offspring can be generated only from the parent population ($(\mu, \lambda)-$ES selection), or from the union of parents and offspring ($(\mu + \lambda)-$ES selection).

Algorithm 2 Generic scheme of evolution strategies algorithm

```
1: Initialize population $P$ randomly
2: Evaluate fitness of all individuals of $P$
3: repeat
4:     Apply crossover to create offspring $S$
5:     Apply mutation to all $S$
6:     Evaluate new candidates (offspring)
7:     Select the best individuals for the next generation, using:
8:     $(\mu, \lambda)$-selection
9:     or:
10:    $(\mu + \lambda)$-selection
11: until Termination condition is satisfied
```

2.4.2 Evolutionary programming

Evolutionary programming (EP) was proposed by Lawrence J. Fogel [101], who originally adopted finite state machines as predictors and evolved them.

The general scheme is described in Algorithm 3.

The parent selection in EP is deterministic and every member of the population creates exactly one offspring via mutation. The crossover in EP is not used, as the members of the population are viewed as part of a specific species rather than members of the same species; and different species are not allowed to recombine (as happens in nature). After having created the offspring, each solution is evaluated and a $(\mu + \mu)$-selection is normally adopted. Thus, each solution participates with other q solutions in a binary tournament assigning a “win” if one solution is better than its opponent. Finally, the μ solutions with the greatest number of wins are retained to be the parents of the next generation.

Algorithm 3 Generic scheme of an evolutionary programming algorithm

```
1: Initialize vectors of the population $P$ randomly
2: Evaluate fitness of all initial individuals of $P$
3: **repeat**
4:     Apply mutation to all solutions of P and generate $|P|$ offspring
5:     Evaluate new candidates (offspring)
6:     $(\mu + \mu)$-selection (stochastic process)
7: **until** Termination condition is satisfied
```

2.4.3 Genetic algorithms

The genetic algorithm (GA) was initially proposed by John Holland in his book *Adaptation in Natural and Artificial Systems* [70]. Holland had been convinced that the recombination of groups of genes by means of mating was a critical part of evolution. By the mid-1960’s Holland developed a programming technique, the genetic algorithm, that was well suited to evolution by both mating and mutation. During the next decade, he worked to extend the scope of genetic algorithms by creating a genetic code that could represent the structure of any computer program. The result was the classifier system, consisting of a set of rules, each of which performs particular actions every time its conditions are satisfied by some piece of information.

The classifier system starts with a population of random strings of 1’s and 0’s and rates each string according to the quality of the result. Depending on the problem, the measure of fitness could be business profitability, game payoff, error rate or any number of other criteria. High-quality of strings are selected to mate among them, and low-quality of strings tend to perish. As generations pass, strings associated with improved solutions will predominate. Furthermore, the mating process continually combines these strings in new ways, generating ever more sophisticated solutions. The types of problems that have been solved using this technique range from developing novel strategies in game theory to designing complex mechanical systems. The simple genetic algorithm is shown in Algorithm 4

Algorithm 4 Generic scheme of a genetic algorithm

```
1: Initialize vectors of the population $P$ randomly
2: Evaluate the cost of each vector (candidate)
3: repeat
4:     Select a pair of parents;
5:     Recombine pairs of parents;
6:     Apply mutation to offspring;
7:     Evaluate offspring;
8:     Select the fittest individuals for the next generation;
9: until Termination condition is satisfied
```

Currently, there exist many variants of GAs, with different solution representations, as well as a variety of selection, crossover and mutation operators [6]. But the characteristics of the so-called simple GA are the use of *binary representation*, *fitness proportional selection*, a low probability *uniform mutation*, and *1-point crossover* (with a high probability of crossover) [59].

2.4.4 Memetic algorithms

The concept of *memetic algorithm* was first introduced in 1989 by Pablo Moscato [112]. The term "memetic" has its roots in the word "meme" introduced by Richard Dawkins in 1976 [30] to denote the unit of imitation in cultural transmission. Memetic algorithms (also called hybrid evolutionary algorithms) are increasingly being used for solving multiobjective optimization problems. The essential idea behind memetic algorithms is the hybridization of local search refinement techniques with a population-based strategy, such as evolutionary algorithms. The main difference between genetic and memetic algorithms is the approach and view of the information's transmission techniques. In the genetic algorithm, the genetic information carried by genes is usually transmitted intact to the offspring, meanwhile in the memetic algorithm, the base unit are the so-called "memes" and they are typically adapted by the individual transmitting information. While genetic algorithms are good at exploring the solution space from a set of candidate solutions, the memetic algorithms explore from a single point, allowing to exploit solutions that are close to optimal solutions.

As stated by Knowles [87], for improving optimization results achieved by genetic algorithms one should: "Hybridize where is possible". The hybridization of local improvement operators among the evolutionary steps of an evolutionary algorithm is essential to improve solutions that are close from becoming optimal. This has been shown in several application domains to bring improvements to the standard results achieved by standalone genetic algorithms in terms of results quality and speed of convergence. The combination of global and local search is a strategy used by many successful global optimization approaches, and has in fact been recognized as a powerful algorithmic paradigm for evolutionary computing [59].

2.4.5 Other approaches

There are other EAs which are not included in the three main paradigms but that are commonly used to solve optimization problems. They are: genetic programming [89], differential evolution [151], particle swarm optimization [101] [84], ant colony optimization [43], cultural algorithms [132], artificial immune systems [31], and scatter search [95] among others.

2.5 Advantages and disadvantages of evolutionary algorithms

The motivation for using evolutionary algorithms as a search and optimization technique is partly dependent on the nature of the problem that we want to solve. In other words, if the optimization problem is mathematically "well-behaved", then one should choose the use of conventional, deterministic and specialized techniques [107]. Examples of such techniques are the classical *linear* and *nonlinear programming* techniques [100], the deterministic gradient-based techniques such as *Newton's method* and its variations and the *simplex method* [28]. However, these conventional optimization techniques had several problems when dealing with functions that present any of the following features: when the function is discontinuous or characterized by many local optima (called as multi-modal) and points at which gradients are undefined, or when the problem involves many parameters that interact in highly nonlinear ways. In those scenarios, the exact optimization techniques usually fail to get the optima. And in those scenarios is where metaheuristics[2] such as "EAs" are powerful alternatives for exploring the search space and finding good solutions that can not be attained with conventional mathematical programming techniques.

Another major advantage of EAs is their flexibility and their applicability to a vast variety of applications. This is essential when we compared EAs with alternative optimization and search methods. Classical optimization techniques usually require a set of initial conditions and hard constraints, such as the linearity or differentiability of the objective function. The EAs are direct search methods which only require information about the objective function values, but do not require any other information. EAs can also operate on any problem encoding (binary, integer, real, and combinatorial) and accept a wide variety of data structures. Among the disadvantages on EAs are that they are not able to guarantee to reach the optimum, and as they are stochastic process they do not guarantee to provide the the same result from one execution to another.

However, the iterative and population based nature of EAs can be computationally expensive and should be taken into consideration, especially when we are dealing with expensive objective functions, In Section 5.1, we discuss the main drawbacks of using a MOEA when many evaluations are required, as they tend to require a lot of function evaluations to obtain a good approximation. In order to achieve good results using an EA, several design choices need to be cautiously decided in advance. For example, the population size, the recombination and mutation probability and the maximum number of generations. Choosing the correct parameters for an EA is far from being a straightforward process, and requires practical expertise, as the success of certain EA parameters and operators cannot be generalized [123].

[2]a metaheuristic is a heuristic method for solving a general class of computational problems. The name combines 'meta' which means 'beyond' or 'higher level' and 'heuristic' which means 'to find'.

3

Multi-objective Optimization

The multi-objective optimization problem (MOP) can be formally defined as the problem of finding:
$\vec{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ which satisfies the m inequality constraints:

$$g_i(\overrightarrow{x}) \leq 0; \quad i = 1, \ldots, m$$

the p equality constraints:

$$h_j(\overrightarrow{x}) = 0; \quad j = 1, \ldots, p$$

and optimizes the vector function:

$$\vec{f}(\vec{x}) = f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x})$$

In other words, we aim to determine from among the set S of all vectors (points) which satisfy the constraints those that yield the optimum values for all the k objective functions simultaneously. The constraints define the feasible region S and any point $\overrightarrow{x}$ in the feasible region is called a feasible point.

3.0.1 Pareto dominance

Pareto dominance is formally defined as follows:

A vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ is said to dominate a vector $\overrightarrow{v} = (v_1, \ldots, v_k)$ if and only if $\overrightarrow{u}$ is partially less than $\overrightarrow{v}$, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$ (assuming minimization).

In order to say that a solution dominates another one, this one needs to be strictly better in at least one objective, and not worse in any of them. So when we are comparing two different solutions A and B, there are 3 possibilities:

- A dominates B.
- A is dominated by B.
- A and B are incomparable.

3.0.2 Pareto optimality

The formal definition of *Pareto optimality* is provided next:

A solution $\overrightarrow{x_u} \in S$ (where S is the feasible region) is said to be Pareto optimal if and only if there is no $\overrightarrow{x_v} \in S$ for which $\vec{v} = f(\vec{x}_v) = (v_1, \ldots, v_k)$ dominates $\vec{u} = f(\vec{x}_u) = (u_1, \ldots, u_k)$, where k is the number of objectives.

In words, this definition says that $\vec{x}_u$ is Pareto optimal if there exists no feasible vector $\vec{x_v}$ which would decrease some objective without causing a simultaneous increase in at least one other objective (assuming minimization).

This definition does not provide us a single solution (in decision variable space), but a set of solutions which form the so-called *Pareto Optimal Set* (P^*). The vectors that correspond to the solutions included in the Pareto optimal set are *nondominated*.

3.0.3 Pareto front

When all nondominated solutions are plotted in objective function space, the nondominated vectors are collectively known as the *Pareto Front* (PF^*). Formally:

$$PF^* := \{\overrightarrow{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))|\vec{x} \in P^*\}$$

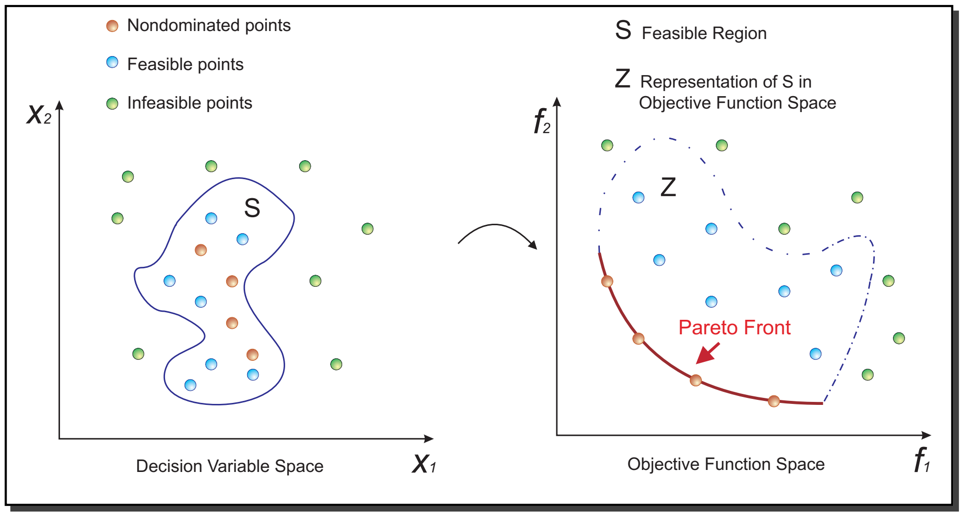


Figure 3.1: Mapping of the Pareto optimal solutions to the objective function space.

It is, in general, impossible to find an analytical expression that defines the Pareto front of a problem, so the most common way to get the Pareto front is to compute a sufficient number of points in the feasible region, and then filter out the nondominated vectors from them.

The previous definitions are graphically depicted in Figure 3.1, showing the *Pareto front*, the *Pareto optimal set* and *dominance* relations among solutions.

3.1 Decision making process

In real world multi-objective optimization, from the set of trade-offs generated by an algorithm, a single solution is usually chosen by the Decision Maker (DM). That is why the DM's preferences are so important since they will affect the multi-objective optimization process and will delimit the search space to certain regions of interest if such information is incorporated in the optimization process. However, this is not an easy task and a number of techniques have been developed for such purpose. In operations research (OS), several taxonomies of multi-objective optimization techniques (i.e., mathematical programming techniques) have been proposed. However, the most popular taxonomies are based on the stage of the search at which the DM's preferences are incorporated. One of such taxonomies (proposed in [27, 108]) is described next:

A priori **techniques.-** This category denotes the process of introducing and incorporating the DM preferences *before* the search process.

Progressive **techniques.-** This category denotes the process of introducing, incorporating and modifying the DM preferences in an interactive way at any time during the search process.

A posteriori **techniques.-** This category denotes the process of incorporating the preferences at the end of the search process.

In the following, a brief overview of these three techniques is provided.

3.1.1 *A priori* techniques

In *a priori* techniques, the DM is requested to know and expose his/her preferences before the optimization begins. The main disadvantage of *a priori* techniques is the assumption that the DM knows his/her preferences and does not allow a flexibility to change those preferences once the search process has begun. In many real world applications, the DM might be uncertain about his/her preferences of the problem and *a priori* techniques are not appropriate in this case because they are unsuitable.

There exist several proposals, some of which are reviewed next:

Lexicographic order: [50] The objective functions are sorted according to their importance, and they are optimized in sequence beginning with the most important and finishing with the less important. The performance of the algorithm is highly dependent on this ordering given by the DM.

Linear aggregating functions: It combines the results of the different objective functions into a single fitness value. This single value is often obtained by means of a linear combination of the objective functions, although that is not the only way to do it. This linear combination is done assigning a relative importance to each of the objective functions(assuming that the objective function values are all normalized). Implementing an algorithm that uses a linear combination of objectives can be very simple, and also efficient, but a linear function is unable to generate nonconvex portions of the Pareto front [29] regardless of the weights used.

Goal programming: [18, 75] The DM assigns targets (goals) that wishes to accomplish for each objective and these values are incorporated into the problem as additional constraints.

3.1.2 *Progressive* techniques

In *progressive* techniques, the DM requests the preferences progressively throughout the optimization process. With this technique, the DM can modify the preferences at any time of the optimization process and make use of any trade-off or beneficial information such as objectives' relationships. The main problem with this technique is that the interactions with the DM can be quite demanding and can require many interactions with the DM at every generation of the optimization process.

There exist several proposals along this line, some of which are reviewed next:

Sequential multi-objective problem solving method: [109] Also known as SEMOPS. It uses a surrogate objective function based on the goal levels (imposed by the DM) and aspiration levels (attainment levels of the objectives which the DM desires to achieve).

Goal sequence method: [154] Suggested as a method for articulating alternative preference scenarios for the optimization problem at hand. This technique allows the articulation of hard and soft preferences on each objective function such as priorities and constraints using logical connectives ("AND" and "OR") to drive the search towards multiple trade-off regions.

Tchebycheff method: [150] Consists of new techniques to convert multi-objective problems into a single objective problem, combining multiple objectives into a single one, using weighted distances.

3.1.3 *A posteriori* techniques

A posteriori techniques require the DM to articulate the preferences and select a single solution or a subset of solutions from a whole set of solutions provided by the multi-objective evolutionary algorithm (MOEA). This technique is the most widely used in evolutionary multi-objective optimization (EMO). A good optimizer should provide the DM with a well-diversified approximation set along the Pareto front. The next step consists in choosing a certain solution based on certain preferences and priorities and might require a further analysis and more profound. This technique allows a better understanding of the problem as well as its objective space. Nevertheless, this technique requires that the MOEA produces a large set of solutions, which can be computational expensive. Also, providing the DM with a large number of solutions to choose from can be very confusing and complicates this approach. In fact a posteriori multi-criteria decision making approaches are among the most difficult to implement and use [47].

There exist several proposals within this type of approach, some of which are reviewed next:

Weighting method: [172] This method consists in solving a scalar optimization problem in which the objective function is a weighted sum of the components of the original vector-valued functions. So, varying the components of the objective function we can generate all the solutions along the Pareto front.

ϵ-constraint method: [63] This approach suggests to keep one single objective to optimize and manage the rest of the objectives as constraints within a specific value. With this method, we can also find solutions in nonconvex parts of the Pareto front but usually at a very high computational cost.

3.2 Evolutionary algorithms for multi-objective optimization

The first attempts to adapt evolutionary algorithm to solve multi-objective optimization problems relied on straightforward transformations of a multi-objective optimization problem into a single-objective one. For example:

Weighted sum approach: Jakob et. al [78] assign different weights to each of the objectives of a problem before the optimization starts. The weights' values reflect the importance and the priority of each objective.

The problem is then converted, by aggregating the weighted objectives, into the optimization of a weighted sum.

ϵ-constraint based approach: The epsilon-constraint method [63] works by choosing one objective function as the only objective and the remaining objective functions as constraints. By a systematic variation of the constraint bounds, different elements of the Pareto front can be obtained. The method relies on the availability of a procedure to solve constrained single-objective problems. Mathematically, we aim to:

$$\begin{array}{rl} \min & f_j(\vec{x}) \\ \text{subject to:} & f_i(\vec{x}) \leq \epsilon_i \ \forall \ i \ \in \{1, \ldots, k\}, i \neq j \end{array}$$

where f: $x \rightarrow \mathbb{R}^k$ and k is the number of objectives.

Goal programming: [18] This method transforms the problem into a minimization task of a weighted sum of deviations of the objectives from user-defined goal vectors.

The main problem with these approaches is that they can only produce a single solution at each execution. In order to produce a diverse set of solutions along the Pareto front, the method needs to be executed several times changing the weight values and the optimizer configuration. In the case of the weighted sum approach, it can be easily demonstrated that is unable to produce solutions on non-convex regions of the Pareto front regardless of the weights adopted [51]. These techniques are computationally expensive and require a considerable number of objective function calls.

3.2.1 Population based evolutionary algorithms

The population-based nature of EAs and their flexible selection mechanisms have proved to be extremely useful and successful for solving multi-objective optimization problems [22]. The two factors that make the use of a population in EAs very practical are: 1) the simultaneous operation on multiple solutions transforms the search for optimal solutions into a cooperative process and hence increases its speed. 2) the Pareto dominance scheme used by most MOEAs makes it possible to tackle the problems and assess candidate solutions to such problems without requiring the aggregation of noncommensurable objectives.

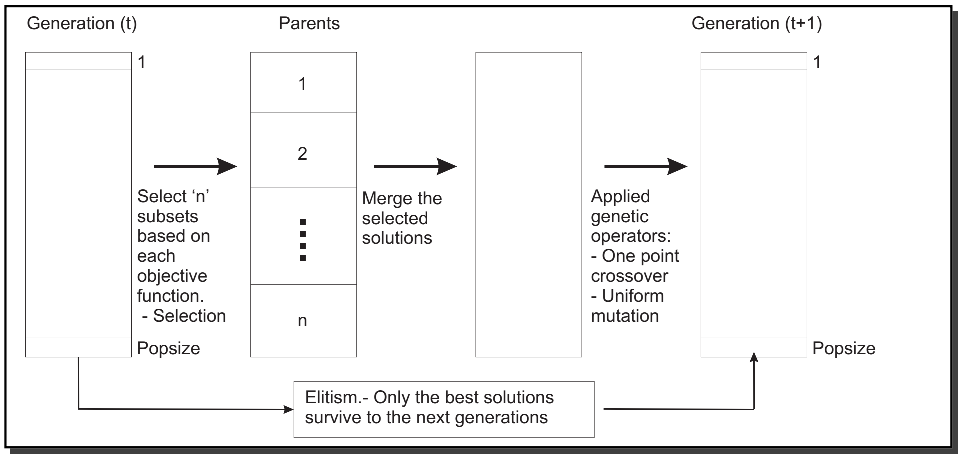

Generation (t)
Parents
Generation (t+1)
1
2
n
1
Popsize
Popsize
Select 'n' subsets based on each objective function.
- Selection
Merge the selected solutions
Applied genetic operators:
- One point crossover
- Uniform mutation
Elitism.- Only the best solutions survive to the next generations

Figure 3.2: VEGA flowchart.

3.2.1.1 Vector evaluated genetic algorithm (VEGA)

It was in 1985 when the first implementation of a multi-objective evolutionary optimizer was introduced. When David Schaffer proposed the Vector Evaluated Genetic Algorithm (VEGA) [143], he was inspired by Grefenstette's GENESIS [62] program. VEGA operates by dividing the population of solutions into N equally sized subpopulations at every generation (see Figure 3.2). Each subpopulation is designed to separately optimize only one of the N objectives. In other words, the selection of the fittest individuals in each subpopulation is based in a single objective of the problem. After performing selection, individuals are shuffled and then they are recombined and mutated. The main problem in VEGA is the selection procedure, which can hardly produce compromise solutions among all the objectives. Most of the solutions found in VEGA are in the extreme parts of the Pareto front.

3.2.2 Pareto based evolutionary algorithms

David Goldberg [59] proposed the use of Pareto dominance to assign fitness values to candidate solutions and select those solutions for recombination and survival. Most of the modern MOEA approaches are influenced by Goldberg's work, and can be categorized into two generations:

First generation MOEAs: This category includes those algorithms that lack the concept of elitism (that ensures the preservation of the good solutions found along the search process). The emphasis of this first generation is the simplicity of the approaches which normally incorporate fitness sharing or analogous processes in order to produce well-distributed sets of solutions. The most representative and widely cited are: the multi-objective genetic algorithm (MOGA) [54], the niched-Pareto genetic algorithm (NPGA) [72], and the nondominated sorting genetic algorithm (NSGA) [149].

Second generation MOEAs: These MOEAs incorporate *elitism* and more sophisticated methods to improve the diversity in the solutions (the so-called diversity estimators). Elitism is usually implemented through an external archive, also called secondary population, which stores the nondominated solutions found during the search process. Also, using a selection operator $(\mu + \lambda)$, by which at each generation, parents and children are placed together in order to select the best of them to build the next generation. The most representative and cited algorithms from this group are the following: the strength Pareto evolutionary algorithm (SPEA) [175], the Pareto archived evolution strategy (PAES) [88], the nondominated sorting genetic algorithm II (NSGA-II) [39], the strength Pareto evolutionary algorithm 2 (SPEA2) [174], the micro-genetic algorithm (μ-GA) [24] and the ϵ-dominance multi-objective evolutionary algorithm (ϵ-MOEA) [38].

3.2.2.1 Multi-objective genetic algorithm (MOGA)

MOGA was proposed by Fonseca and Fleming [54], in which the strategy is to rank a certain individual according to the number of solutions in the current population that dominates it (see Figure 3.3). It means, that a nondominated solution is assigned a rank of *one* denoting that there is no other solution in the population dominating it, and the rest of the solutions are penalized according to the number of solutions that dominate them. The rank of an individual i in the population is thus given as: $rank(i) = 1 + q_i$ where q_i is the number of individuals that dominate individual i in the objective space. Also, fitness sharing is implemented and applied in the objective space in order to obtain a well distributed set of solutions. To calculate the fitness sharing, it is first calculated as: $\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right), & d_{ij} < \sigma_{share}; \\ 0, & \text{otherwise.} \end{cases}$
where d_{ij} is the distance between individuals i and j, and σ_{share} is the niche

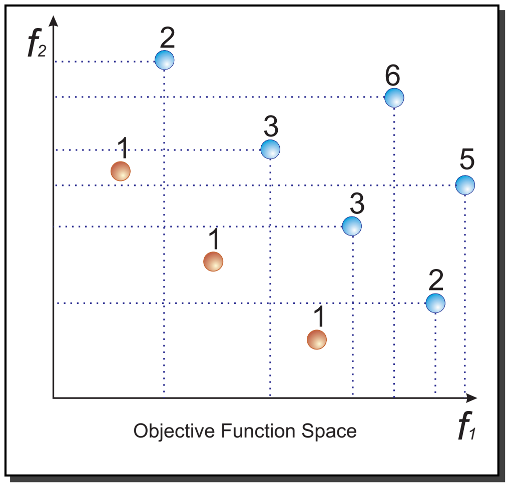


Figure 3.3: Ranking process in MOGA, the rank of the solution depends on the number of solutions that dominate it, $rank = 1$ means a nondominated solution.

radius. The sharing distance σ_{share} determines the extent of sharing allowed in terms of the radius distance (see Figure 3.4).

3.2.2.2 Niche Pareto genetic algorithm (NPGA)

Horn et. al [72] proposed a mechanism to perform a Pareto dominance tournament. Two candidate solutions are randomly chosen from the population. A certain number of solutions are randomly selected and constitute a *specific* subset (usually 10% of the population). These two selected solutions are compared with the *specific* subset, and if one of the two solutions is nondominated with the *specific* subset, then it is selected for variation. In the case of a tie, the winner is chosen based on fitness sharing, which means that the solution that resides in a less crowded region is selected.

3.2.2.3 Nondominated sorting genetic algorithm (NSGA)

Srinivas and Deb [149] proposed an algorithm based on the nondominated sorting strategy. This strategy classifies all the nondominated solutions into one category and they are all assigned the rank of *one*, denoting their membership with the first level of nondominance. The solutions with $rank = 1$ are removed from the population and the process continues until all the solutions

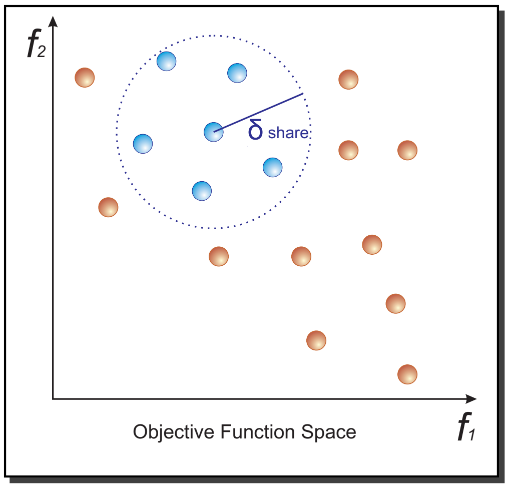


Figure 3.4: Example of fitness sharing, where all the solutions inside the σ_{share} radius are penalized in their total fitness.

in the population are assigned a rank to a certain level of nondominance (see Figure 3.5). This process only gives us a partial sorting of the solutions as all of them have the same dummy fitness value. In order to maintain diversity in the population, fitness sharing is applied to every level of dominance in order to degrade the fitness values of the solutions, based on the density of the neighborhood (called σ_{share} or niche radius defined by the user). The sharing in each front is achieved by calculating a sharing function value between two individuals in the same front as follows:

$$Sh_{d_{ij}} = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^2, & d_{ij} < \sigma_{share}; \\ 0, & \text{otherwise.} \end{cases}$$

where d_{ij} is the distance between individuals i and j, and σ_{share} is the niche radius. A parameter *niche count* is calculated by adding the above sharing function values for all individuals in the current front. Finally, the shared fitness value of each individual is calculated by dividing its dummy fitness value with its niche count. The best individuals are always preferred over other solutions, favoring the generation of new individuals near the nondominated solutions and the fitness share mechanism helps the algorithm to distribute the nondominated solutions along the Pareto front. The high sensitivity to σ_{share} resulted in a less efficient performance of NSGA.

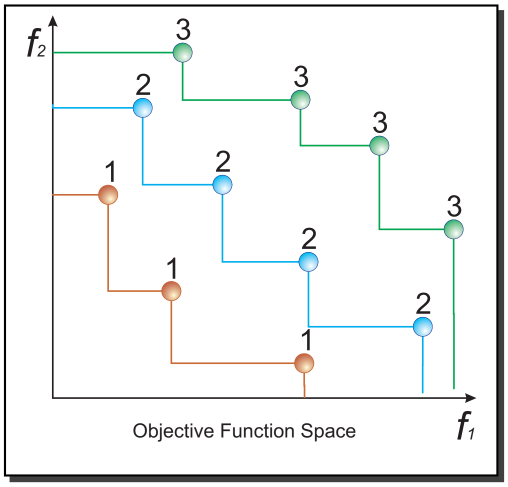


Figure 3.5: Ranking process in NSGA.

3.2.2.4 Strength Pareto evolutionary algorithm (SPEA)

Proposed by Zitzler and Thiele [175], SPEA introduced the use of two population with P for the population and P' for the external archive, which contains the nondominated solutions found so far in the process. The fitness process is based on Pareto optimality. At each generation, the nondominated individuals in P are copied to the archive and any dominated solutions in the archive are removed. In SPEA, individuals in the archive are ranked with reference to the number of members that dominate from the population (called as *strength*), while individuals in the population are evaluated with reference to the number of members that are dominated from the archive. Each solution in the population is then assigned a fitness value which is proportional to the strength of solutions that dominates. SPEA uses a clustering technique [111] to maintain diversity in the external archive, which is pruned by means of clustering once it exceeds a certain (pre-defined) size.

3.2.2.5 Pareto archived evolution strategy (PAES)

Proposed by Knowles and Corne [88], it is a simple $(1 + 1)$ evolution strategy with a single parent generating a single offspring. PAES uses an external archive (with an upper bound on its size) that contains all the nondominated solutions generated so far. Each solution generated is a candidate to be accepted in the external archive that uses an adaptive hyper-grid in the

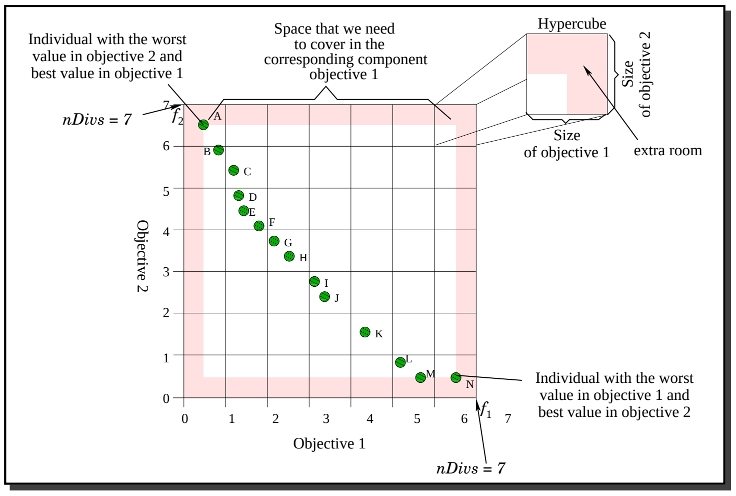

Individual with the worst value in objective 2 and best value in objective 1
Space that we need to cover in the corresponding component objective 1
Hypercube
Size of objective 2
Size of objective 1
extra room
nDivs = 7
f_2
f_1
Objective 2
Objective 1
A
B
C
D
E
F
G
H
I
J
K
L
M
N
Individual with the worst value in objective 1 and best value in objective 2
nDivs = 7

Figure 3.6: Adaptive grid in PAES.

objective space (see figure 3.6) to divide it into several hyper boxes. In the case which an offspring solution is nondominated by the reference set, another solution that resides in the most crowded region is removed from the external archive.

3.2.2.6 Nondominated sorting genetic algorithm II (NSGA-II)

Proposed by Deb et. al [39] as a second and improved version of the NSGA. NSGA-II ranks a solution using the nondominated sorting scheme. It incorporates an efficient density estimator called *crowding* in objective function space (see Figure 3.7). For each ranking level, a crowding distance is estimated by calculating the sum of the Euclidean distances between the two neighboring solutions from either side of the solution along each of the objectives. NSGA-II uses a simple $(\mu + \lambda)$ scheme for the selection survival, which means that μ parents compete against λ offspring to survive until the next generation. Currently, NSGA-II remains as one of the most competitive MOEAs known to date.

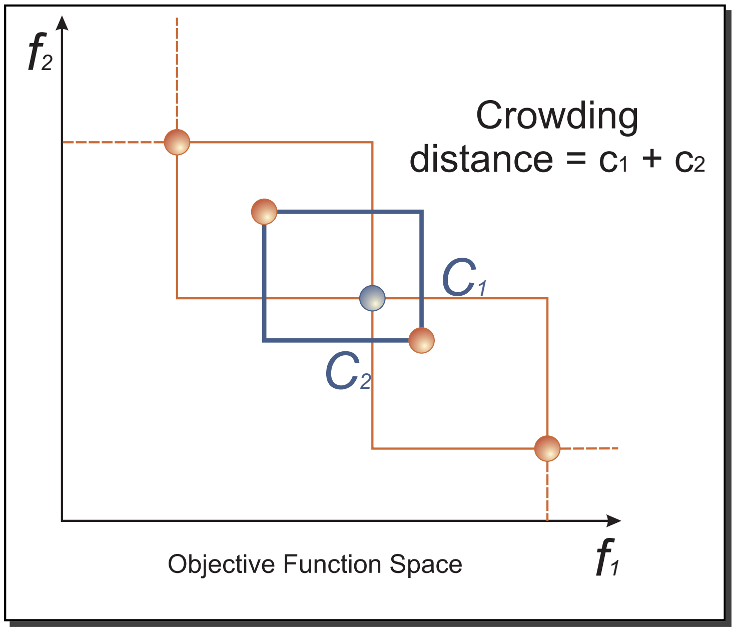


Figure 3.7: Crowding distance as a diversity operator used in NSGA-II.

3.2.2.7 Strength Pareto evolutionary algorithm 2 (SPEA2)

Proposed by Zitzler et. al [174], it contains three new features introduced to improve the original SPEA: 1) It takes into consideration the count of the dominating and the dominated solutions to help the ranking technique; 2) it uses the $k-th$ nearest neighbor strategy as a density estimator around each solution; 3) it incorporates the truncation method, which maintains a fixed size for the external population avoiding to lose the extreme parts of the Pareto front.

3.2.2.8 μ-genetic algorithm (μ-GA)

Proposed by Coello and Toscano [24], is a genetic algorithm with a small (μ) population (no more than 5 individuals) that is combined with a reinitialization process, and an adaptive grid similar to the one used in PAES. The population is divided in two: 1) the replaceable and the 2) nonrepleaceable memory. The nonrepleaceable memory does not intervene in the evolution process and it remains as the source of diversity. The replaceable memory is constantly updated with new nondominated solutions that come from the external archive. At every cycle of the μ-GA, the population is filled with four solutions from each memory (two from the replaceable and two from the nonreplaceable memory). During each cycle, the crossover and mutation operators are applied, and two solutions are chosen from the final population and are candidates to get into the external archive. The authors show that the μ-GA is very efficient (computationally speaking) and the quality of its results are also very competitive with respect to those generated by other MOEAs.

3.2.2.9 ϵ-dominance multi-objective evolutionary algorithm (ϵ-MOEA)

Proposed by Deb et. al [41], ϵ-MOEA is based on the ϵ-dominance concept first introduced by Laummans et al. [98] which is a relaxed form of Pareto dominance. The main idea is to keep two populations co-evolving: 1) the main population and 2) the secondary population. The procedure begins initializing the main population with random solutions, and all the nondominated solutions found during the search are candidate solutions to be in the secondary population (which only keeps record of the nondominated solutions). Next, two solutions are randomly selected from both populations (one from the main and other from the second one) and after applying crossover and mutation operators, two offspring solutions are generated. The mechanism to maintain diversity in the secondary population consists of dividing the objective space into a certain number of hyper-boxes of size ϵ, such that each hyper-box can only contain a single solution.

3.3 Performance measures

In order to allow a quantitative comparison of results among the different algorithm, there are two distinct goals that we pursue: (1) get the solutions as close to the Pareto optimal solutions as possible (closer to the true Pareto front) and (2) get the solutions as diverse as possible along the Pareto front (good distribution of the solutions). Apparently, these two goals are independent from each other and there exist different performance measures to deal with one or both of the goals. Thus, it does not exist a single performance measure that can indicate the superiority of one algorithm over another in these two aspects. So, there is a clear need of having at least two performance measures for adequately evaluate both goals (convergence and diversity) of a MOEA.

In Table 3.3 we can see a general scheme of the different performance measures that we adopted and their classification. In the Table 3.3 we see the characteristics of the performance measures such as:

- **unary.-** indicates if it is a unary performance measure (i.e. performance measures which assign a single value to each nondominated set of solutions).
- **binary.-** indicates if it is a binary performance measure (i.e. performance measures which assign a single value to a pair of nondominated set of solutions).
- **convergence.-** indicates that the performance measure assigns a single value corresponding to the convergence of the nondominated set.
- **diversity.-** indicates that the performance measure assigns a single value corresponding to the diversity of the nondominated set.
- **req.** PF_{true}**.-** indicates if the performance measure requires the true Pareto front to assign a single value to the nondominated set.
- **best value.-** indicates the best value that can be obtained from the performance measure.
- **Pareto compliant.-** indicates if the performance measure is Pareto dominance compliant. An indicator $I : \Omega \rightarrow \mathbb{R}$ is Pareto compliant if for all $A, B \in \Omega : A \preceq B \Rightarrow I(A) \leq I(B)$.

A MOEA will be termed as a good multi-objective solver if both goals are properly satisfied. This is, we expect to find solutions that are very close to the true Pareto front and, at the same time, are well distributed along the Pareto front.

A detailed explanation of each performance metric is provided next:

Number of points: It shows us how far the number of solutions found is from the maximum capacity of the grid. In all our experiments, the grid was defined with a capacity of 100 points. So, the closer to 100 that an algorithm gets, the better the value of this performance measure.

Performance Measures	Unary	Binary	Convergence	Diversity	req. true Pareto front	best value	Pareto compliant
Error Ratio (ER) [162]	X		X		X	0.0	X
Two Set Coverage (SC) [173]		X	X			1.0	X
($I^1_{\varepsilon+}$) [176]	X		X			0.0	X
(IGD) [163]	X		X		X	0.0	
Spread (S) [35]	X		X	X	X	0.0	
$\sigma(Crowding)$ or SDC	X			X		0.0	
SSC [175]	X		X			1.0	X
Spacing [173]	X			X		0.0	
χ^2-Like Dev Meas [149]	X			X	X	0.0	

Table 3.1: Main characteristics of the performance measures

Error Ratio (ER) Van Veldhuizen [162] simply counts the number of solutions 'N' which are not members of the Pareto optimal set, or mathematically:

$$ER = \frac{\sum_{i=1}^{|N|} e_i}{|N|}$$

where $e_i = 1$ if the solution is dominated by any member of the Pareto optimal set, and $e_i = 0$ otherwise. This metric takes a value of 0 when all the solutions are in the true Pareto front and $ER = 1$ means that no solution is in the true Pareto front.

Two Set coverage (SC) Zitzler [173] proposed this metric that can be termed relative coverage comparison of two sets. SC is defined as the mapping of the order pair (A, B) to the interval [0,1] as follows:

$$SC(A,B) = \frac{|\{b \in B; \exists a \in A : b \preceq a\}|}{|B|}$$

If all points in A dominate or are equal to all points in B, then by definition $SC = 1$. $SC = 0$ implies the opposite. Observe that this is not a distance measure that allows to determine how close these sets are from each other. $SC(A,B)$ represents the total percentage of solutions in B that are dominated for A, note that $SC(A,B) \neq 1 - SC(B,A)$

Unary additive epsilon indicator ($I^1_{\varepsilon+}$): The epsilon indicator family has been introduced by Zitzler et al. [176] and comprises a multiplicative and additive version. The unary additive epsilon indicator of an approximation set A ($I^1_{\varepsilon+}(A)$) gives the minimum factor ϵ by which each point in the true Pareto front PF_{true} can be added such that the resulting transformed approximation set is dominated by A:

$$I^1_{\varepsilon+}(A) = inf_{\epsilon \in \mathbb{R}}\{\forall z^2 \in PF_{true} \backslash \exists z^1 \in A : z^2_i \leq z^1_i + \epsilon \ \forall i\}.$$

$I^1_{\varepsilon+}(A)$ is to be minimized and implies that A is on the true Pareto front PF_{true}.

Inverted generational distance (IGD): Van Veldhuizen and Lamont [163, 164] proposed the generational distance (GD) metric as a way of estimating how far are the elements in the Pareto front produced by an algorithm from those in the true Pareto front of the problem. Mathematically:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

where n is the number of nondominated solutions found by the algorithm being analyzed and d_i is the Euclidean distance between each of these and the nearest member of the true Pareto front. A value of zero for this metric indicates that our algorithm has converged to the true Pareto front. Therefore, any other value indicates how “far” we are from the true Pareto front. We adopted a variation of this metric, called inverted generational distance (IGD) in which we use as a reference the true Pareto front, and we compare each of its elements with respect to the approximation produced by an algorithm. This provides a better estimation and avoids some of the pathological cases of poor convergence that could arise (see [20] for a more extended discussion on this metric).

Spread (Δ): Deb [35] proposed the metric Δ with the idea of measuring both progress towards the Pareto-optimal front and the extent of spread. To this end, if A is a Pareto front, Δ is defined as follows:

$$\Delta = \frac{\sum_{i=1}^{k} d_i^e + \sum_{i=1}^{|PF_{true}|} |d_i - \overline{d}|}{\sum_{i=1}^{k} d_i^e + |PF_{true}|\overline{d}}.$$

where d_i^e denotes the distance between the i-th coordinate for both extreme points in A and the true Pareto front PF_{true}. F, and d_i measures the distance of each point in A to its closer point in PF_{true}.

From the above definition, it is easy to conclude that $0 \leq \Delta \leq 1$ and the lower the Δ value, the better the distribution of solutions. A perfect distribution, that is $\Delta = 0$, means that the extreme points of the Pareto-optimal front have been found and d_i is constant for all i.

Standard deviation of the crowding distances (SDC): This performance measure was proposed in this thesis, in which we tried to get more information with the crowding distance, so we measure the standard deviation from a Pareto set as:

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

Now, $0 \leq SDC \leq \infty$ and the lower the value of SDC, the better the distribution of vectors in A, $\overline{d_i}$ is the mean value of all d_i. A perfect distribution, that is $SDC = 0$, means that d_i is constant for all i.

Size of the space covered (SSC): This metric was proposed by Zitzler and Thiele [175], and it measures the hypervolume of the portion of the objective space that is dominated by the set, which is to be maximized. In other words, SSC measures the volume of the dominated points. Hence, the larger the SSC value, the better.

Spacing (Spacing): This metric was proposed by Schott [146] and calculates a relative distance between consecutive solutions in the Pareto front, it can be formally defined as:

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

where $d_i = min_{j \in A} \sum_{m=1}^{k} |f_m^i - f_m^j|$ and $\bar{d}$ is the mean value of the distance d_i. The distance measure is the minimum value of the sum of the absolute difference in objective function values between the $i-th$ solution and any other solution in the Pareto optimal set. A value of 0 means that all the solutions are equally distributed along the Pareto front.

Chi-square-like deviation measure (χ^2-Like deviation): This metric was proposed by Srinivas and Deb [149] to measure the diversity of the set of solutions obtained. Solutions are compared with respect to a uniformly distributed set of the true Pareto front. Let P be the set of vectors uniformly distributed along the Pareto-optimal front and F the set of solutions to be compared. Then, for each $i \in \{1, ..., |P|\}$, let us denote by n_i the number of solutions in F whose distance from i is less than δ (δ is set beforehand and we use Euclidean distance). Then, the deviation is measured like a Chi-square distribution such as

$$\chi = \sqrt{\sum_{i=1}^{|P|+1} (\frac{n_i - \overline{n_i}}{\sigma_i})^2}.$$

The ideal distribution is achieved when all of the neighborhoods of points in P have the same number of vectors, that is, if for each point in P there are $\overline{n_i} = |F|/|P|$ points in F whose distance from this vector is less than δ, then $\chi = 0$. The variance σ_i^2 is proposed to be $\sigma_i^2 = \overline{n_i}(1 - \frac{\overline{n_i}}{|F|})$, for all $i \in \{1, 2, ..., |P|\}$. Index $i = |P| + 1$ is used for those points that are far from all points in P. For this index, $\overline{n}_{|P|+1} = 0$ and $\sigma_{|P|+1}^2 = |F|(1 - \frac{1}{|P|})$ are also proposed in [149]. Then, it is easy to see that $0 \leq \chi < \infty$ and the lower the χ value the better the distribution of F with respect to P. The parameter δ depends on P and it is crucial for the final χ value. Neighborhoods must be disjoint, so we take δ as a half of the minimum distance between two points in P.

3.4 Test functions

In order to challenge the search capabilities of the MOEAs, a set of multi-objective test functions were used during this thesis. These test functions encapsulate special characteristics, such as multi-modality, non-convexity and discontinuity, which are known to generally cause difficulties to most MOEAs to solve them. Some problems contains a disconnected and asymmetric Pareto fronts in two and three objective functions that causes several problems to MOEAs to reach all the regions in the true Pareto front. In Appendix A we provide the specific definitions of the different test functions that we used to validate the different approaches proposed in the thesis, and in Appendix B we show the Pareto fronts of the test functions and the optimal Pareto set (when it is possible to plot the decision variables). Also, it is important to mention that these test functions not necessarily reflect the main characteristics of the optimization problems found in the real world. It is true that some of these functions contain important features that make them particularly difficult to solve. Thus, the underlying assumption is that if a MOEA can solve such test functions, it should also be able to tackle real-world problems; although this is not necessarily true.

3.4.1 Deb's test functions

A methodology for constructing MOPs exhibiting desired characteristics has been proposed by Deb [33]. Their main characteristics are ease of construction and scalability to any number of objective functions and decision variables. Also, the associated Pareto fronts have to be known and easy to construct. Deb defines both a *local* and *global* Pareto optimal set. His global Pareto optimal set is the well known *Pareto front*, and the local Pareto optimal is "behind" the global Pareto front. He then indicates that a deceptive multi-objective optimization problem is one which at least contains both (a global and a local) Pareto fronts. This also is known as a *multifrontal* problem.

To accomplish this, Deb defines several generic bi-objective optimization problems as:

Minimize $F = (f_1(\vec{x}), f_2(\vec{x}))$, where

$$\begin{aligned} f_1(\vec{x}) &= f(x_1, \ldots, x_m), \\ f_2(\vec{x}) &= g(x_{m+1}, \ldots, x_N)\ h(f(x_1, \ldots, x_m), g(x_{m+1}, \ldots, x_N)) \end{aligned}$$

where function f_1 is a function of m where $m < N$ decision variables and f_2 a function of all N decision variables. The function g is one of $N - m$ decision variables which are not included in function f. The function h is directly a function of f and g function values. The f and g functions are also restricted to positive values in the search space, i.e., $f > 0$ and $g > 0$.

Deb lists five different functions for possible f and g and four for h. These functions may then be "mixed and matched" to create MOPs with desired characteristics. Deb states these functions have the following general effect:

f – This function controls vector representation uniformity along the Pareto front.

g – This function controls the resulting MOP's characteristics – whether it is multifrontal or has an isolated optimum.

h – This function controls the resulting Pareto front's characteristics (e.g., convex, disconnected, etc.)

These functions respectively influence the search along and towards the Pareto front, and the shape of a Pareto front. Deb implies that a MOEA has difficulty finding the global Pareto front because it gets "trapped" in a local optimum, namely a local Pareto front.

3.4.2 The ZDT test functions

Zitzler et al. [174] proposed a set of test functions which are known as the ZDT (Zitzler-Deb-Thiele) set, and will be described next. Notice that ZDT5 is not described in this part because it is a binary problem and is not used in this thesis.

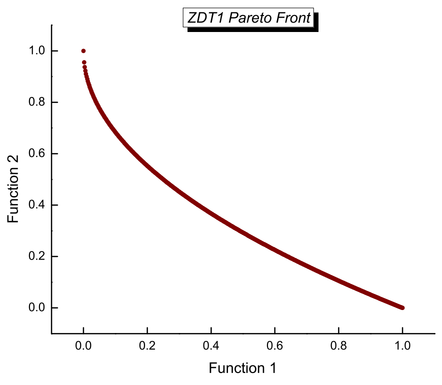


Figure 3.8: ZDT1 Pareto front.

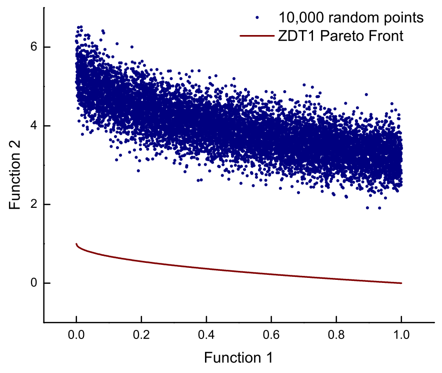


Figure 3.9: 10,000 randomly generated solutions are plotted for ZDT1.

Test Problem ZDT1: It has a convex Pareto-optimal front:

$$\begin{aligned} f_1(\vec{x}) &= x_1, \\ f_2(\vec{x}, g) &= g(\vec{x}) \cdot (1 - \sqrt{f_1(\vec{x})/g(\vec{x})}) \\ g(\vec{x}) &= 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i \end{aligned}$$

where $n = 30$, and $x_i \in [0, 1]$. The true Pareto front is formed with $g(\vec{x}) = 1$. In Figure 3.8 we show the true Pareto front. Figure 3.9 shows 10,000 random solutions in objective space together with the true Pareto front with the objective to show the difficulty of the problem to reach the true Pareto front with a random sampling.

Test Problem ZDT2: It has a concave Pareto-optimal front:

$$\begin{aligned} f_1(\vec{x}) &= x_1, \\ f_2(\vec{x}, g) &= g(\vec{x}) \cdot (1 - (f_1(\vec{x})/g(\vec{x}))^2) \\ g(\vec{x}) &= 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i \end{aligned}$$

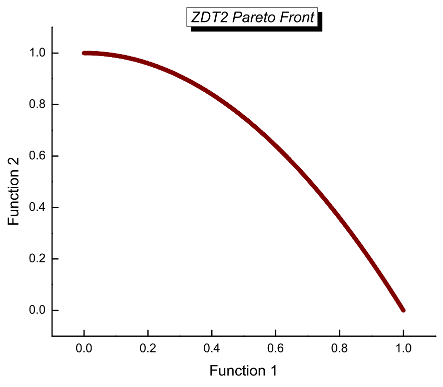


Figure 3.10: ZDT2 Pareto front.

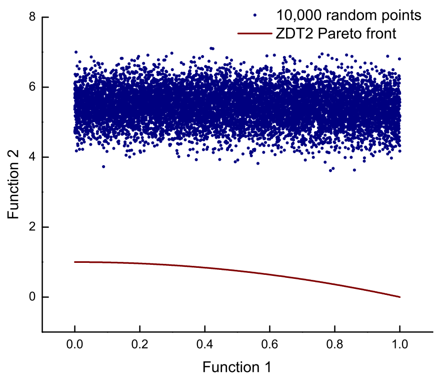


Figure 3.11: 10,000 randomly generated solutions are plotted for ZDT2.

where $n = 30$, and $x_i \in [0,1]$. The true Pareto front is formed with $g(\vec{x}) = 1$. In Figure 3.10 we show the true Pareto front. Figure 3.11 shows 10,000 random solutions plotted in objective space together with the true Pareto front.

Test Problem ZDT3: It has a Pareto-optimal front disconnected and convex:

$$\begin{aligned} f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}, g) &= g(\vec{x}) \cdot \left(1 - \sqrt{\frac{f_1(\vec{x})}{g(\vec{x})}} - \frac{f_1(\vec{x})}{g(\vec{x})} \cdot \sin(10\pi f_1(\vec{x}))\right) \\ g(\vec{x}) &= 1 + \frac{9}{n-1} \cdot \sum_{i=2}^{n} x_i \end{aligned}$$

where $n = 30$, and $x_i \in [0,1]$. The true Pareto front is formed with $g(\vec{x}) = 1$. The introduction of the *sine* function causes discontinuities in the Pareto-

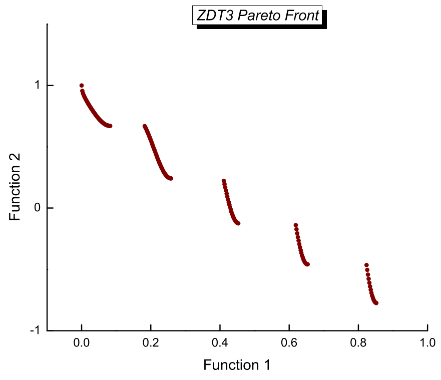


Figure 3.12: ZDT3 Pareto front.

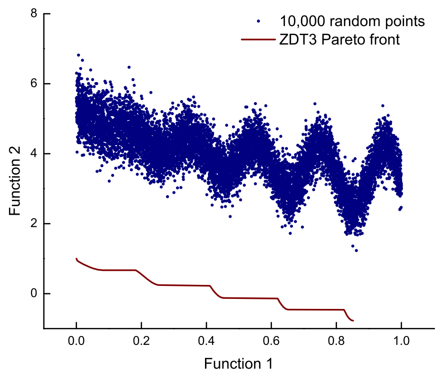


Figure 3.13: 10,000 randomly generated solutions are plotted for ZDT3.

optimal front. However, there is no discontinuity in the parameter space. In Figure 3.12 we show the true Pareto front. Figure 3.13 shows 10,000 random solutions plotted in objective space together with the true Pareto front.

Test Problem ZDT4: It contains 21^9 local Pareto fronts and, therefore, tests for the MOEA's ability to deal with multimodality:

$$\begin{aligned} f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}, g) &= g(\vec{x}) \cdot (1 - \sqrt{\frac{f_1(\vec{x})}{g(\vec{x})}}) \\ g(\vec{x}) &= 1 + 10 \cdot (n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \end{aligned}$$

where $n = 10$, $x_1 \in [0, 1]$ and $x_2, \ldots, x_n \in [-5, 5]$. The true Pareto front is formed with $g(\vec{x}) = 1$. The best local Pareto front is formed with $g(\vec{x}) = 1.25$. In Figure 3.14 we show the true Pareto front. Figure 3.15 shows 10,000 ran-

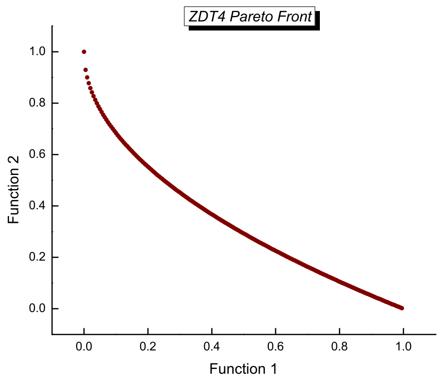


Figure 3.14: ZDT4 Pareto front.

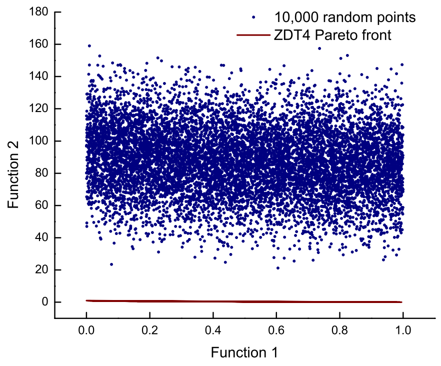


Figure 3.15: 10,000 randomly generated solutions are plotted for ZDT4 test problem.

dom solutions plotted in objective space together with the true Pareto front.

Test Problem ZDT6: It includes two difficulties caused by the nonuniformity of the search space: first, the Pareto optimal set is nonuniformly distributed along the Pareto front (the front is biased for solutions for which $f_1(\vec{x})$ is near to one); and second, the density of the solutions is lowest close to the Pareto front and highest away from the front:

$$\begin{aligned} f_1(\vec{x}) &= 1 - \exp(-4x_1) \cdot \sin^6(6\pi x_1) \\ f_2(\vec{x}, g) &= g(\vec{x}) \cdot \left(1 - \left(\frac{f_1(\vec{x})}{g(\vec{x})}\right)^2\right) \\ g(\vec{x}) &= 1 + 9 \cdot \left(\frac{1}{9} \cdot \sum_{i=2}^{n} x_i\right)^{0.25} \end{aligned}$$

where $n = 10$, $x_i \in [0, 1]$. The true Pareto front is formed with $g(\vec{x}) = 1$ and is nonconvex. In Figure 3.16 we show the true Pareto front. In Figure 3.17 10,000 random solutions are plotted in objective space together with the true Pareto front.

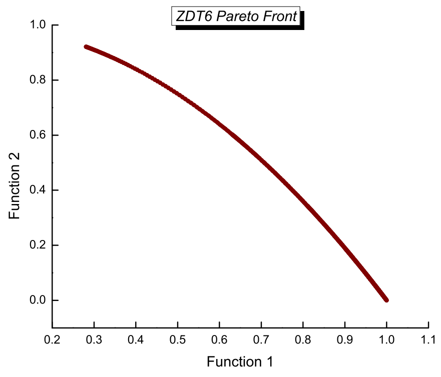


Figure 3.16: ZDT6 Pareto front.

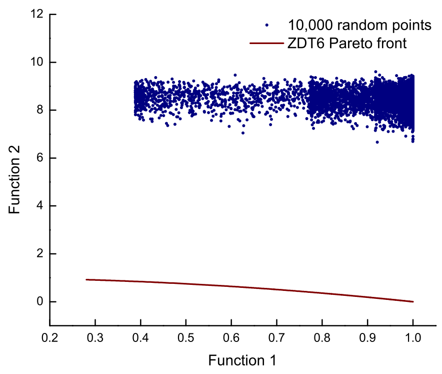


Figure 3.17: 10,000 randomly generated solutions are plotted for ZDT6.

3.4.3 The DTLZ test functions

Deb et al. [40] have proposed a set of test functions for testing and comparing MOEAs. This test suite, known as the DTLZ (Deb-Thiele-Laumanns-Zitzler) set, attempts to define generic MOEA test problems that are scalable to a user defined number of objectives. An increase in dimensionality of the objective space also causes a large portion of a randomly generated initial population to be nondominated to each other, thereby reducing the effect of the selection operator in a MOEA.

Because of these interesting features this benchmark is commonly used today in order to test a MOEA's ability to converge to the true Pareto front in problems with three or more objectives. Since the desired front can be analytically obtained for these problems, a convergence metric (such as average distance to the front) can be used to track the convergence of a MOEA.

Test Problem DTLZ1: A simple test problem using M objectives; the Pareto front is linear, separable and multimodal.

Minimize:
$f_1(\vec{x}) = \frac{1}{2}x_1x_2\ldots x_{M-1}(1+g(\vec{x}_M))$,
$f_2(\vec{x}) = \frac{1}{2}x_1x_2\ldots(1-x_{M-1})(1+g(\vec{x}_M))$,

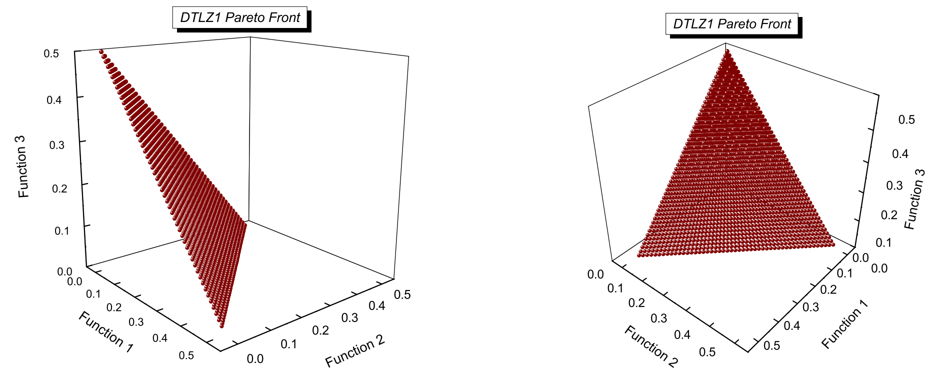


Figure 3.18: Two views of the true Pareto front of DTLZ1.

$$\vdots \qquad \vdots$$
$f_{M-1}(\vec{x}) = \frac{1}{2}x_1(1-x_2)(1+g(\vec{x}_M)),$
$f_M(\vec{x}) = \frac{1}{2}(1-x_1)(1+g(\vec{x}_M)),$

subject to $0 \le x_i \le 1 \quad \forall \quad i = 1,2,...,n$
where: $\vec{x}_M = M, M+1, \ldots, x_n$ and
$g(\vec{x}_M) = 100\left[|\vec{x}_M| + \sum_{x_i \in \vec{x}_M}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right]$

The Pareto-optimal solution corresponds to $\vec{x}^*_M = 0.5$ and the objective function values on the linear hyper-plane: $\sum_{m=1}^{M} f_i = 0.5$. The true Pareto front of this problem is shown in Figure 3.18. A value of $k = 5$ is suggested here. In the above problem, the total number of variables is $n = M + k - 1$. The difficulty in this problem is to converge to the hyper-plane. The search space contains $11^k - 1$ local Pareto fronts, each of which can attract a MOEA. The problem can be made more difficult by using other multi-modal g functions (using a larger k) and/or replacing x_i by a nonlinear mapping $x_i = N_i(y_i)$ and treating y_i as decision variables. It is interesting to note that for $M > 3$ all Pareto-optimal solutions on a three-dimensional plot involving f_M and any other two objectives will lie on or below the above hyper-plane.

Test Problem DTLZ2: This test problem is described next:

Minimize:

$f_1(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2)$,
$f_2(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2)$,
$f_3(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\sin(x_{M-2}\pi/2)$,
$\vdots \qquad \vdots$
$f_{M-1}(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\sin(x_2\pi/2)$,
$f_M(\vec{x}) = (1+g(\vec{x}_M))\sin(x_1\pi/2)$.
subject to $0 \leq x_i \leq 1 \quad \forall \quad i = 1, 2, ..., n$
where: $\vec{x}_M = M, M+1, \ldots, x_n$ and
$g(\vec{x}_M) = \sum_{x_i \in \vec{x}_M}(x_i - 0.5)^2$

The Pareto-optimal solutions corresponds to $x_i = 0.5$ for all $x_i \in \vec{x}_M \; \forall \; i = M, M+1, \ldots, n$ and all objective function values must satisfy: $\sum_{i=1}^{M}(f_i)^2 = 1$. The true Pareto front of this problem is shown in Figure 3.19. It is recommended to use $k = |\vec{x}_M| = 10$. The total number of variables is $n = M+k-1$. This function can also be used to investigate a MOEA's ability to scale up its performance with a large number of objectives. Like in DTLZ1, for $M > 3$, the Pareto-optimal solution must lie inside the first quadrant of the unit sphere in a three-objective plot with f_M as one of the axes. To make the problem more difficult, each variable x_i (for $i = 1, \ldots, M-1$) can be replaced by the mean value of p variables $x_i = \frac{1}{p}\sum_{k=(i-1)p+1}^{p} x_k$.

Test Problem DTLZ3: This is the same as DTLZ2 except for a new g function. The Pareto front is concave, scalable and multimodal.

Minimize:

$f_1(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\cos(x_{M-2}\pi/2)\cos(x_{M-1}\pi/2)$,
$f_2(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\cos(x_{M-2}\pi/2)\sin(x_{M-1}\pi/2)$,
$f_3(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\ldots\sin(x_{M-2}\pi/2)$,
$\vdots \qquad \vdots$
$f_{M-1}(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1\pi/2)\sin(x_2\pi/2)$,
$f_M(\vec{x}) = (1+g(\vec{x}_M))\sin(x_1\pi/2)$.

subject to $0 \leq x_i \leq 1 \quad \forall \quad i = 1, 2, ..., n$
where: $\vec{x}_M = M, M+1, \ldots, x_n$ and
$g(\vec{x}_M) = 100[|\vec{x}_M| + \sum_{x_i \in \vec{x}_M}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]$

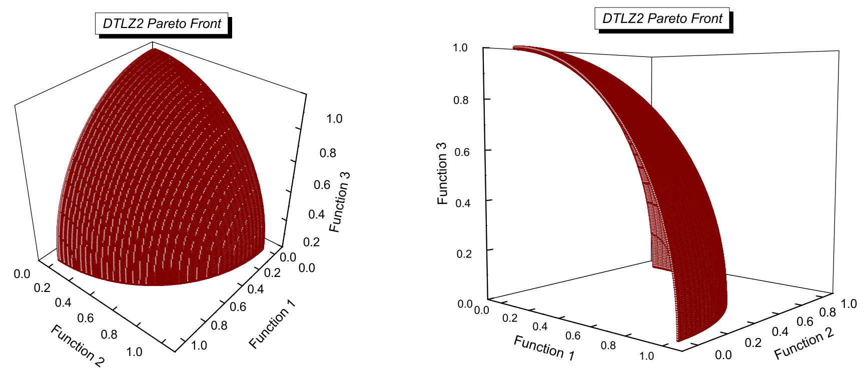


Figure 3.19: Two views of the true Pareto front of DTLZ2.

It is suggested $k = |\vec{x}_M| = 10$. There are a total of $n = M+k-1$ decision variables in this problem. The above g function introduces $3k-1$ local Pareto fronts, and one Pareto-optimal front. All local Pareto fronts are parallel to the Pareto-optimal front and a MOEA can get stuck at any of these local Pareto fronts, before converging to the Pareto-optimal front, which is located at $g^* = 0$. The Pareto-optimal front corresponds to $\vec{x}_M = (0.5, \ldots, 0.5)^T$. The next local Pareto is at $g^* = 1$. In Figure 3.20, we show the true Pareto front of this problem.

Test Problem DTLZ4: This problem uses a modified meta-variable mapping over DTLZ ($y \rightarrow y^\alpha, \alpha > 0$); the Pareto front is concave, separable and unimodal. This problem tests a MOEA's ability to maintain a good distribution of solutions as they tend to find only the extremes of the Pareto front.

Minimize:

$f_1(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1^\alpha\pi/2)\cos(x_2^\alpha\pi/2)\ldots\cos(x_{M-2}^\alpha\pi/2)\cos(x_{M-1}^\alpha\pi/2),$
$f_2(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1^\alpha\pi/2)\cos(x_2^\alpha\pi/2)\ldots\cos(x_{M-2}^\alpha\pi/2)\sin(x_{M-1}^\alpha\pi/2),$
$f_3(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1^\alpha\pi/2)\cos(x_2^\alpha\pi/2)\ldots\sin(x_{M-2}^\alpha\pi/2),$
$\vdots \qquad \vdots$
$f_{M-1}(\vec{x}) = (1+g(\vec{x}_M))\cos(x_1^\alpha\pi/2)\sin(x_2^\pi\pi/2),$
$f_M(\vec{x}) = (1+g(\vec{x}_M))\sin(x_1^\alpha\pi/2).$

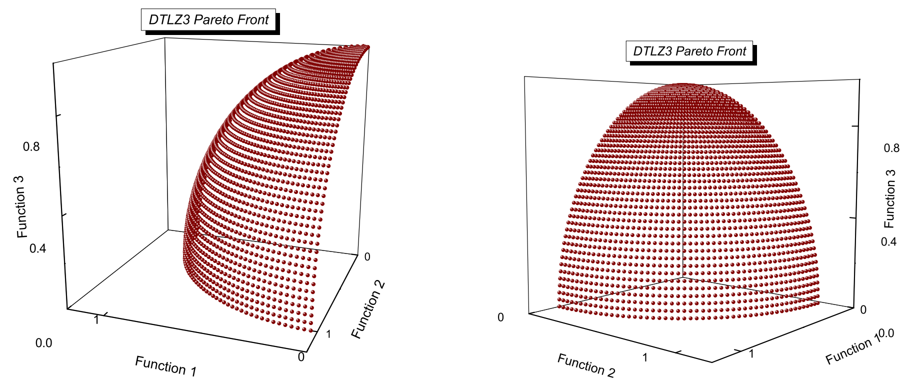


Figure 3.20: Two views of the true Pareto front of DTLZ3.

subject to $0 \leq x_i \leq 1 \quad \forall \quad i = 1, 2, ..., n$
where: $\alpha = 100$, $\vec{x}_M = M, M+1, \ldots, x_n$ and
$g(\vec{x}_M) = \sum_{x_i \in \vec{x}_M} (x_i - 0.5)^2$

Here, all variables x_1 to x_{M-1} are varied in [0, 1]. It is also suggested $k = 10$. There are $n = M + k - 1$ decision variables in the problem. The true Pareto front of this problem is shown in Figure 3.21. This modification allows a dense set of solutions to exist near the $f_M - f_1$ plane. It is interesting to note that although the search space has a variable density of solutions, the classical weighted-sum approaches or other directional methods may not have any added difficulty in solving these problems compared to DTLZ2. Since MOEAs attempt to find multiple and well-distributed Pareto-optimal solutions in one simulation run, these problems may hinder MOEAs to achieve a well-distributed set of solutions.

This methodology is not the only way to construct MOPs exhibiting some set of desired features such as curve, surface, convex, non-convex, continuous, discrete, disjoint, scalable, and others. Real-world MOPs may have similar genotype and/or phenotype characteristics but look nothing at all like the examples proposed. Thus, the fact a MOEA can successfully solve these (and other analogous) test problems may have no bearing on its performance in solving real-world MOPs. Nevertheless, and as indicated before, we expect

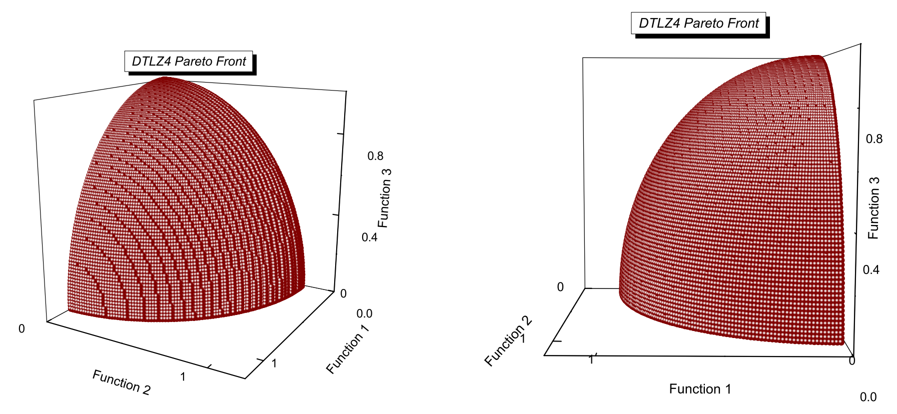


Figure 3.21: Two views of the true Pareto front of DTLZ4.

that the features included in the test functions adopted do reflect sources of difficulty of real-world problems. Additionally, such benchmarks were adopted also because of their popularity in the current specialized literature.

4 Exploration: Differential Evolution and Particle Swarm Optimization

4.1 Differential evolution (DE)

Differential Evolution (DE) is a relatively recent evolutionary algorithm designed to optimize problems over continuous domains which was proposed by Kenneth Price and Rainer Storn in the mid-1990s [151, 152]. In DE, each decision variable is represented by a real number. As in any other evolutionary algorithm, the initial population of DE is randomly generated, and then evaluated. After that, the selection process takes place. During the selection stage, three parents are chosen and they generate a single offspring which competes with a parent to determine who passes to the following generation. DE generates a single offspring (instead of two as in a genetic algorithm) by adding the weighted difference vector between two parents to a third parent. In the context of single-objective optimization, if the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with respect to which it was compared. In addition, the best parameter vector $X_{best,G}$ is evaluated for every generation G in order to keep track of the progress that is made during the minimization process. More formally, the process is described as follows:

For each vector $\overrightarrow{x_{i,G}}; i = 0, 1, 2, \ldots, N-1$, a trial vector $\overrightarrow{v}$ is generated using:

$$\overrightarrow{v} = \overrightarrow{x_{r1,G}} + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$$

with $r_1, r_2, r_3 \in [0, N-1]$, integer and mutually different, and $F > 0$.

The integers r_1, r_2 and r_3 are randomly chosen from the interval $[0, N-1]$ and are different from i. F is a real and constant factor which controls the amplification of the differential variation $(\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$.

Figure 4.1 shows a two dimensional example that illustrates the different vectors which play a role in DE.

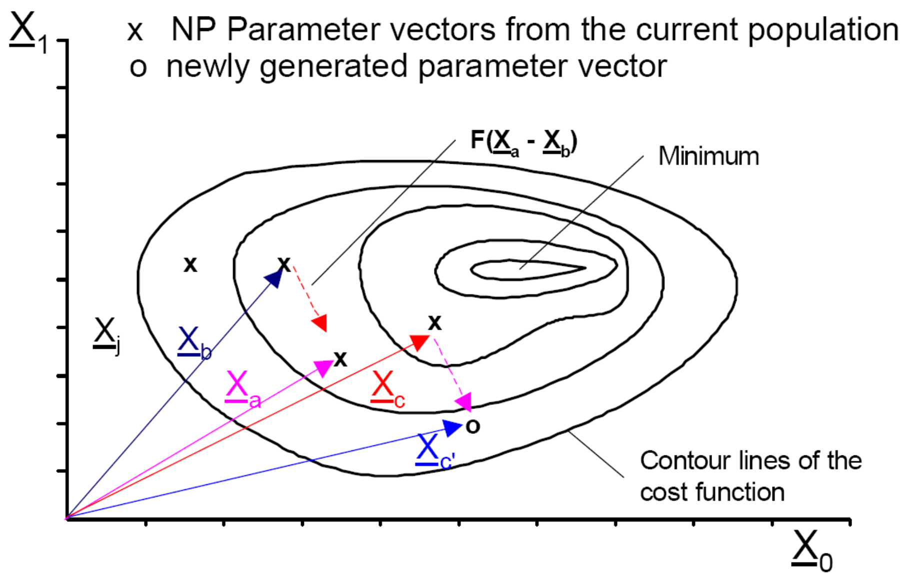

$\underline{X}_1$
x NP Parameter vectors from the current population
o newly generated parameter vector
$F(\underline{X}_a - \underline{X}_b)$
Minimum
$\underline{X}_j$
$\underline{X}_b$
$\underline{X}_a$
$\underline{X}_c$
$\underline{X}_{c'}$
Contour lines of the cost function
$\underline{X}_0$

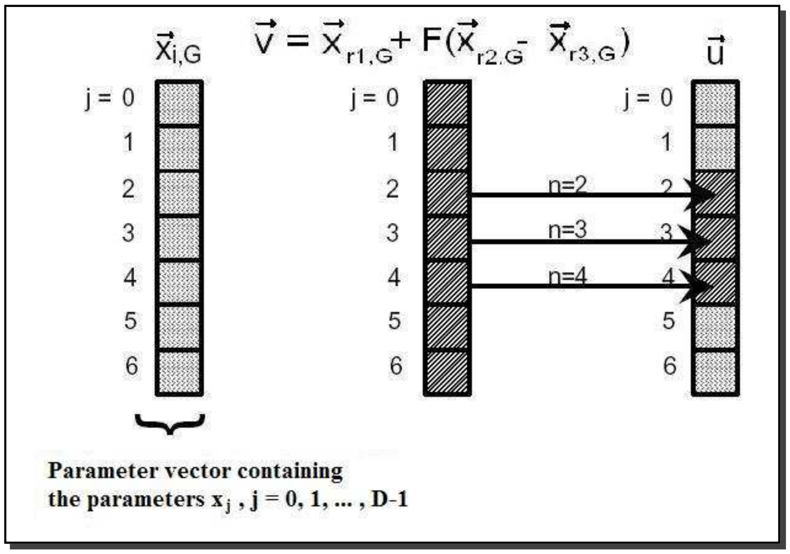
Figure 4.1: Two dimensional example of an objective function showing its contour lines and the process of generating an offspring.

In order to increase the diversity of the parameter vectors, the following vector is adopted:

$$\overrightarrow{u} = (u_1, u_2, \ldots, u_n)^T$$

with:

$$u_j = \begin{cases} v_j, & \text{if } (x \in [0,1]) \leq CR \text{ or } j = i; \\ x_{ij,G}, & \text{otherwise.} \end{cases}$$



$\vec{x}_{i,G}$
$\vec{v} = \vec{x}_{r1,G} + F(\vec{x}_{r2,G} - \vec{x}_{r3,G})$
$\vec{u}$
j = 0
1
2
3
4
5
6
n=2
n=3
n=4
Parameter vector containing the parameters x_j, j = 0, 1, ... , D-1

Figure 4.2: Differential evolution crossover with $D = 7, n = 2$ and $L = 3$

where $i = 1, 2, \ldots, n$; j is a randomly selected value from $\{1, \ldots, n\}$; and CR is a user-defined *crossover* rate in the range [0, 1]. In other words, a certain sequence of the vector elements of $\overrightarrow{u}$ are identical to the elements of $\overrightarrow{v}$, when $j = i$ at least one of the elements of $\overrightarrow{u}$ acquires the original values of $\overrightarrow{x_{i,G}}$. This idea is illustrated in Figure 4.2 with $D = 7, n = 2$ and $L = 3$. The variable n is randomly chosen in $[0, D-1]$; $L \in [0, D-1]$ with the probability $Pr(L = v) = (CR)^v.CR \in [0, 1]$, which is the crossover probability and it is also a parameter in DE. The random variables for both n and L are calculated for all new offspring $\overrightarrow{v}$.

In order to decide whether the new vector $\overrightarrow{u}$ shall become a population member at generation $G+1$, it is compared to $\overrightarrow{x_{i,G}}$. If $\overrightarrow{u}$ yields a lower objective function value than $\overrightarrow{x_{i,G}}$, then it replaces it (i.e., $\overrightarrow{x_{i,G}} = \overrightarrow{u}$). Otherwise, the old value $\overrightarrow{x_{i,G}}$ is retained.

4.1.1 Different strategies of DE

The different strategies that can be adopted in DE [122] depend on the type of problem that we want to solve. Such strategies are based on the vector that is perturbed, the number of different parents that are considered and the crossover that is used. The ten different strategies that were proposed by Price[122] are:

1. DE/best/1/exp
2. DE/random/1/exp
3. DE/target-to-best/1/exp
4. DE/best/2/exp
5. DE/random/2/exp
6. DE/best/1/bin
7. DE/random/1/bin
8. DE/target-to-best/1/bin
9. DE/best/2/bin
10. DE/random/2/bin

The last convention used is in the form:*DE/x/y/z*.

- **DE:** refers to the acronym of differential evolution.
- **x:** this term specifies how the base vector or reference parent is chosen. For example, *best* means that the base vector is the current best vector. Similarly, *random* means that base vectors are randomly chosen, while $target - to - best$ means that base vectors are chosen to lie on the line defined by the target vector and the best vector.
- **y:** this term refers to how many vector differences contribute to generate the offspring. For example, 1 means that only one vector difference is used, and 2 means that two difference vectors are used to generate a single offspring.
- **z:** this term refers to the crossover used in DE. If the strategy uses *binomial* crossover, then it is appended with the term *bin*. And the term *exp* indicates that trial vectors are generated using the *exponential* crossover. In the exponential crossover, one variable is initially chosen at random and copied from the trial vector to the offspring. The subsequent variables are determined by comparing the CR to a uniformly distributed random number between 0 and 1, i.e., $rand(0,1)$.

As long as $rand(0,1) \leq CR$, the variables continue to be taken from the trial vector, but the first time that $rand(0,1) > CR$, the remaining variables are taken from the reference parent. In binomial crossover (uniform crossover), each variable has the same probability, p_{cr}, of inheriting its value from the reference parent. With high values of CR, both crossover operators show the same behavior.

The strategy adopted to solve a specific problem is normally determined independently through a trial-and-error process. So, a specific strategy that works better to solve a problem may not work to solve another one. Also, the different parameters that are adopted to solve a problem may not work well with some other problems. The most common used strategy is *DE/rand/1/bin* which is graphically illustrated in Figure 4.3.

A detailed explanation of the example show in Figure 4.3 is provided next:

1 A vector is randomly chosen and marked as the *reference parent* from the current population, which has a fitness of 94.

2 Two parents are randomly chosen, with a fitness of 9 and 63.

3 A weighted difference vector is obtained from the two parents and is multiplied by the factor F.

4 A third parent is randomly chosen from the current population (with a fitness of 56), and it is added to the difference of the other parents to obtain a mutated vector.

5 A binomial crossover between the reference parent (step 1) and the offspring generated (step 4) is performed and its fitness is calculated, i.e., fitness $= 24$.

6 At last, a comparison is performed between the reference parent and the trial vector in order to determine who passes to the next generation. In this case: offspring (24) <parent (94), thus, the trial vector is retained and used in the next generation.

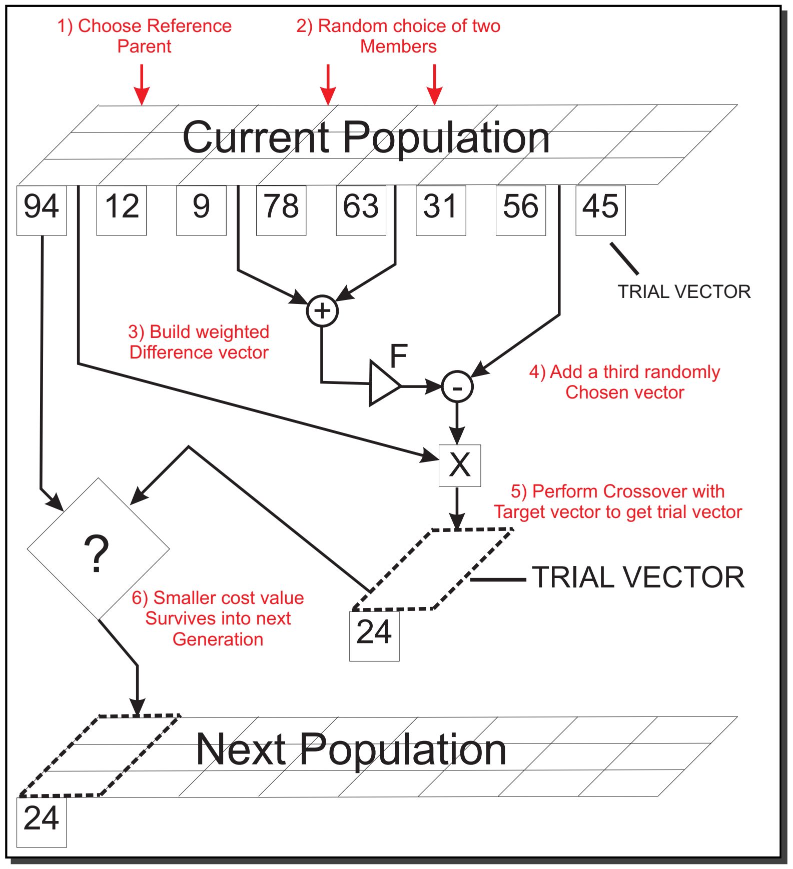

1) Choose Reference Parent
2) Random choice of two Members
Current Population
94
12
9
78
63
31
56
45
TRIAL VECTOR
3) Build weighted Difference vector
F
4) Add a third randomly Chosen vector
X
5) Perform Crossover with Target vector to get trial vector
?
TRIAL VECTOR
6) Smaller cost value Survives into next Generation
24
Next Population
24

Figure 4.3: Offspring creation process in the DE/random/1/bin variant.

4.1.2 Previous related work

Currently, there are several papers that propose ways of extending DE to handle multiple objectives. The most representative of them are briefly discussed next:

- **Pareto differential evolution (PDE)**:

  It was proposed by Abbass in 2002 [2]. An initial population is generated at random from a Gaussian distribution. All dominated solutions are removed from the population. The remaining nondominated solutions are retained for reproduction. If the number of nondominated solutions exceeds some threshold, a distance metric relation ($D(x)$, as defined in equation 4.1) is used to remove those parents which are very close to each other. Three parents are selected at random. A child is generated from the three parents and is placed into the population if it dominates the first selected parent; otherwise a new selection process takes place. Also, it is important to mention that there is another version of this approach (called the self-adaptive Pareto differential evolution) in which self-adaptive crossover and mutation operators are adopted [1].

$$D(x) = \frac{(min\|x - x^i\| + min\|x - x^j\|)}{2} \tag{4.1}$$

where $x \neq x^i \neq x^j$. That is, the nearest neighbor distance is the average Euclidean distance between the two closest points. The nondominated solution with the smallest neighbor distance is removed from the population until the total number of nondominated solutions is reduced to 50.

- **Pareto-based differential evolution approach**:

  It was proposed by Madavan in 2002 [102]. In this algorithm, differential evolution is extended to multi-objective optimization by incorporating a nondominated sorting and ranking selection procedure proposed by Deb et al. [36, 39]. Once the new candidate is obtained using DE operators, the new population is combined with the existing parents population and then the best members of the combined population (parents plus offspring) are chosen. This algorithm is not compared with respect to any other approach and is tested on 10 different unconstrained problems performing 250,000 evaluations. The authors indicate that the approach has difficulties to converge to the true Pareto front in two problems (Kursawe's test function [94] and ZDT4 [173]).

- **Multi-objective differential evolution (MODE)**:

  It was proposed by Xue in 2003 [170]. This algorithm uses a variant of the original DE, in which the best individual is adopted to create the offspring. A Pareto-based approach is introduced to implement the selection of the best individual. If a solution is dominated, a set of nondominated individuals can be identified and the "best" turns out to be any individual (randomly picked) from this set. Also, the authors adopt $(\mu + \lambda)$ selection, Pareto ranking and crowding distance in order to produce and maintain well-distributed solutions. MODE is used to solve five high dimensional unconstrained problems with 250,000 evaluations and the results are compared only to those obtained by SPEA [175].

- **Differential evolution for multi-objective optimization**:

  It was proposed by Babu in 2003 [5]. This algorithm uses the single-objective Differential Evolution strategy with an aggregating function to solve bi-objective problems. A single optimal solution is obtained after N iterations using both a *penalty function method* (to handle the constraints) and the *weighting factor method* (to provide the importance of each objective from the user's perspective) [35] to optimize a single value. The authors present results for two bi-objective problems and compare them with respect to a simple GA. The authors indicate that the DE algorithm provides the exact optimum with a lower number of evaluations than the GA.

- **Vector evaluated differential evolution for multi-objective optimization (VEDE)**:

  It was proposed by Parsopoulos in 2004 [117]. It is a parallel, multi-population Differential Evolution algorithm, which is inspired by the Vector Evaluated Genetic Algorithm (VEGA) [142] approach. A number M of subpopulations are considered in a ring topology. Each population is evaluated using one of the objective functions of the problem, and there is an exchange of information among the populations through the migration of the best individuals. VEDE is validated using four bi-objective unconstrained problems and was compared to VEGA. The authors indicate that the proposed approach outperformed VEGA in all cases.

- **Nondominated sorting differential evolution (NSDE)**:

  This approach was proposed by Iorio in 2004 [76]. It is a simple modification of the NSGA-II [39]. The only difference between this approach and the NSGA-II is in the method for generating new individuals. The NSGA-II uses a real-coded crossover and mutation operator, but in the NSDE, these operators are replaced with the operators of Differential Evolution. The authors proposed a new DE strategy called

DE/current-to-rand/1 in which the CR does not need to be specified. NSDE is used to solve rotated problems with a certain degree of rotation on each plane. The results of the NSDE outperformed those produced by the NSGA-II.

- **Generalized differential evolution (GDE)**:

  This approach was proposed by Kukkonen in 2004 [91]. GDE extends the selection operation of the basic DE algorithm for constrained multi-objective optimization. The basic idea in this selection rule is that the trial vector is required to dominate the old population member used as a reference either in constraint violation space or in objective function space. If both vectors are feasible and nondominated with respect to each other, the one residing in a less crowded region is chosen to become part of the population of the next generation. GDE its validated using five bi-objective unconstrained problems. Results are compared with respect to the NSGA-II and SPEA [175]. The authors report that the performance of GDE is similar to the NSGA-II, but they claim that their approach requires a lower CPU time. GDE is able to outperform SPEA in all the test functions adopted. Later on, in 2005, the authors proposed an extension of this approach called GDE3 [92] which is capable of handling constraints and obtains a better distribution of the solutions.

- **Differential evolution for multi-objective optimization (DEMO)**:

  DEMO was proposed by Robic in 2005 [133]. This algorithm combines the advantages of DE with the mechanisms of Pareto-based ranking and crowding distance sorting. DEMO only maintains one population and it is extended when newly created candidates take part immediately in the creation of the subsequent candidates. This enables a fast convergence towards the true Pareto front, while the use of nondominated sorting and crowding distance (derived from the NSGA-II [39]) of the extended population promotes the uniform spread of solutions. DEMO is validated using five high-dimensionality unconstrained problems outperforming in some cases to the NSGA-II, PDEA [1], PAES [88], SPEA [175] and MODE [170].

- **Nondominated sorting differential evolution with directional convergence and spread (NSDE-DCS)**:

  It was proposed by Iorio et al. in 2006 [77]. This algorithm extends the original NSDE [76] by incorporating the ranking scheme information into the DE scheme. The purpose is to produce better offspring and accelerate the convergence in the algorithm as well as to promote the spread among them. The algorithm is assessed using four multidimensional test problems with 30,000 function evaluations and its results were compared with respect to the original NSGA-II [39] and the original NSDE [77].

- **ϵ-orthogonal differential evolution for multi-objective optimization (ϵ-ODEMO)**:

  It was proposed by Cai et al. in 2007 [16]. ϵ-ODEMO is analogous to the ϵ-MOEA [38] with the exception that in ϵ-ODEMO the initial population is generated using the orthogonal design method [65] and it also adopts DE/rand/1/exp to produce new solutions. Additionally, an external archive based on ϵ-dominance is used to retain the nondominated solutions. ϵ-ODEMO is validated using the ZDT set test and two test problems with three objectives (DTLZ1 and DTLZ6). Results were compared against the ϵ-MOEA with 20,000 function evaluations. The authors show that the use of the orthogonal array method helps the algorithm to improve performance.

- **Adaptive variable strategy Pareto differential evolution (AVSPDE)**:

  It was proposed by Fu et al. in 2008 [55]. This algorithm replaces the crossover and mutation operator from the NSGA-II [39] by the DE operators. Also, the algorithm switches between two DE variants (DE/rand/1/bin and DE/rand/2/bin) dynamically during its execution, based on information extracted (in real-time) from the tournament selection. The authors use two test functions with 15,000 function evaluations, showing a fast convergence and a diverse spread when AVSPDE is compared against the NSDE-DCS [77] and NSGA-II [39].

After analyzing the different MOEAs based on differential evolution available in the specialized literature, we identified Pareto ranking and crowding distance as two of the most common and effective mechanisms adopted. We also found that most of the previous approaches were tested in unconstrained test problems with only two objectives and that little attention had been paid to the importance of the selection criteria (which is a key issue to regulate the high selection pressure of differential evolution). We also noticed that only one of these previous proposals adopted ϵ-dominance [98] to generate a set of well-spread solutions. These were the main motivations for our algorithmic design, which is presented in the next section.

4.1.3 Proposed algorithm based on DE

Our approach keeps three populations: the main population (which is used to select the parents), a secondary (external) population, in which we adopt the concept of Pareto-adaptive ϵ-dominance (see Section 7.1) to retain the nondominated solutions found and to distribute them in an uniform way and a third population that retains dominated solutions removed from the second population. The concept of ϵ-dominance [98] does not allow two solutions with a difference less than ϵ_i in the i-th objective to be nondominated with respect to each other, thereby allowing a good spread of solutions. The pseudo-code of our proposed DE-based MOEA (called ϵ-MyDE) is shown in Algorithm 5.

ϵ-MyDE uses real numbers representation, where each chromosome is a vector of real numbers (each number corresponds to a decision variable of the problem). We also incorporate a constraint-handling mechanism that allows infeasible solutions to intervene during recombination. This helps to solve in a more efficient way highly constrained multi-objective optimization problems.

At the beginning of the evolutionary process, our approach randomly initializes all the individuals of the population. Each decision variable is normalized within its allowable bounds. The expression that we adopt is the following:

$$x_i = \mathrm{LI}_i + U(0,1) \cdot (\mathrm{LS}_i - \mathrm{LI}_i)$$

where $j = 0, 1, 2, \ldots, n-1$. (n is the total number of decision variables), LI_j and LS_j are the upper and lower bounds of the variable j, respectively. $U(0,1)$ generates a random number between 0 and 1 with a uniform distribution.

Our approach has two selection mechanisms that are activated based on

Algorithm 5 Algorithm: ϵ-MyDE

```
1: Input: MyDE Parameters: $|P|$, $G_{max}$, $CR$, $P_m$ and $elitism$.
2: Output: nondominated solutions found in the secondary population.
3: Randomly initialize vectors of the population $P$
4: Evaluate the cost of each vector
5: for $i = 0$ to $G$ do
6:     repeat
7:         Select (randomly) three different vectors
8:         Perform crossover using DE scheme
9:         Perform mutation
10:        Evaluate objective values
11:        if offspring is better than main parent then
12:            replace it in population
13:        end if
14:    until population is completed
15:    Identify nondominated solutions in population
16:    Add nondominated solutions into secondary population
17:    Add dominated solutions into third population
18: end for
```

the total number of generations and a parameter called *elitism* $\in (0.2, 1)$, which regulates the selection pressure. For example, if $elitism = 0.6$ and the total number of generations is $G_{max} = 200$, this means that during the first 120 generations (60 % of G_{max}), a random selection will be adopted, and during the last 80 generations an elitist selection will be adopted.

$$\text{Type of Selection} = \begin{cases} \text{Random}, & gen < (elitism * G_{max}) \\ \text{Elitist}, & \text{otherwise} \end{cases}$$

where:
gen = generation number.
G_{max} = total number of generations.

In both selections (random and elitist), a single parent is selected as a reference. This parent is used to compare the offspring generated by three different parents. This mechanism guarantees that all the parents of the main population will be reference parents for only one time during the generating process. Both types of selection are described next:

1. **Random selection.-** three different parents are randomly selected from the primary population.

2. **Elitist selection.-** three different parents are selected from the secondary population such that they maintain a close distance f_{near} among them. In Figure 4.4, we illustrate the f_{near} parameter. If no parent exists which fulfills this condition, we randomly select another parent from the secondary population.

$$f_{close} = \frac{\sqrt{\sum_{i=0}^{k} (x_{i,max} - x_{i,min})^2}}{2^k}$$

where:
k = number of objective functions.
$x_{i,max}$ = upper bound of the i-th objective function in the secondary population.
$x_{i,min}$ = lower bound of the i-th objective function in the secondary population.

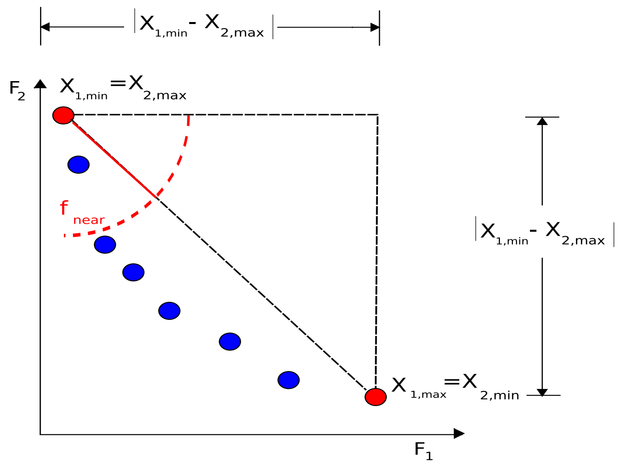


Figure 4.4: Parameter f_{close} for a bi-objective optimization problem

In Figure 4.5, we illustrate (for a bi-objective problem) the main difference between the random and the elitist selection once the main parent has

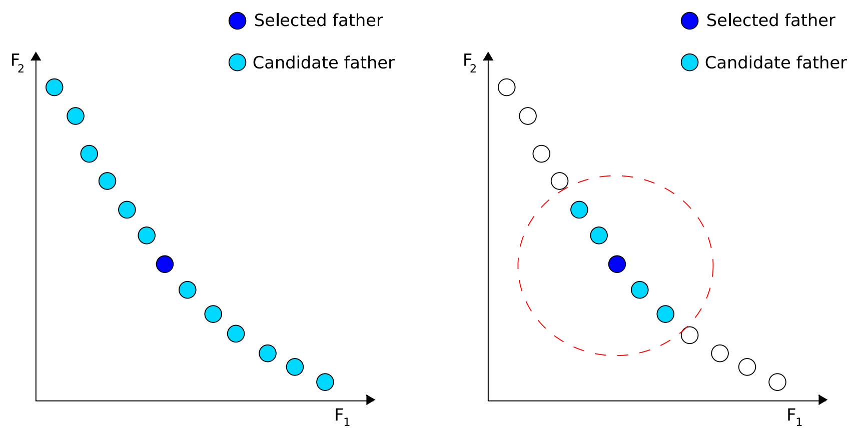


Figure 4.5: Graphical illustration of (1) random selection (right) and (2) elitist selection (left)

been selected, and it is required to select two more parents. The candidate solutions are those inside the dotted circle.

Recombination in our approach is performed using the following procedure. For each parent vector $\overrightarrow{p_i}$; $i = 0, 1, 2, \ldots, P-1 (P = \text{population})$, the offspring vector $\overrightarrow{h}$ is generated as:

$$h_j = \begin{cases} p_{r1,j} + F \cdot (p_{r2,j} - p_{r3,j}), & \text{if } x < p_{crossover} \text{or CR;} \\ p_{ref,j}, & \text{otherwise.} \end{cases}$$

where $j = 0, 1, 2, \ldots, n-1$. (n is the number of variables for each solution vector), $x \in U(0,1)$, $p_{r1}, p_{r2}, p_{r3} \in [0, |P|-1]$, are integers and mutually different, and $F > 0$. The integers r_1, r_2 and r_3 are the indexes of the selected parents randomly chosen from the interval $[0, \ |P|-1]$ and ref is the index of the reference parent. F is a constant factor (a real number) which controls the amplification of the differential variation $p_{r2,j} - p_{r3,j}$. The optimal value of F for most of the functions lies within the range from 0.4 to 1.0 [152].

Differential evolution does not use an specific mutation operator, since such operator is embedded within its recombination operator. However, in multi-objective optimization problems, we found it is necessary to provide an additional mutation operator to allow a better exploration of the search space (mainly in constrained problems). We adopted uniform mutation [42] for that sake:

$$h_j = \begin{cases} \mathrm{LI}_j + U(0,1) \cdot (\mathrm{LS}_j - \mathrm{LI}_j), & \text{if } x < p_{mutation}; \\ p_j, & \text{otherwise.} \end{cases}$$

where $j = 0, 1, 2, \ldots, n-1$. (n is the number of variables). LI_j and LS_j are the lower an upper bounds for the variable j, respectively.

Once a child has been generated, it is compared with respect to the reference parent, against which it competes to determine who passes to the following generation.

It is important to mention that in our approach, we normalize the constraints so that their value ranges between 0 and 1. This normalization is transparent for the user (the algorithm does this without requiring any input from the user). This normalization mechanism is described next:

For each constraint C_i, two different variables CS and CI store its upper and lower values. Therefore, whenever a constraint is required, the following expression is adopted:

$$NC_i = \frac{\mathrm{C}_i - \mathrm{CI}_i}{\mathrm{CS}_i - \mathrm{CI}_i}$$

The rules of comparison between a child and its parent are the following:

Version for unconstrained problems

* *If parent dominates child*, the parent is chosen.

* *If child dominates parent*, the child is chosen.

* *If both are nondominated with respect to each other*, perform a $flip(0.5)$[1] to determine who passes to the following generation.

Version for constrained problems

* *If parent is infeasible and child is infeasible*, the solution that is closest to the feasible region is selected.

* *If parent is feasible and child is infeasible*, the child is chosen if and only if the child is at least at a distance of 0.1 of the feasible region and a $flip$ (0.5) returns true. Otherwise, the father is chosen.

[1]The function $flip(0.5)$ generates a random value r in the interval (0,1), if $0.5 \leq r$ then it returns 0, returns 1 otherwise.

* *If parent is infeasible and child is feasible*, the parent is chosen if and only if the parent is at least at a distance of 0.1 of the feasible region and a $flip$ (0.5) returns true. Otherwise, the child is chosen.

* *If parent is feasible and child is feasible*, Pareto dominance is verified between them and is treated as an unconstrained problem.

Note that the scheme previously described allows some infeasible solutions to intervene during the recombination process. We found that this sort of scheme is particularly useful when dealing with highly constrained problems. It has been previously shown that maintaining infeasible solutions that lie in the frontier between the feasible and infeasible regions, helps to obtain better solutions in highly constrained problems (particularly when dealing with equality constraints) [105, 145].

As indicated before, our proposed approach uses an external archive (also called secondary population). In order to include a solution into this archive, it is compared with respect to each member already contained in the archive using the $pa\epsilon$-dominance grid [68]. Any member that is removed from the secondary population is included in the third population. The $pa\epsilon$-dominance grid is created once we obtain 100 nondominated solutions. The results from this proposed algorithm are presented in chapter 8, once that we have introduced the local search procedure based on Rough Sets and the mechanism based on ϵ-dominance to retain the nondominated solutions found during the search process.

4.2 Particle swarm optimization (PSO)

Particle Swarm Optimization (PSO) is a heuristic search technique (which is considered as an evolutionary algorithm by its authors James Kennedy and Russell Eberhart [44]) that simulates the movements of a flock of birds which aim to find food. PSO has been found to be a very successful optimization approach both in single-objective and in multi-objective problems [84, 131].

In PSO, each solution is represented by a particle. Particles group in "swarms" (there can be either one swarm or several in one population) and the evolution of the swarm to the optimal solutions is achieved by a velocity equation. This equation is composed of three elements: a velocity inertia, a cognitive component "*pbest*" and a social component "*gbest*". Depending on the topology adopted (i.e., one swarm or multiple swarms), each particle can be affected by either the best local and/or the best global particle in its swarm. PSO presents a good convergence rate to the optimum (or its vecinity) in multi-objective optimization [156], but normally has difficulties to achieve a good distribution of solutions with a low number of evaluations [25].

In the PSO algorithm, the particles (including the *pbest*) are randomly initialized at the beginning of the search process. Next, the fittest particle from the swarm is identified and assigned to the *gbest* solution (i.e., the global best, or best particle found so far). After that, the swarm flies through the search space (in n dimensions, in the general case). The flight function adopted by PSO is determined by equation (4.2), which updates the position and fitness of the particle using equation (4.3). The new fitness is compared with respect to the particle's *pbest* position. If it is better, then it replaces the *pbest* (i.e., the personal best, or the best value that has been found for this particle so far). This procedure is repeated for every particle in the swarm until the termination criterion is reached.

$$v'_{i,j} = w \cdot v_{i,j} + c_1 \cdot U(0,1)(pbest_{i,j} - x_{i,j}) + c_2 \cdot U(0,1)(gbest_j - x_{i,j}) \quad (4.2)$$

$$x'_{i,j} = x_{i,j} + v'_{i,j} \quad (4.3)$$

where c_1 and c_2 are constants that indicate the attraction from the *pbest* or *gbest* position, respectively; w refers to the inertia of the previous movement; $\vec{x}_i = (x_{i1}, x_{i2}, ..., x_{in})$ represents the $i-th$ particle; $j = 1, 2, \ldots, n$ and n represents the number of decision variables; U(0,1) denotes a uniformly random number generated within the range (0,1); $\vec{v}_i = (v_{i1}, v_{i2}, ..., v_{in})$ represents the rate change (velocity) of particle i. Equation (4.2) describes the velocity that is constantly updated by each particle and equation (4.3) updates the position of the particle in each decision variable.

4.2.1 Previous related work

The main difficulties to extend PSO to handle multiple objectives are the leader selection, the comparison operator, and the continuous problems to converge in some test functions due to a loss of diversity in the swarm. There are plenty of proposals to extend PSO for dealing with multiple objectives (see for example [4, 9, 10, 23, 103]) and the survey by Reyes-Sierra and Coello Coello [131]. A brief description of the most representative proposals is provided next and a summary of their features is presented in Table 4.1:

Abido [3]: The authors propose a PSO that retains two nondominated populations, one set that stores the nondominated solutions obtained by a single particle at the time and helps to decide the p_{best} particle in the flight equation and the second set stores the nondominated solutions obtained by all the particles. Both sets use the average linkage based hierarchical clustering algorithm [111] used by SPEA [175] to reduce the set size if it exceed a certain prespecified value. The authors test the algorithm in four bi-objective problems and compare it against the SPEA with 50,000 function evaluations, showing that the proposed MOPSO is able to capture the shape of the different characteristics of the Pareto fronts.

Alvarez-Benitez et al. [4]: The authors propose methods based exclusively on Pareto dominance for selecting leaders from an unconstrained nondominated (external) archive. The authors propose and evaluate four mechanisms for confining particles to the feasible region, that is, constraint-handling methods. The authors show that a probabilistic selection favoring archival particles that dominate few particles provides good convergence towards the Pareto front while properly covering it at the same time. Also, they conclude that allowing particles to explore regions close to the constraint boundaries is important to ensure convergence to the Pareto front. This approach uses a turbulence factor that is added to the position of the particles with certain probability.

Bartz et al. [9]: This approach starts from the idea of introducing elitism (through the use of an external archive) into PSO. Different methods for selecting and deleting particles (leaders) from the archive are analyzed to generate a satisfactory approximation of the Pareto front. Selecting methods are either inversely related to the fitness value or based on the previous success of each particle. The authors provide some statistical analysis in order to assess the impact of each of the parameters used by their approach.

Baumgartner et al. [10]: This approach, based on the *fully connected* topology, uses linear aggregating functions. In this case, the swarm is equally partitioned into n subswarms, each of which uses a different set of weights and evolves into the direction of its own swarm leader. The approach adopts a gradient technique to identify the Pareto optimal solutions.

Coello Coello et al. [23, 25]: This proposal is based on the idea of having an external archive in which every particle deposits its flight experiences after each flight cycle. The search space explored is divided in hypercubes. Each hypercube receives a fitness value based on the number of particles it contains. Once a hypercube has been selected, the leader is randomly chosen. This approach also uses a mutation operator that acts both on the particles of the swarm, and on the range of each design variable of the problem to be solved.

Fieldsend and Singh [49]: This approach uses an unconstrained elite external archive (in which a special data structure called "dominated tree" is adopted) to store the nondominated individuals. The archive interacts with the primary population in order to define leaders. The selection of the gbest for a particle in the swarm is based on the structure defined by the dominated tree. First, a composite point of the tree is located based on dominance relations, and then the closest member (in objective function space) of the composite point is chosen as the leader. On the other hand, a set of personal best particles found (nondominated) is also maintained for each swarm member, and the selection is performed uniformly. This approach also uses a "turbulence" operator which is basically a mutation operator that acts on the velocity value used by the PSO algorithm.

Mahfouf et al. [103]: The authors propose an adaptive weighted PSO (AW-PSO) algorithm, in which the velocity is modified by including an acceleration term that increases as the number of iterations increases. This aims to enhance the global search ability at the end of the run and to help the algorithm to jump out of local optima. The authors use dynamic weights to generate Pareto optimal solutions. When the population is losing diversity, a mutation operator is applied to the positions of certain particles and the best of them are retained. Finally, the authors include a nondominated sorting algorithm to select the particles from one iteration to the next.

Moore and Chapman [110]: This was the first attempt to produce a multi-objective particle swarm optimizer. In this approach, the personal best (*pbest*) of a particle is a list of all the nondominated solutions it has found in its trajectory. When selecting a *pbest*, a particle from the list is randomly chosen. Since a ring topology is used, when selecting the best particle of the neighborhood, the solutions contained in the *pbest* lists are compared, and a nondominated solution with respect to the neighborhood is chosen. The authors do not indicate how they choose the p_{best} particle when more than one nondominated solution is found in the neighborhood.

Parsopoulos et al. [118]: This is a parallel version of the vector evaluated particle swarm (VEPSO) method for multi-objective problems. VEPSO is a multi-swarm variant of PSO, which is inspired on the Vector Evaluated Genetic Algorithm (VEGA) [142]. In VEPSO, each swarm is evaluated using only one of the objective functions of the problem under consideration, and the information it possesses for this objective function is communicated to the other swarms through the exchange of their best experience (*gbest* particle). The authors argue that this process can lead to Pareto optimal solutions.

Reyes-Sierra and Coello Coello [129]: This approach is based on Pareto dominance and the use of a nearest neighbor density estimator for the selection of leaders (by means of a binary tournament). This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions. On the other hand, the concept of ϵ-dominance is used to select the particles that remain in the archive of final solutions. Additionally, the authors propose a scheme in which they subdivide the population (or swarm) into three different subsets. A different mutation operator is applied to each subset. Finally, this approach incorporates fitness inheritance [148] in order to reduce the total number of fitness function evaluations performed.

Tripathi et al. [157]: This algorithm is called adaptive multi-objective particle swarm optimization (AMOPSO) and incorporates inertia and the acceleration coefficient as control variables that the PSO needs to evolve. AMOPSO uses an external archive to retain the elitist solutions and the crowding-distance measure to help with diversity, selects the g_{best} with a roulette wheel selection and includes the PSO parameters (w, c_1 and c_2) into the evolution. Also, AMOPSO includes a mutation operator to allow a better exploration and is compared with several MOPSOs taken from the literature performing 25,000 fitness functions in seven bi-objective functions and two problems with three objective functions.

Author(s) / (reference)	Selection	Elitism	Mutation
Abido [3]	Pareto Ranking	yes	no
Alvarez-Benitez et al. [4]	Pareto Ranking	yes	no
Bartz et al. [9]	Population-based	yes	no
Baumgartner et al. [10]	Linear Aggregation	no	no
Coello et al. [23, 25]	Pareto Ranking	yes	yes
Fieldsend and Singh [49]	Dominated Tree based	yes	yes
Mahfouf et al. [103]	Population-based	no	yes
Moore and Chapman [110]	Population-based	no	no
Parsopoulos et al. [118]	Population-based	no	yes
Reyes and Coello [129]	Pareto Ranking	yes	yes
Tripathi [157]	Roulette Wheel	yes	yes

Table 4.1: Features of different multi-objective particle swarm optimizers available in the specialized literature.

The appropriate selection of leaders is essential for the good performance of a MOPSO. If the particle chooses an inappropriate leader (i.e., a leader who is too far away in the search space) then most of the flight will be fruitless because the particle will not be visiting promissory regions of the search space.

4.2.2 Proposed algorithm based on PSO

Our proposed approach is a hybrid composed by two different parts. The first part uses surrogates (it will be discussed in Chapter 5), in particular support vector machines (SVM) (see Section 5.2.5). The second part uses PSO to optimize the approximation obtained by the surrogate model and its main purpose is to produce a reasonably good approximation of the true Pareto front using a meta-model.

The surrogate model adopted in this work is shown in Figure 4.6. A multi-objective particle swarm optimizer (MOPSO) is adopted to optimize the approximate model generated by the surrogate (using support vector machines). Our MOPSO maintains two populations: the main one (which is used to select the parents), and a second population that retains the global nondominated solutions.

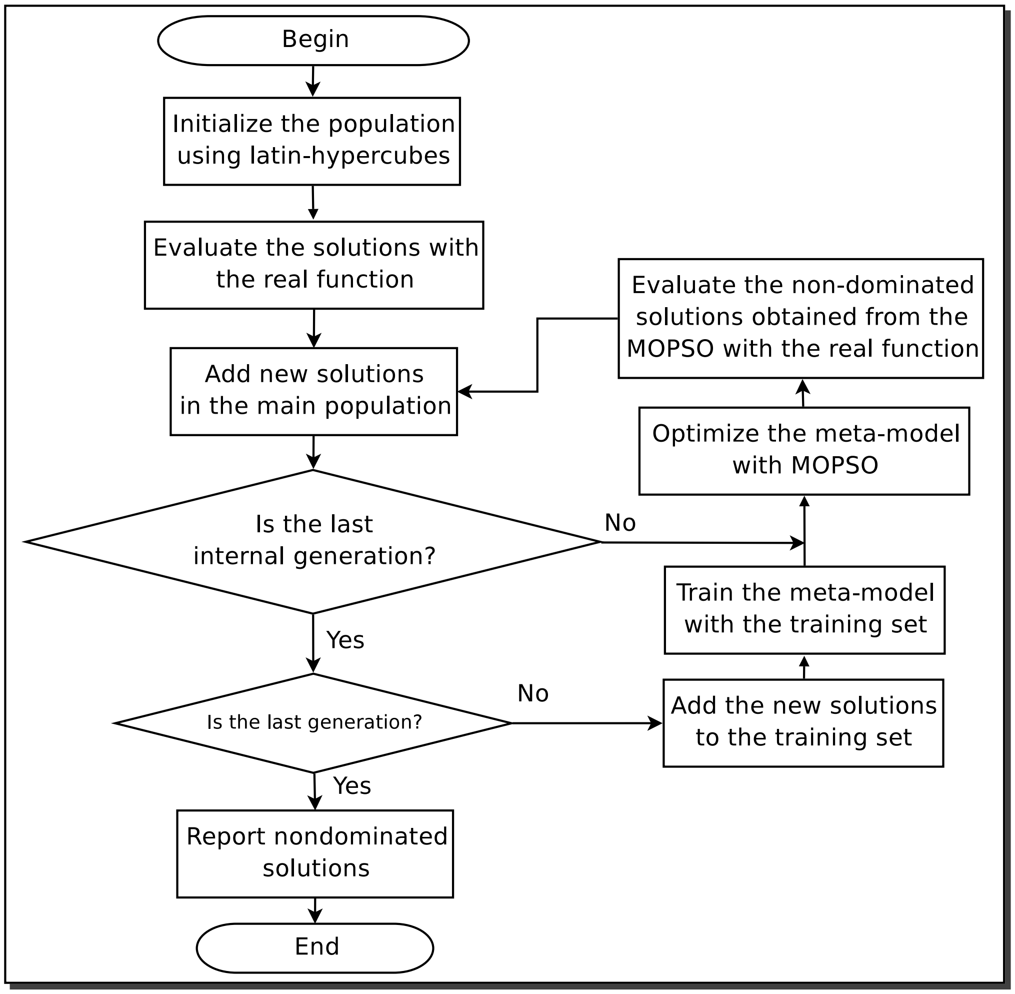

Begin
Initialize the population using latin-hypercubes
Evaluate the solutions with the real function
Add new solutions in the main population
Is the last internal generation?
No
Yes
Is the last generation?
No
Yes
Report nondominated solutions
End
Evaluate the non-dominated solutions obtained from the MOPSO with the real function
Optimize the meta-model with MOPSO
Train the meta-model with the training set
Add the new solutions to the training set

Figure 4.6: Flowchart of the algorithm adopted.

First, we generate P individuals using Latin Hypercubes [104] (see Section 5.3.1), which guarantees a good distribution of the initial population in a multidimensional space. Then, we evaluate these P individuals with the real functions, and train the meta-model using the surrogate model.

All the nondominated solutions found by the MOPSO, are evaluated with the real function and added to the main population. Once all the points are in the main population, they are used to re-train the meta-model and get another approximation of the real objectives. As it is shown in Figure 4.6, the first phase procedure contains an internal cycle which evaluates using the surrogate model instead of the real function evaluation. Once the internal cycle finishes the evolution (with the PSO) of the surrogate model, the nondominated solutions are evaluated with the real fitness function and added to the main population. After that, the training set is updated and the meta-model is re-trained with the new solutions to get a new (and hopefully better) approximation. This procedure is repeated until the $MaxGen$ number of generations is reached. We use an external population which contains the nondominated solutions and use the $pa\epsilon$-dominance grid proposed in [68] (see Chapter 7) to maintain diversity.

Our MOEA is based on the PSO algorithm [84], which uses a leader selection based on the g_{best} model, in which we choose the leader particle from the nondominated set. We also add a turbulence operator in order to jump into search regions that the PSO flight equation is not able to reach. Replacing the comparison operator (to determine whether a solution is better than other solution) is a natural modification to a PSO algorithm aimed to handle multiple objectives.

The analogy of PSO with EAs makes evident the notion that using a Pareto ranking scheme [59] could be the straightforward way to extend the approach to handle multiple objectives. However, if we merge a Pareto ranking scheme with the PSO algorithm, a set of nondominated solutions will be produced (by definition, all nondominated solutions are equally good). Having several nondominated solutions implies the inclusion into the algorithm of both: an additional criteria to decide whether a new nondominated solution is p_{best} or g_{best} and a strategy to select the guide particles (p_{best} and g_{best}).

However, the selection of an “appropriate” leader becomes a difficult task, since there can be more than one leader in the g_{best} set. Therefore, an additional strategy to select one of the multiple g_{best} to use in the PSO’s velocity equation is still necessary. Some possible leader selection strategies are described next, and are shown in Figure 4.7.

a. Random and dominator: the leader is randomly selected and preferably the leader chosen should dominate the reference particle; if that is not the case then a random leader is selected.

b. Random: a leader is randomly selected - no constraints are imposed on what sort of leader can a particle choose.

c. Closest: a particle picks up as its leader to the geographically closest leader.

d. Farthest: a particle picks up as its leader to the geographically farthest solution in the set.

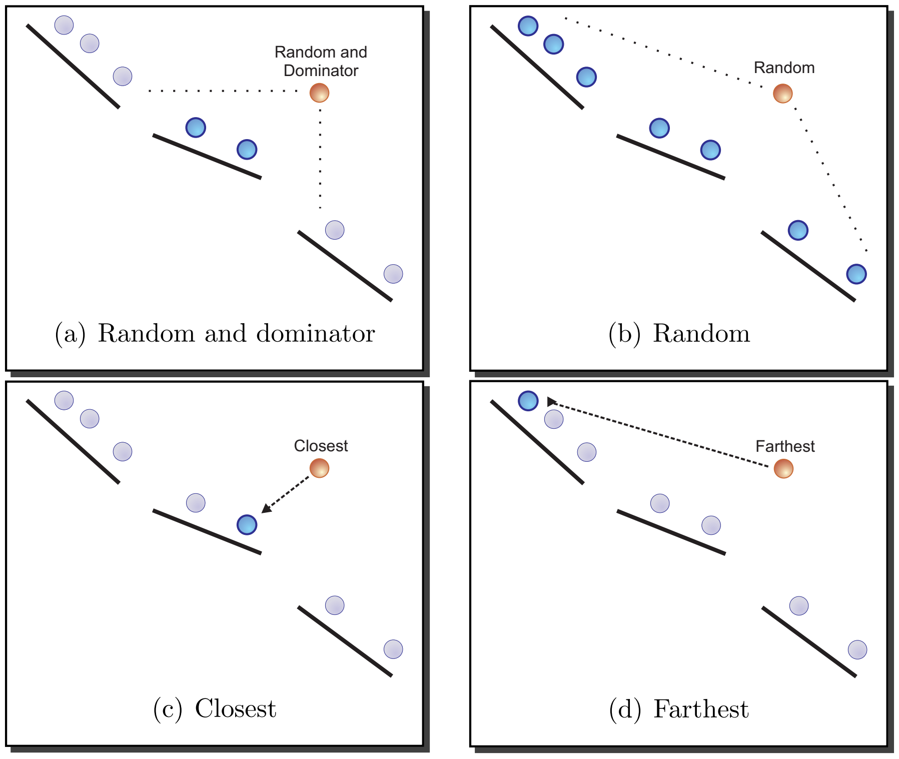


Figure 4.7: Different leader selection strategies available for a MOPSO.

4.2.3 MOPSO results

Our experimental design considers that only a few function evaluations are performed in several multi-dimensional test problems from the **ZDT** [173] test suite (see Appendix A for further information). The problems in this set are bi-objective, unconstrained and have between 10 and 30 decision variables each. Three performance measures were adopted in order to allow a quantitative assessment of our results (see Section 3.3 for further information): (1)

Inverted generational distance (**IGD**), (2) Two set coverage (**SC**), and (3) Spread (Δ). For each test problem, 10 independent runs were performed.

We ran a comparative analysis with only 600 real function evaluations, comparing the leader selection in the PSO: (1) randomly chosen dominator, (2) randomly, (3) closest and (4) farthest.

The parameters of our approach were: main population size $P = 100$, maximum number of evaluations = 600, MOPSO's internal population size ($P_{mopso} = 100$), maximum number of generations ($G_{mopso} = 100$), PSO flight equation (w= 0.1, $c_1 = 0.1$ and $c_2 = 1.4$), mutation_rate = $1/n$ (n = number of decision variables). From the parameters, we can observe that we prefer the g_{best} model from the PSO, this means that we prefer that the new particle follows the g_{best} particle instead of the p_{best} particle by giving a value of $c_1 = 0.1$ and $c_2 = 1.4$.

The results are reported in Tables 4.2 and 4.3 correspond to the performance measures adopted (IGD, Δ and SC). We show in boldface the best mean values per test function of 10 independent runs performed using each of the leader selection models compared. We show the plot of all the non-dominated solutions generated by a single run of the different algorithms in Figure 4.8. In all cases, we generated the true Pareto front, so we could make a graphical comparison of the quality of the solutions produced by our approach. The summary of our results is the following:

MOPSO-1 (randomly chosen dominator): When we select the leader that usually dominates the particle, the algorithm shows a good performance in general, obtaining the best results for ZDT1, ZDT2 and ZDT3 in IGD and Spread. Graphically, it can be seen that it obtained a good approximation to the true Pareto front in ZDT1, ZDT2 and ZDT3.

MOPSO-2 (randomly): When we use the random selection from the non-dominated set, the performance of the algorithm is acceptable, showing the best performance in ZDT4 and ZDT6 with respect the other approaches. Graphically, it can be seen that is not able to produce better results than other approaches and that this approach stays far away from the true Pareto front.

MOPSO-3 (closest): When we select the closest particle as a leader from the g_{best} set, the algorithm shows a poor performance in general, except for ZDT4, in which outperforms the other schemes.

MOPSO-4 (farthest): When we select the farthest particle as the leader from the g_{best} set, we intend to help the algorithm to maintain diversity in the population. However, this seems to cause that the algorithm cannot approach the Pareto front fast enough. Consequently, the performance of this scheme is very poor, obtaining good results only in ZDT6.

From this analysis, we can see that in ZDT4 the performance was very poor in general, getting trapped in one of the multiple local optima that this problem contains, but in the other approaches the result is not bad at all if we consider that only 600 real evaluations are performed. So, if we need to choose only one leader selection scheme, the best compromise seems to be MOPSO-1 (randomly chosen dominator). This was, therefore, our choice, and this MOPSO was then hybridized with the rough sets in order to spread the solutions that it generated. The results produced by our hybrid approach are compared with respect to the NSGA-II, which is a MOEA representative of the state-of-the-art in the area and the results are presented in Chapter 8. Also, in Chapter 5 we perform a comparative study using different surrogate methods to approximate the functions using the PSO algorithm described in this section.

4.3 Final remarks

We have introduced an approach that uses differential evolution to solve both unconstrained and constrained multi-objective optimization problems. The high convergence rate that characterizes the differential evolution algorithm was controlled using two elitist selection schemes. The constraint-handling scheme adopted in our algorithm allowed a successful exploration of the search space even in the presence of problems whose optimum lies on the boundary between the feasible and infeasible regions. Later in the chapter, we used four different schemes to select a leader in the PSO flight equation for multi-objective problems in combination with the surrogate model to approximate the functions using supervised learning. From the initial comparison, we concluded that the PSO scheme based on selecting a leader that dominates the particle was the most appropriate model to use as the search engine, because it provides a good (although insufficient) approximation of the Pareto front. Finally, the use of ϵ-dominance introduced the capability of controlling the convergence of our approach while achieving a good spread of solutions. We decided to use both heuristics (DE and PSO) because they have both shown a high convergence rate to the optimum (or its vicinity) in single and multi-objective optimization, and both have a performance that relies on few parameters, which facilitates their use to get a good initial approximation of the Pareto front. Several papers have been derived from this chapter (see [66, 136, 137, 138]).

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT1	IGD	MOPSO-1	**0.0122**	0.0039	0.0074	0.0199
		MOPSO-2	0.0178	0.0034	0.0134	0.0222
		MOPSO-3	0.0201	0.0013	0.0176	0.0227
		MOPSO-4	0.0163	0.0037	0.0117	0.0219
	Δ	MOPSO-1	**0.6044**	0.0709	0.5160	0.7291
		MOPSO-2	0.6730	0.0677	0.5402	0.7744
		MOPSO-3	0.7287	0.0373	0.6782	0.7936
		MOPSO-4	0.6598	0.0502	0.5312	0.7179

SC(A, B)		B				
		MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
A	MOPSO-1	–	0.9417	0.8829	0.6598	**0.8281**
	MOPSO-2	0.0164	–	0.5385	0.3290	0.2891
	MOPSO-3	0.0817	0.3462	–	0.2646	0.1731
	MOPSO-4	0.1000	0.5343	0.6225	–	0.3856

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT2	IGD	MOPSO-1	**0.0346**	0.0048	0.0200	0.0365
		MOPSO-2	0.0357	0.0003	0.0354	0.0362
		MOPSO-3	0.0366	0.0024	0.0353	0.0437
		MOPSO-4	0.0356	0.0003	0.0348	0.0361
	Δ	MOPSO-1	**0.1832**	0.3679	0.0000	0.9892
		MOPSO-2	0.2960	0.4522	0.0000	0.9981
		MOPSO-3	0.2993	0.4572	0.0000	1.000
		MOPSO-4	0.3985	0.4881	0.0000	0.9995

SC(A, B)		B				
		MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
A	MOPSO-1	–	0.2750	0.2500	0.0000	0.1750
	MOPSO-2	0.6300	–	0.7500	0.3500	0.3667
	MOPSO-3	0.6200	0.200	–	0.1000	0.1000
	MOPSO-4	0.9200	0.6000	0.9000	–	**0.500**

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT3	IGD	MOPSO-1	**0.0287**	0.0099	0.0170	0.0462
		MOPSO-2	0.0351	0.0087	0.0221	0.0489
		MOPSO-3	0.0439	0.0109	0.0329	0.0742
		MOPSO-4	0.0314	0.0062	0.0197	0.0397
	Δ	MOPSO-1	**0.7178**	0.0718	0.5638	0.8179
		MOPSO-2	0.7544	0.0263	0.7081	0.8016
		MOPSO-3	0.7810	0.0222	0.7307	0.8127
		MOPSO-4	0.7582	0.0797	0.6055	0.8405

SC(A, B)		B				
		MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
A	MOPSO-1	–	0.6286	0.7344	0.5153	**0.6261**
	MOPSO-2	0.2381	–	0.6835	0.2799	0.3211
	MOPSO-3	0.2025	0.1711	–	0.1446	0.1052
	MOPSO-4	0.3489	0.4912	0.7011	–	0.3974

Table 4.2: Results of inverse generational distance (IGD), spread (Δ) and set coverage (SC) for the ZDT1, ZDT2 and ZDT3 test problems.

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT4	IGD	MOPSO-1	1.2384	0.2204	0.8144	1.6364
		MOPSO-2	1.1121	0.2442	0.8715	1.6318
		MOPSO-3	**1.0162**	0.1538	0.7503	1.2818
		MOPSO-4	1.2157	0.2108	0.8910	1.6635
	Δ	MOPSO-1	0.9806	0.0279	0.9234	1.0338
		MOPSO-2	0.9802	0.0135	0.9614	0.9981
		MOPSO-3	**0.8560**	0.2890	0.0000	1.0516
		MOPSO-4	0.8786	0.2932	0.0000	0.9968

SC(A, B)		B				
		MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
A	MOPSO-1	–	0.1821	0.2850	0.2857	0.2509
	MOPSO-2	0.7333	–	0.6217	0.5000	**0.3739**
	MOPSO-3	0.6583	0.3815	–	0.6000	0.3271
	MOPSO-4	0.6333	0.4125	0.3000	–	0.2375

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT6	IGD	MOPSO-1	0.0351	0.0028	0.0285	0.0405
		MOPSO-2	**0.0327**	0.0028	0.0286	0.0371
		MOPSO-3	0.0339	0.0034	0.0286	0.0377
		MOPSO-4	0.0330	0.0026	0.0271	0.0361
	Δ	MOPSO-1	0.9494	0.1284	0.7218	1.1933
		MOPSO-2	0.8850	0.1057	0.6669	1.0961
		MOPSO-3	0.8998	0.0944	0.7791	1.1026
		MOPSO-4	**0.8715**	0.0508	0.7921	0.9764

SC(A, B)		B				
		MOPSO-1	MOPSO-2	MOPSO-3	MOPSO-4	Mean
A	MOPSO-1	–	0.3560	0.3526	0.3286	0.3457
	MOPSO-2	0.4494	–	0.3536	0.2087	0.1874
	MOPSO-3	0.4333	0.3176	–	0.3306	0.2160
	MOPSO-4	0.5494	0.5501	1.0	–	**0.5167**

Table 4.3: Results of inverse generational distance (IGD), spread (Δ) and set coverage (SC) for the ZDT4 and ZDT6 test problems.

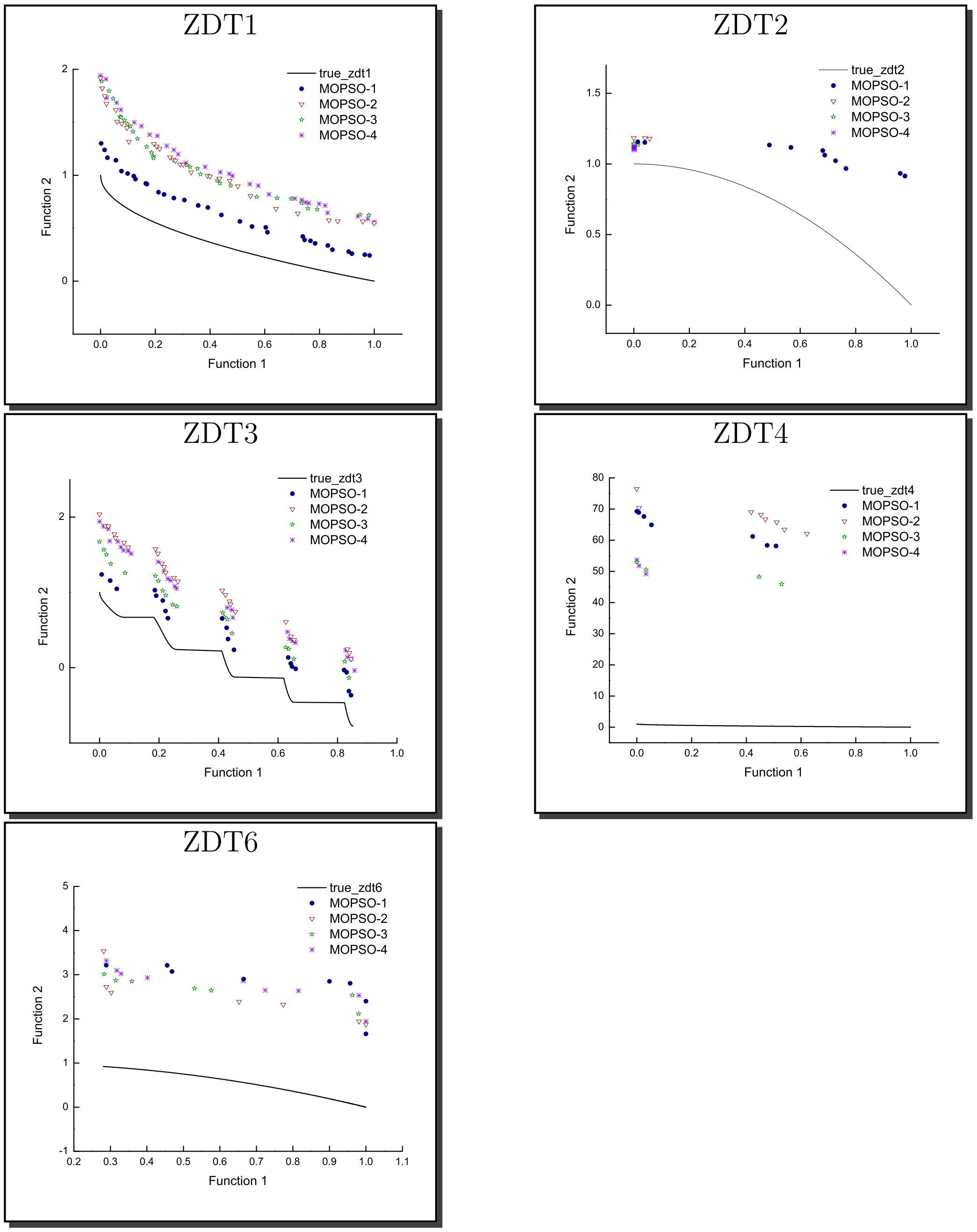


Figure 4.8: Pareto fronts generated by the surrogates methods for the ZDT test problems.

5 Convergence: Fitness Approximation

This chapter describes some of the possible schemes by which knowledge can be incorporated into an evolutionary algorithm, with a particular emphasis on MOEAs. The taxonomy of approaches that we will cover in this chapter is shown in Figure 5.1. In Section 5.2, we discuss schemes that incorporate knowledge into the fitness evaluations of an evolutionary algorithm. The three schemes normally adopted (problem approximation, functional approximation, and evolutionary approximation) are all discussed in this chapter. We also provide a brief explanation of the surrogate models that we decided to use in this thesis: 1) radial basis functions, 2) artificial neural networks, and 3) support vector machines. Finally, a comparison of the surrogate models is presented with a low number of fitness function evaluations and an analysis of them is presented.

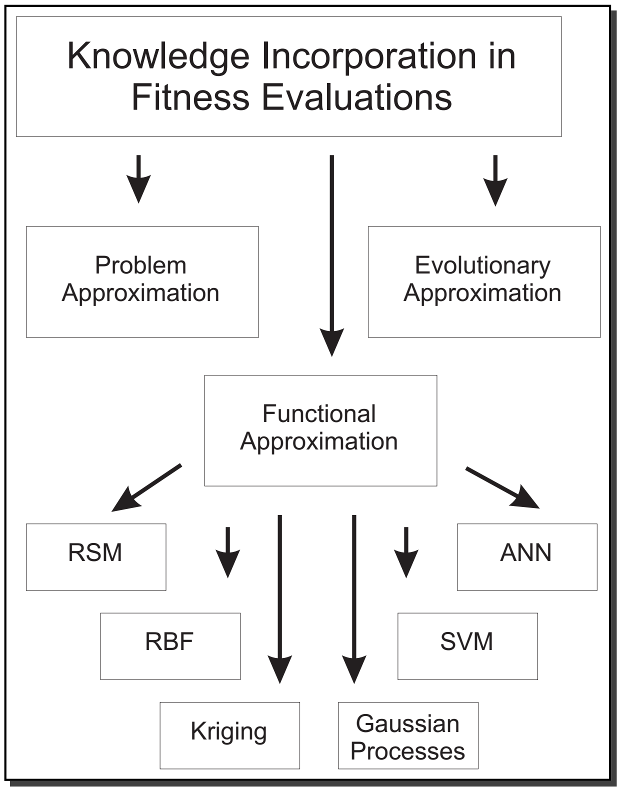


Figure 5.1: A taxonomy of approaches for incorporating knowledge into evolutionary algorithms.

5.1 Knowledge incorporation

The high number of fitness evaluations often required by evolutionary algorithms is normally expensive, time-consuming and problematic in many real-world applications. Particularly in the following cases, a computationally efficient approximation of the original fitness function reducing either the number or duration of the fitness evaluations, is necessary:

- If the evaluation of the fitness function is computationally expensive.
- If the fitness function cannot be defined in an algebraic form (e.g., when the fitness function is generated by a simulator).
- If additional physical devices must be used to determine the fitness function and this requires human interaction.

- If parallelism is not allowed.
- If the total number of evaluations of the fitness function is limited by financial constraints.

There exist some cases in which the approximation is used to predict promising new solutions at a smaller evaluation cost than that of the original problem. Jin [81] discusses various approximation levels or strategies adopted for fitness approximation:

Problem approximation: Tries to replace the original statement of the problem by one which is approximately the same as the original problem but which is easier to solve. To save the cost of experiments, numerical simulations instead of physical experiments are used to pseudo-evaluate the performance of a design.

Functional approximation: In this approximation, a new expression is constructed for the objective function based on previous data obtained from the real objective functions. In this case, models obtained from data are often known as meta-models or surrogates (see Section 5.2)

Evolutionary approximation: This approximation is specific for EAs and tries to save function evaluations by estimating an individual's fitness from other similar individuals. A popular subclass in this category is known as fitness inheritance.

Currently, there exist several evolutionary algorithms that use a meta-model to approximate the real fitness function and reduce the total number of fitness evaluations without degrading the quality of the results obtained. To achieve this goal, meta-models should be combined with the original fitness function in a proper manner. The mechanism adopted to balance the use of the meta-model and the real objective function is known as *evolution control*. Evolution control takes an important role when meta-models are combined with the original fitness function. In such cases, most meta-models could converge to a local optimum if they are provided incorrect knowledge (or information) about the real function. There are two different forms to combine the approximated model and the real function:

Individual-based evolution control: In this case, some individuals use meta-models to evaluate their fitness value and others (in the same generation) use the real fitness function. The main issue in individual-based evolution control is to determine which individuals should use the meta-model and which ones should use the real fitness function during the present generation. They can be randomly chosen from the current population, or one could simply choose the best individuals in the population to be evaluated using the real function. In Figure 5.2, it can be seen that from the same population: three individuals are evaluated with the real function and other three individuals are evaluated with the meta-model

Generation-based evolution control: The main issue in generation-based evolution control is to determine in which generations the meta-model should be used and in which generations the real fitness function should be used. In this control, the real fitness function is applied at every i generations, where i is predefined and fixed throughout the evolutionary process. In Figure 5.3, we can see that all the solutions from the i generation are evaluated with the real function and the solutions from the $i-1$ and $i+1$ generations are evaluated with the meta-model.

In the above cases, the approximation is used to predict promising new solutions at a smaller evaluation cost than that of the original problem. Current functional approximation models include Polynomials (e.g., response surface methodologies [126, 58]), neural networks (e.g., multi-layer perceptrons (MLPs) [71, 73, 121]), radial-basis-function (RBF) networks [115, 158, 169], support vector machines (SVMs) [144, 11], Gaussian processes [159, 15], and Kriging [45, 127]; all of them can be used for constructing meta-models.

5.2 Surrogates

Surrogate models can perform a number of tasks in support of a computational analysis. Through interpolation, extrapolation and/or integration, these models can be used to address complex problems involving experimental design, system analysis and prediction.

In a single-objective optimization context, surrogate models have been successful in dealing with highly demanding problems where the cost of evaluating the real fitness function is very expensive (computationally speaking).

The accuracy of the surrogate model relies on the number of samples provided in the search space, as well as on the selection of the appropriate

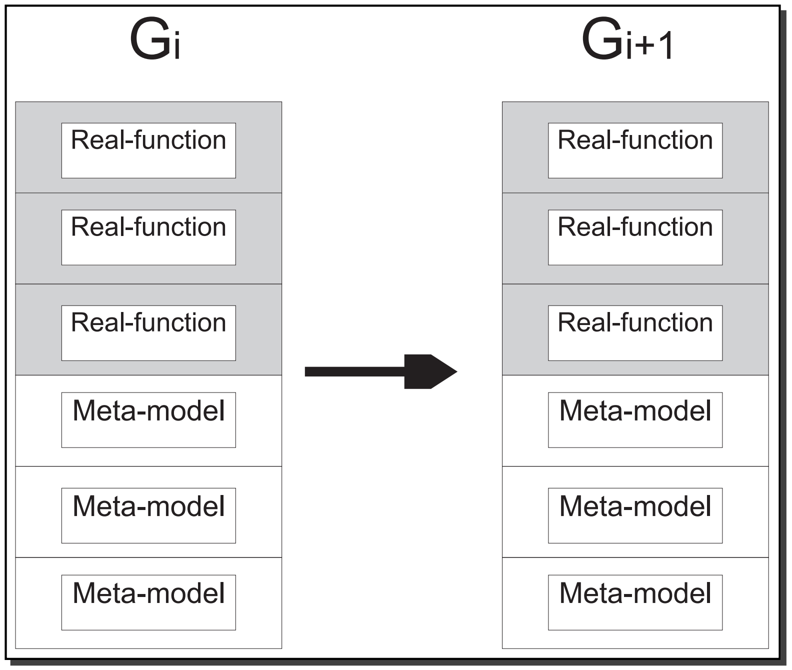


Figure 5.2: Individual-based evolution control.

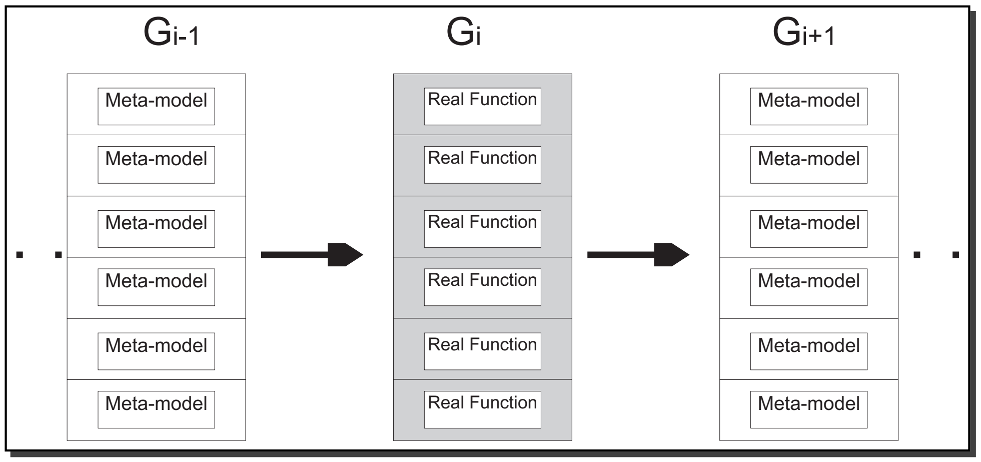


Figure 5.3: Generation-based evolution control.

model to represent the objective functions. There exist a variety of techniques for constructing surrogate models (see for example [160]). One approach is least-square regression using low-order polynomials, also known as response surface methods. A statistical alternative for constructing surrogate models is Kriging, which is also referred to as "Design and Analysis of Computer Experiments" (DACE) models [135] and Gaussian process regression [166]. Comparisons of several surrogate modeling techniques are presented by Giunta and Watson [56] and by Jin et al. [80].

A surrogate model is built when the objective functions are to be estimated. This local model is built using a set of data points that lie on the local neighborhood of the design. Since surrogate models will probably be built thousands of times during the search, computational efficiency is the main objective.

Chafekar et al. [17] proposed a multi-objective evolutionary algorithm called OEGADO, which runs several Genetic Algorithms (GAs) concurrently with each GA optimizing one objective function at a time, and forming a reduced model (based on quadratic approximation functions) with this information. At regular intervals, each GA exchanges its reduced model with the others. This technique can solve constrained optimization problems in 3,500 and 8,000 evaluations and is compared with respect to the NSGA-II [39] and the $\epsilon - MOEA$ [37, 38].

Emmerich et al. [46] presented a local Gaussian random field meta-model (GRFM) to predict objective function values by exploiting information obtained during previous evaluations. This scheme was created for optimizing computationally expensive problems. This method selects the most promising population members at each generation so that they are evaluated using the real objective function. This approach was tested on a 10 dimensional airfoil optimization problem and was compared with respect to the NSGA-II in the generalized Schaffer problems.

5.2.1 Polynomials: response surface methods (RSM)

The response surface methodology comprises regression surface fitting in order to obtain approximate responses, design of experiments in order to obtain minimum variances of the responses and optimizations using the approximated responses.

An advantage of this technique is that the fitness of the approximated response surfaces can be evaluated using powerful statistical tools. Additionally, the minimum variances of the response surfaces can be obtained using design of experiments with a small number of experiments.

For most response surfaces, the functions adopted for the approximations are polynomials because of their simplicity, although other types of functions are, of course, possible. For the cases of quadratic polynomials, the response surface is described as follows:

$$\hat{y} = (\beta_0) + \sum_{i=1}^{n} (\beta_i \cdot x_i) + \sum_{i,j=1, i \leq j}^{n} (\beta_{i,j} \cdot x_i \cdot x_j) \tag{5.1}$$

where n is the number of variables, and β_0 and β_i are the coefficients to be calculated. To estimate the unknown coefficients of the polynomial model, both the least squares method (LSM) and the gradient method can be used, but either of them requires at least the same number of samples of the real objective function than the β_i coefficients in order to obtain good results.

Goel et al. [58] is one of the few works that has used RSM in multi-objective problems. In this report, the NSGA-II [39] and a local search strategy called "$\epsilon - constraint$" are adopted to generate a solution set that is used for approximating the Pareto optimal front by a response surface method (RSM). This method is applied to a rocket injector design problem.

There are few applications of the use of surrogates in evolutionary multi-objective optimization. Two of them are briefly discussed next.

Bramanti et al. [14] tried to reduce the computational cost of a multi-objective evolutionary algorithm using neural networks interpolation for building an objective response surface in order to find multiple trade-off solutions in electromagnetic design problems.

Wilson et al. [167] used two types of surrogate approximations (response surfaces and Kriging models) in order to reduce the computational expense of designing piezomorph actuators. The method shows that is flexible and can accommodate a wide variety of experimental designs and approximations. The authors also show that this method works well for both convex and non-convex Pareto fronts.

5.2.2 Radial basis functions

Radial Basis Functions (RBFs) were first introduced by R. Hardy in 1971 [64]. This term is made up of two different words: *radial* and *basis functions*. Each of these terms will be explained next:

$$g : \mathbb{R}^d \to \mathbb{R} : (x_1, \dots, x_d) \mapsto \phi(\|x_1, \dots, x_d\|_2)$$

for some function $\phi : \mathbb{R} \to \mathbb{R}$. This means that the function value of g at a point $\overrightarrow{x} = (x_1, \dots, x_d)$ only depends on the Euclidean norm of $\overrightarrow{x}$:

$$\|\overrightarrow{x}\|_2 = \sqrt{\sum_{i=0}^{d} x_i^2} = \text{distance of } \overrightarrow{x} \text{ to the origin}$$

And this explains the term *radial.* The term *basis function* is explained next. Let's suppose we have certain points (called centers) $\overrightarrow{x}_1, \dots, \overrightarrow{x}_n \in \mathbb{R}^d$. The linear combination of the function g centered at the points $\overrightarrow{x}$ is given by:

$$f : \mathbb{R}^d \mapsto \mathbb{R} : \overrightarrow{x} \mapsto \sum_{i=1}^{n} \lambda_i g(\overrightarrow{x} - \overrightarrow{x_i}) = \sum_{i=1}^{n} \lambda_i \phi(\|\overrightarrow{x} - \overrightarrow{x_i}\|) \tag{5.2}$$

where $\|\overrightarrow{x} - \overrightarrow{x_i}\|$ is the Euclidean distance between the points $\overrightarrow{x}$ and $\overrightarrow{x}_i$. So, f becomes a function which is in the finite dimensional space spanned by the basis functions:

$$g_i : \overrightarrow{x} \mapsto g(\|\overrightarrow{x} - \overrightarrow{x_i}\|)$$

Now, let's suppose that we already know the values of a certain function $H : \mathbb{R}^d \mapsto \mathbb{R}$ at a set of fixed locations $\overrightarrow{x_i}, \dots, \overrightarrow{x_n}$. These values are named $f_j = H(\overrightarrow{x_j})$, so we try to use the $\overrightarrow{x_j}$ as centers in the equation 5.2. If we want to force the function f to take the values f_j at the different points $\overrightarrow{x_j}$, then we have to put some conditions on the λ_i. This implies the following:

$$\forall j \in \{1, \dots, n\} \; f_j = f(\overrightarrow{x_j}) = \sum_{i=1}^{n} (\lambda_i \cdot \phi(\|\overrightarrow{x_j} - \overrightarrow{x_i}\|))$$

In these equations, only the λ_i are unknown, and the equations are linear

Type of Radial Function		
LS	linear splines	$\lvert r\rvert$
CS	cubic splines	$\lvert r\rvert^3$
MQS	multiquadrics splines	$\sqrt{1+(\epsilon r)^2}$
TPS	thin plate splines	$\lvert r\rvert^{2m+1}\ln\lvert r\rvert$
GA	Gaussian	$e^{-(\epsilon r)^2}$

Table 5.1: Radial basis functions

in their unknowns. Therefore, we can write these equations in matrix form:

$$\begin{bmatrix} \phi(0) & \phi(\|x_1-x_2\|) & \dots & \phi(\|x_1-x_n\|) \\ \phi(\|x_2-x_1\|) & \phi(0) & \dots & \phi(\|x_2-x_n\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|x_n-x_1\|) & \phi(\|x_n-x_2\|) & \dots & \phi(0) \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{5.3}$$

Typical choices for the basis function $g(\vec{x})$ include linear splines, cubic splines, multiquadrics, thin-plate splines and Gaussian functions as shown in Table 5.1.

Ong et al. [114] used surrogate models (RBFs) to solve computationally expensive design problems with constraints. The authors used a combination of a parallel evolutionary algorithm coupled with sequential quadratic programming in order to find optimal solutions of an aircraft wing design problem. In this case, the authors construct a local surrogate model based on radial basis functions in order to approximate the objective and constraint functions of the problem.

Karakasis et al. [82] used surrogate models based on radial basis functions in order to deal with computationally expensive problems. A method called inexact pre-evaluation (IPE) is applied into a MOEA's selection mechanism. Such method helps to choose the individuals that are to be evaluated using the real objective function, right after a meta-model approximation has been obtained by the surrogate. The results are compared against a conventional MOEA in two test-problems, one from a benchmark and one from the turbomachinery field.

Voutchkov & Keane [74] studied several surrogate models (RSM, RBF and Kriging) in the context of multi-objective optimization using the NSGA-

II [39] as the MOEA that optimized the meta-model function given by the surrogate. The surrogate model is trained with 20 initial points and the NSGA-II is run on the surrogate model. Then, the 20 best resultant points given by the optimization are added to the existing data pool of real function evaluations and the surrogate is re-trained with these new solutions. A comparison of results is made in 4 test functions (from 2 to 10 variables), performing 400 real fitness function evaluations.

5.2.3 Kriging

In Kriging, the meta-model prediction is formed by adding up two different models as follows:

$$y(\overrightarrow{x}) = a(\overrightarrow{x}) + b(\overrightarrow{x})$$

where $a(\overrightarrow{x})$ represents the "average" long-term range behavior and the expected value of the true function. This function can be modeled in various ways, such as with polynomials or with trigonometric series as:

$$a(\overrightarrow{x}) = a_0 + \sum_{i=1}^{L}\sum_{j=1}^{R} a_{ij}(x_i)^j$$

where: R is the polynomial order with L dimensions and $b(\overrightarrow{x})$ stands for a local deviation term. $b(\overrightarrow{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. This function represents a short-distance influence of every data point over the global model. The general formulation for $b(\overrightarrow{x})$ is a weighted sum of N functions, $K_n(\vec{x})$ that represent the covariance functions between the n^{th} data point and any point $\vec{x}$:

$$b(\overrightarrow{x}) = \sum_{n=1}^{N} b_n K(h(x, x_n)) \text{ and } h(x, x_n) = \sqrt{\sum_{i=1}^{L}\left(\frac{x_i - x_{in}}{x_i^{max} - x_i^{min}}^2\right)}$$

where x_i^{min} and x_i^{max} are the lower an upper bounds of the search space and x_{in} denotes the $i-th$ component of the data point x_n. However, the shape of $K(h)$ has a strong influence on the resulting aspect of the statistical model. And that is why it is said that Kriging is used as a estimator or an interpolator.

Knowles [86] proposed “ParEGO”, which consists of a hybrid algorithm based on a single optimization model (EGO) and a Gaussian process, which is updated after every function evaluation, coupled to an evolutionary algorithm. EGO is a single-objective optimization algorithm that uses Kriging to model the search landscape from the solutions visited during the search and learns a model based on Gaussian processes (called DACE). This approach is used to solve multi-objective optimization problems of low dimensionality (up to 6 decision variables) with only 100 and 250 fitness function evaluations.

5.2.4 Artificial neural networks

Artificial neural network (ANN) implemented through a multilayer preceptron is a flexible scheme capable of approximating an arbitrary complex function [13]. An ANN basically builds a map between a set of inputs and the respective outputs and are good to deal with nonlinear regression analysis with noisy signals. A multilayer feedforward neural network consists of an array of input nodes connected to an array of output nodes through successive intermediate layers. Each connection between nodes has a weight, which initially has a random value, and that is adjusted during a training process. The output of each node of a specific layer is a function of the sum on the weighted signals coming from the previous layer. The crucial points in the construction of an ANN are the selection of inputs and outputs, the architecture of the ANN, that is, the number of layers and the number of nodes in each layer, and finally, the training algorithm.

The multi-layer perceptron (MLP) is a multilayered feedforward network that has been widely used in function approximation problems, because it has been often found to provide compact representations of mappings in real-world problems. An MLP is composed of neurons and the output (y) of each neuron is thus:

$$y = \phi \left(\sum_{i=1}^{n} w_i \cdot a_i + b \right)$$

where a_i are the inputs of the neuron, and w_i is the weight associated with the i^{th} input. The nonlinear function ϕ is called the activation function as it determines the activation level of the neuron.

In Figure 5.4, we show an MLP network with one layer of linear output neurons and one layer of nonlinear neurons between the input and output

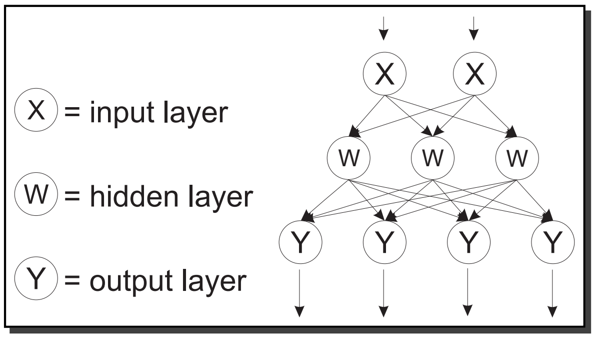


Figure 5.4: A graphical representation of an MLP network with one hidden layer

neurons. The middle layers are usually called *hidden layers*.

To learn a mapping $\mathbb{R}^n \rightarrow \mathbb{R}^m$ by an MLP, its architecture should be the following: it should have n input nodes and m output nodes with a single or multiple hidden layer. The number of nodes in each hidden layer are generally a design decision.

5.2.4.1 Training an ANN

In general terms, supervised training consists of presenting to the network patterns whose output we know (the training set) finding the output of the net and adjusting the weights so as to make the actual output more like the desired (or teaching signal). The two most useful training protocols are: off-line and on-line. In off-line learning, all the data are stored and can be accessed repeatedly. In on-line learning, each case is discarded after it is processed and the weights are updated. With off-line learning, we can compute the objective function for any fixed set of weights, so we can see whether we are making progress in training.

Error back-propagation is the simplest and most widely used algorithm to train feedforward neural networks. In this algorithm the training is performed by minimizing a loss function, usually the sum of square errors over the N elements of the training set. In this case we used a generalization of the square error function given by:

$$J(W) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{c}(t_{ki} - z_{ki})^2 = \frac{1}{2}\sum_{i=1}^{N}||\overrightarrow{t_i} - \overrightarrow{z_i}||^2$$

where $\vec{t_i}$ and $\vec{z_i}$ are the i^{th}-target and the i^{th}-network output vectors of length c, respectively; W represents all the weights in the network. The backpropagation learning rule is based on a gradient descent. The weights are initialized with random values, and are changed in a direction to reduce the error following the next rule:

$$W_{new} = W_{old} - \eta \frac{\partial J}{\partial W}$$

The weight update for the hidden-output weights is given by:

$$\partial W_{kj} = \eta(t_k - z_k) f'(net_k) y_j$$

and the input-to-hidden weights learning rule is:

$$\partial W_{ji} = \eta \cdot x_i \cdot f'(net_j) \sum_{k=1}^{n} w_{kj} \partial_k$$

where η is the learning rate, i, j, k are the corresponding node index for each layer and net_j is the inner product of the input layer with the weights w_{ji} at the hidden unit.

Some applications of ANNs to evolutionary multi-objective optimization are the following:

Farina [48] proposed “NN-GRS”, which is an extension of the single-objective neural network-based GRS (generalized response surface) method. The main idea of this approach is to maintain two different objective functions, one real and another one which is an approximation (neural network based).

Nain & Deb [113] proposed “NSGA-II-ANN”, which combines the NSGA-II algorithm [39] with a new method based on neural networks as the basic approximation technique for fitness computation. This meta-model is updated at each generation, and it provides a more refined approximate model to guide the search of the NSGA-II in subsequent generations.

5.2.5 Support vector machines

Support vector machines (SVM) have become popular in recent years for solving problems in classification, regression and novelty detection. An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and thus, any local solution is also a global optimum. In SVM regression, our goal is to find a function $f(x)$ that has at most an ϵ deviation from the obtained targets y_i for all the training data, and at the same time is as flat as possible. Let's suppose we are given training data $\chi = (x_t, y_t)_{t=1}^N$ where $y_t \in \mathbb{R}$. Then, the $f(x)$ is given by:

$$f(x) = \langle w, x \rangle + b \; with \; w \in \mathbb{R}^d, x \in \mathbb{R}^d, b \in \mathbb{R}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in χ. A small w means that the regression is flat. One way to ensure this, is to minimize the norm, $||w||^2 = \langle w, w \rangle$. The problem can be written as a convex optimization problem:

$$\begin{array}{ll} \text{m}inimize & \frac{1}{2}||w||^2 \\ subject\ to & \left\{ \begin{array}{ll} y_i - \langle w, x_i \rangle - b & \leq \epsilon \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon \end{array} \right. \end{array} \tag{5.4}$$

And one can introduce two slack variables ξ_i, ξ_i^*, for positive and negative deviations, where $\xi_i > 0$ corresponds to a point for which $y_i + \epsilon$ and:

$$\begin{array}{ll} \text{m}inimize & C \sum_{i=1}^{l} (\xi_i + \xi_i^*) + \frac{1}{2}||w||^2 \\ subject\ to & \left\{ \begin{array}{ll} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{array} \right. \end{array} \tag{5.5}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated. The ϵ-insensitive loss function [161](see equation 5.6) means that we tolerate errors up to ϵ and also that errors beyond that value have a linear effect and not quadratic. This error function is therefore more tolerant to noise and is thus more robust.

$$|\xi|_\epsilon = \left\{ \begin{array}{ll} 0, & \text{if } |\xi| \leq \epsilon; \\ |\xi| - \epsilon, & \text{otherwise.} \end{array} \right. \tag{5.6}$$

Figure 5.5, shows a graphic of the ϵ-insensitive loss function. Note that only the points outside the shaded region contribute to the cost of the function. It turns out that in most cases, the optimization problem defined by equation (5.5) can be solved more easily in its dual formulation. The dual formulation also provides the capability for extending SVM to nonlinear functions using a standard dualization method utilizing Lagrange multipliers, as described by Fletcher [52]. So, optimizing the Lagrangian and substitute $t_i = \langle w, x_i \rangle$ for simplicity we have:

$$\begin{aligned} L &= C\sum_{i=1}^{N}(\xi_i + \xi_i^*) + \frac{1}{2}||w||^2 - \sum_{i=1}^{N}(\mu_i\xi_i + \mu_i^*\xi_i^*) \\ &\quad - \sum_{i=1}^{N}\alpha_i(\epsilon + \xi_i + y_n - t_n) - \sum_{i=1}^{N}\alpha_i^*(\epsilon + \xi_i^* + y_n - t_n) \end{aligned} \tag{5.7}$$

Then, we can substitute for $y(x)$ using the linear model equation: $y(x) = w^T\phi(x) + b$ and set the derivatives of the Lagrangian with respect to 1) w, 2) b, 3) ξ_i and 4) ξ_i^* to zero, giving:

$$\frac{\partial L}{\partial w} = 0 \;\Rightarrow\; w = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\phi(x_i) \tag{5.8}$$

$$\frac{\partial L}{\partial b} = 0 \;\Rightarrow\; \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{5.9}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \;\Rightarrow\; \alpha_i + \mu_i = C \tag{5.10}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \;\Rightarrow\; \alpha_i^* + \mu_i^* = C \tag{5.11}$$

$$\tag{5.12}$$

Using these results to eliminate the corresponding variables from the Lagrangian, we see that the dual problem involves maximizing:

$$L'(a, a^*) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_m\rangle - \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)t_n \tag{5.13}$$

with respect to α_i and α_i^*, where $\langle x_i, x_m\rangle$ is the kernel function. So, the problem becomes in a constrained maximization problem with the box constraints:

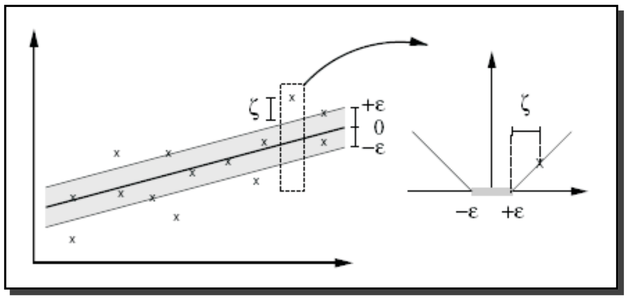


Figure 5.5: ϵ-insensitive loss function for SVM

$$0 \leq \alpha_i \leq C$$
$$0 \leq \alpha_i^* \leq C$$

And the predictions for new inputs can be made using:

$$y(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \langle x_i, x_m \rangle + b \tag{5.14}$$

The support vectors are those data points that contribute to predictions given by equation 5.14, in other words those for which either $\alpha_i \neq 0$ or $\alpha_i^* \neq 0$. These are points that lie on the boundary of the ϵ-tube or outside the tube. All points within the tube have $\alpha_i = \alpha_i^* = 0$.

5.3 Results using surrogates

In the Figure 5.6, it is shown the diagram in which we try to reduce the number of real function evaluation by training the meta-model using different function approximation techniques to approximate all the functions involved in the multi-objective optimization problem and optimizing this meta-model using the PSO algorithm (Described in Section 4.2.2). The flowchart is described in detail next:

Initialize population.- The initialization of the population will be done using Latin-Hypercubes [104] (see Section 5.3.1) because they guarantee a uniform distribution of the solutions in the search space. So, this procedure can be chosen to this purpose as our approximation model requires a good distribution of the sample points provided in order to build a good approximation of the real functions.

Real function evaluation.- All the solutions are to be evaluated using the real fitness function.

Add solutions to population.- It is important to determine which of these solutions are good and keep them to help train (or re-train) the meta-model.

Train meta-model.- The different surrogates that we used are:

- Radial Basis Function (see Section 5.2.2)
- Artificial Neural Networks (see Section 5.2.4)
- SVMs for regression (see Section 5.2.5)

to approximate the different functions with the same training points.

PSO optimization.- Once that the different meta-models are trained, we solve a multi-objective problem using a PSO-based MOEA on the meta-models instead of performing real function evaluations. As we are dealing with multi-objective problems, we decided to train the multiple objectives separately. Consequently, we obtain a different surrogate model per each objective. Thus, these objectives are still in conflict with each other as they are an approximate model of the real objectives. This means that we have another multi-objective problem based on the different surrogates obtained during the training process.

5.3.1 Latin-Hypercubes

A Latin cube [104] is a selection of one point from each row and column of a square matrix. In k dimensions, the corresponding item is a set of P points, where, in each dimension, there is exactly one point per column or range of values. In k dimensions, these objects are called Latin-Hypercubes. Once a Latin-Hypercube has been created, we choose the center of each hypercube as the place where the initial P individuals are chosen. Then, we evaluate these P individuals with the real functions, and train the meta-model using the surrogate model.

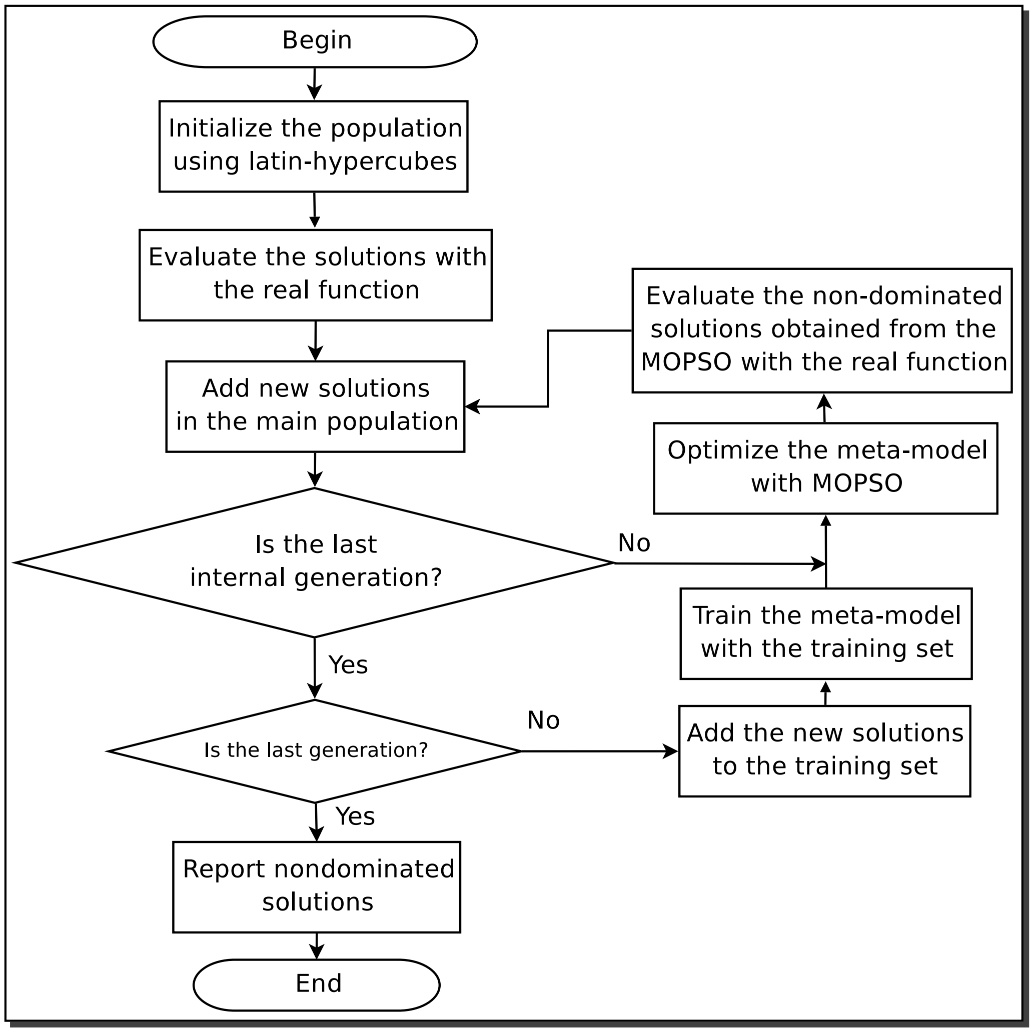

Begin
Initialize the population using latin-hypercubes
Evaluate the solutions with the real function
Add new solutions in the main population
Is the last internal generation?
No
Yes
Is the last generation?
No
Yes
Report nondominated solutions
End
Evaluate the non-dominated solutions obtained from the MOPSO with the real function
Optimize the meta-model with MOPSO
Train the meta-model with the training set
Add the new solutions to the training set

Figure 5.6: Surrogate model adopted in this section.

5.4 Comparative study

Our main goal is to reduce the number of fitness function evaluations. Thus, our experimental design considers that only a few function evaluations are performed in several multi-dimensional test problems from the **ZDT** set. So, in this case, we compare the results obtained by the surrogate-based MOEA with only 600 real function evaluations.

For our experiments, we adopted the **ZDT** test problems [173], all of which are bi-objective, unconstrained and have between 10 and 30 decision variables each. The detailed description of these test functions are in Appendix A.

Three performance measures were adopted in order to allow a quantitative assessment of our results: (1) Inverted generational distance (**IGD**), (2) Two set coverage (**SC**), and (3) Spacing (S). For each test problem, 10 independent runs were performed.

5.4.1 Surrogate phase analysis

This phase of our approach uses several parameters: main population size $P = 100$, maximum number of evaluations = 600, multi-objective particle swarm optimizer (MOPSO) internal population size ($P_{mopso} = 100$), maximum number of generations ($G_{mopso} = 100$), PSO flight equation (w= 0.1, $c_1 = 1.4$ and $c_2 = 0.1$), turbulence_rate = $1/n$ (n = number of decision variables). The number of hidden nodes is fixed to a value of 50 when we use the ANN.

In this study, we perform 600 real function evaluations using different surrogate methods: ANN, RBF and SVM. The results reported in Tables 5.2 and 5.3 correspond to the performance metrics adopted (SC, IGD and Spacing). We show in boldface the best mean values per test function of 10 runs per each test function for all the algorithms compared. We show the plot of all the nondominated solutions generated by a single run of the different algorithms in Figure 5.7. In all cases, we generated PF_{true}, so we could make a graphical comparison of the quality of the solutions produced by our approach. The summary of our results is the following:

- **ANN:** When we use the neural network, the algorithm shows a poor performance dealing with high-dimensional problems, but outperforms the RBFs in all the performance measures, specially in IGD. It also seems that the ANN needs more training points to make a better approximation of the function. Graphically, it can be seen that it obtained a worse approximation to the true Pareto front than the two other surrogate approaches. In ZDT2 and ZDT3, the ANN shows a better performance than the RBF, but not with respect to the SVM approach.

- **RBF:** When using the radial basis function, the algorithm shows the worst overall performance with respect to IGD and Spacing. Graphically, it can also be seen that this approach stays far away from the true Pareto front.

- **SVM:** Using the support vector machines, the algorithm obtains the best results in all the test functions with respect to the IGD performance measure. With respect to Spacing, the algorithm shows a good behavior, although in ZDT2 and ZDT4, it does not obtain the best results.

From results obtained in the first phase of our approach, we can conclude that the **SVM approach is the one that shows the best overall performance** in these particular multi-dimensional test functions. So, we decided to choose the SVM approximation fitness to hybridize it with the Rough Sets (see section 6.1 for further details).

5.5 Final remarks

We have presented an initial study of surrogate methods to solve multi-objective problems with high dimensionality. Three different methods were used in our comparative study: Artificial Neural Networks (ANNs), Radial Basis Functions (RBFs) and Support Vector Machines (SVMs), all of them in their regression form, in order to approximate the function using supervised learning. From this study, we concluded that the SVMs were the most appropriate model for dealing with the type of problems of our interest, because it provides a good approximation of the true Pareto front, although it is unable to produce a good spread of the solutions. Thus, we decided to include a local search procedure based on rough sets theory in order to intensify the search around the solutions obtained by the SVMs. The publications derived from this are: [97, 139, 140, 141].

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT1	IGD	ANN	0.0782	0.0216	0.0402	0.1073
		RBF	0.1014	0.0133	0.0694	0.1145
		SVM	**0.0125**	0.0050	0.0072	0.0210
	S	ANN	0.8445	0.0444	0.7553	0.9045
		RBF	0.8874	0.0358	0.8176	0.9288
		SVM	**0.5793**	0.0785	0.4697	0.7331

SC(A, B)		B			
		ANN	RBF	SVM	Mean
A	ANN	–	0.8581	0.0	0.4290
	RBF	0.1033	–	0.0	0.0516
	SVM	0.9561	0.9256	–	**0.9408**

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT2	IGD	ANN	0.1040	0.0451	0.0392	0.1743
		RBF	0.1568	0.0225	0.1077	0.1803
		SVM	**0.0358**	0.0007	0.0339	0.0365
	S	ANN	**0.2341**	0.3609	0.0000	0.8781
		RBF	0.9445	0.0372	0.8589	0.9847
		SVM	0.6953	0.4552	0.0000	0.9985

SC(A, B)		B			
		ANN	RBF	SVM	Mean
A	ANN	–	0.75	0.0	0.375
	RBF	0.2667	–	0.0	0.1333
	SVM	1.0	1.0	–	**1.0**

	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT3	IGD	ANN	0.1111	0.0201	0.0808	0.1491
		RBF	0.1788	0.0370	0.0732	0.2160
		SVM	**0.0262**	0.0098	0.0162	0.0497
	S	ANN	0.8479	0.0464	0.7770	0.9178
		RBF	0.8678	0.0434	0.7932	0.9162
		SVM	**0.7454**	0.0500	0.6761	0.8585

SC(A, B)		B			
		ANN	RBF	SVM	Mean
A	ANN	–	0.8584	0.0	0.4292
	RBF	0.1	–	0.0	0.05
	SVM	0.9165	0.9150	–	**0.9157**

Table 5.2: Results of inverse generational distance (IGD), spacing (S) and set coverage (SC) for ZDT1, ZDT2 and ZDT3.

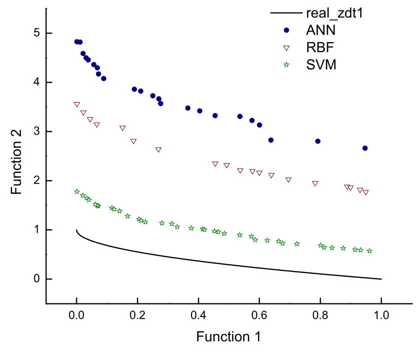
	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT4	IGD	ANN	1.3725	0.1961	0.9999	1.5805
		RBF	1.4249	0.1439	1.1032	1.6187
		SVM	**1.1788**	0.1957	0.8642	1.4998
	S	ANN	0.8746	0.1368	0.5633	1.0381
		RBF	**0.7931**	0.4000	0.0000	1.1100
		SVM	0.9900	0.0152	0.9574	1.0077

SC(A, B)		B			
		ANN	RBF	SVM	Mean
A	ANN	–	0.2824	0.0250	0.1537
	RBF	0.6717	–	0.1	0.3858
	SVM	0.83	0.7774	–	**0.8037**

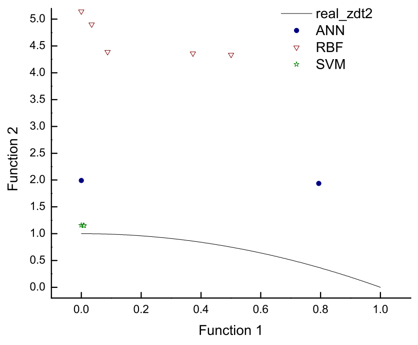
	Perf. Meas. - Algorithm		mean	St. dev.	best	worst
ZDT6	IGD	ANN	0.0914	0.0436	0.0351	0.1343
		RBF	0.1177	0.0301	0.0471	0.1390
		SVM	**0.0285**	0.0031	0.0234	0.0325
	S	ANN	0.9846	0.0673	0.9163	1.1582
		RBF	0.9602	0.0345	0.9062	1.0352
		SVM	**0.8411**	0.0473	0.7517	0.9420

SC(A, B)		B			
		ANN	RBF	SVM	Mean
A	ANN	–	0.2696	0.0	0.1348
	RBF	0.4582	–	0.0	0.2291
	SVM	0.7864	0.7280	–	**0.7572**

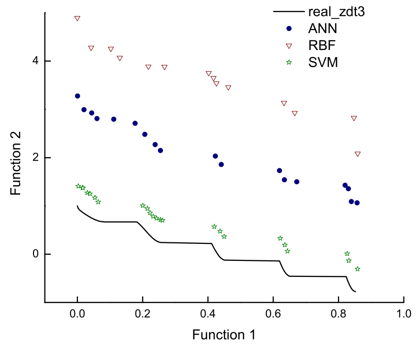
Table 5.3: Results of inverse generational distance (IGD), spacing (S) and set coverage (SC) for ZDT4 and ZDT6.



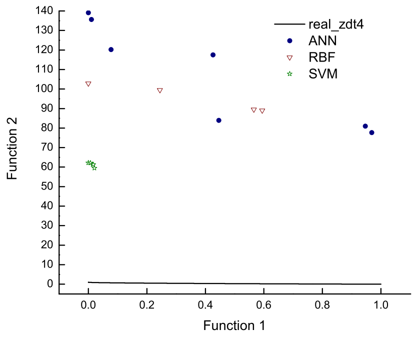
(a) ZDT1



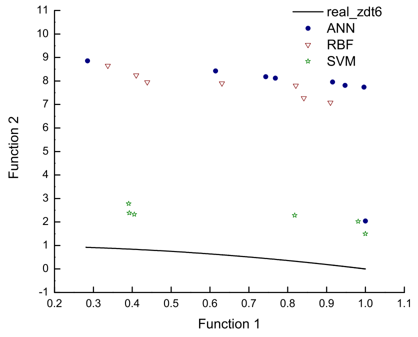
(b) ZDT2



(c) ZDT3



(d) ZDT4



(e) ZDT6

Figure 5.7: Pareto fronts generated by the surrogate methods for the ZDT test problems.

6

Exploitation: A Local Search Based on Rough Sets Theory

In this chapter, we propose the use of rough sets to improve the approximation provided by a multi-objective evolutionary algorithm (MOEA). The main idea is to hybridize the previous algorithms described in Chapter 4 with a local search procedure based on rough sets theory to approximate the Pareto front of a multi-objective optimization problem with a low computational cost. The proposed hybrid operates in two stages: in the first one, a multi-objective version of differential evolution is used as our search engine in order to generate a good approximation of the true Pareto front. Then, in the second stage, rough sets theory is adopted in order to improve the spread of the solutions found so far. To assess our proposed hybrid approach, we adopt a set of standard test functions and performance measures taken from the specialized literature. Our results are compared with respect to the NSGA-II, which is an approach representative of the state-of-the-art in the area.

6.1 Rough sets theory

Rough sets theory is a new mathematical approach to deal with imperfect knowledge. The problem of imperfect knowledge has been tackled for a long time by philosophers, logicians and mathematicians. Recently, it also became a crucial issue for computer scientists, particularly in the area of artificial intelligence (AI). There are many approaches to the problem of how to understand and manipulate imperfect knowledge. The most used one is the fuzzy set theory proposed by Lofti Zadeh [171]. Rough sets theory was proposed by Pawlak [119], and presents another attempt to this problem. Rough sets theory has been used by many researchers and practitioners all over the world and has been adopted in many interesting applications. The rough sets approach seems to be of fundamental importance to AI and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, knowledge discovery from databases, expert systems, inductive reasoning and pattern recognition. Basic ideas of rough set theory and its extensions, as well as many interesting applications, can be found in books (see [120]), special issues of journals (see [99]), proceedings of international conferences, and in the internet (see www.roughsets.org).

Let's assume that we are given a set of objects U called the *universe* and an indiscernibility relation $R \subseteq U \times U$, representing our lack of knowledge about elements of U (in our case, R is simply an equivalence relation based on a grid over the feasible set; this is just a division of the feasible set in (hyper)-rectangles). Let X be a subset of U. We want to characterize the set X with respect to R. The way rough sets theory expresses vagueness is employing a boundary region of the set X built once we know a finite number of points both inside X and outside X. If the boundary region of a set is empty it means that the set is *crisp*; otherwise, the set is *rough* (inexact). A nonempty boundary region of a set means that our knowledge about the set is not enough to define the set precisely (see Figure 6.1).

Then, each element in U is classified as *certainly* inside X if it belongs to the lower approximation or *partially* (*probably*) inside X if it belongs to the upper approximation (see Figure 6.1). The *boundary* is the difference of these two sets, and the bigger the boundary the worse the knowledge we have of set X. On the other hand, the more precise is the grid implicity used to define the indiscernibility relation R, the smaller the boundary regions are. But, the more precise is the grid, the bigger the number of elements

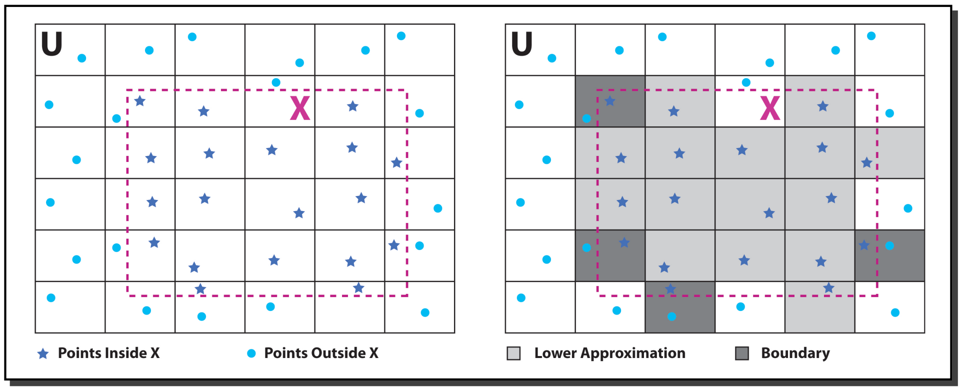


Figure 6.1: Rough approximation

in U, and then, the more complex the problem becomes. Then, the less elements in U the better to manage the grid, but the more elements in U the better precision we obtain. Consequently, the goal is obtaining "small" grids with the maximum precision possible. These two aspects are called **density** and **quality** of the grid. Note however, that at this point we will face the following problem:

- The more precise the grid is, the higher the computational cost required to manage it.
- The less precise the grid is, the less knowledge we get about the Pareto optimal set.

6.1.1 Use of rough sets in multi-objective optimization

For our MOPs we will try to approximate the Pareto front using a rough set grid. To do this, we will use an initial approximation of the Pareto front (provided by any other method) and will implement a grid in order to get more information about the front that will let us improve this initial approximation. Then, at this point we have to face the following problem: the more precise the grid is, the higher the computational cost required to manage it. Conversely, the less precise the grid is, the less knowledge we get about the Pareto front. Thus, we need to design a grid that balances these two aspects. In other words, a grid that is not so expensive (computationally speaking) but that offers a reasonably good knowledge about the Pareto front to be used to improve the initial approximation. To this aim, we must

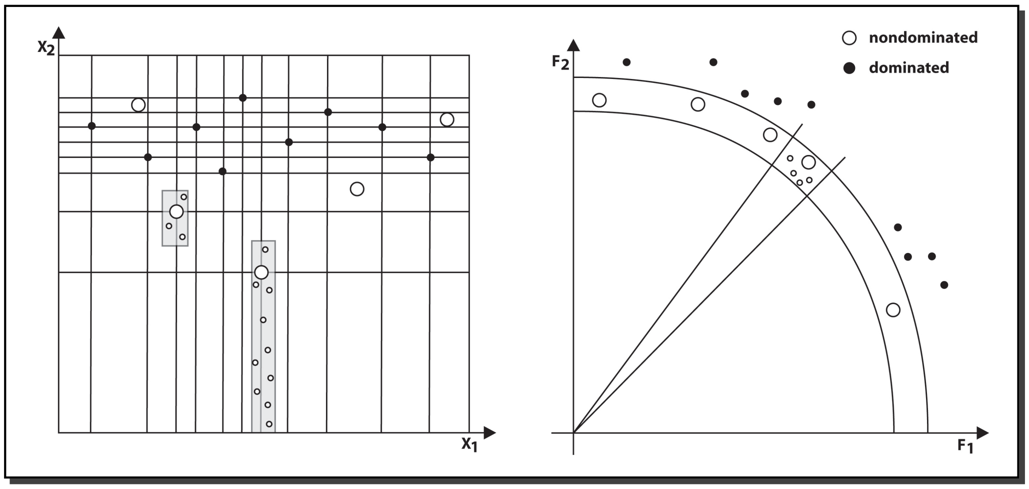


Figure 6.2: Use of a rough approximation for unconstrained MOPs

design a grid and decide which elements of U (that we will call *atoms* and will be just rectangular portions of decision variable space) are close to the Pareto optimal set and which are not. Once we have the *efficient atoms*, we could easily intensify the search over these atoms as they are built in decision variable space.

To create the grid, we have two main possibilities, when we are dealing with unconstrained MOPs and when we deal with the constrained MOPs. Next we describe the creation of the grid for each case:

Unconstrained problems To create this grid, as an input we will have several feasible points divided in two sets: the nondominated points (NS) and the dominated ones (DS). Using these two sets we want to create a grid to describe the set NS in order to intensify the search on it in the decision variable space. This is, we want to describe the Pareto set in decision variable space because then we could easily use this information to generate more efficient points and then improve this initial approximation. Figure 6.2 shows how information in objective function space can be translated into information in decision variable space through the use of a grid.

Constrained problems To create this grid, as an input we will have several points divided in three sets: the nondominated points (NS), the dominated ones (DS), and the infeasible solutions (IS) (when dealing

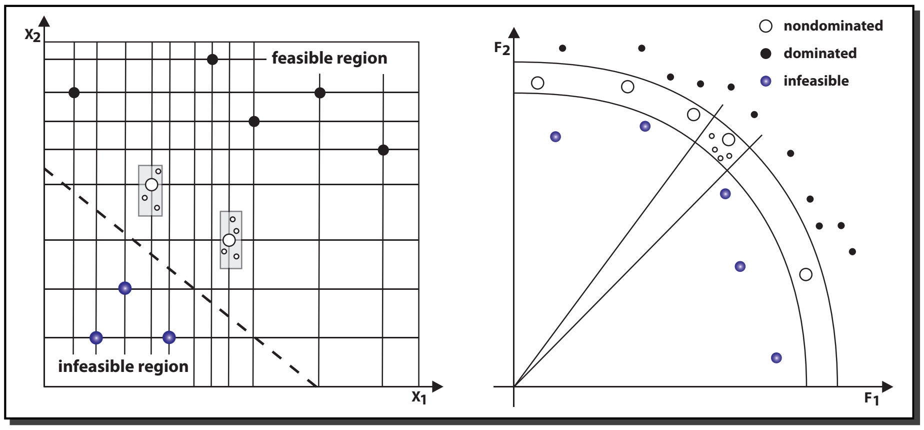


Figure 6.3: Use of a rough approximation for constrained MOPs

with constrained problems). Using these three sets we want to create a grid to describe the set NS in order to intensify the search on it. This is, we want to describe the Pareto set in decision variable space because then we could easily use this information to generate better solutions and then improve this initial approximation. Figure 6.3 shows how information in objective function space can be translated into information in decision variable space through the use of a grid, where we deal with some dominated solutions, some infeasible solutions, and some nondominated solutions. The new solutions correspond to the offspring generated inside the atoms. It can be observed that the use of the dominated solutions and infeasible solutions help the grid to build the atoms around the nondominated solutions and prevent building dominated and/or infeasible solutions.

We must note the importance of the DS and IS sets as in a rough set method the information comes from the description of the boundary of the three sets. We do not pretend to precisely characterize all the sets: 1) nondominated set NS, 2) dominated set DS and 3) Infeasible set IS. Our purpose is to delimit the nondominated solutions in such a way that the atom built associated to each nondominated solution presents the same good characteristics of the nondominated solution, causing that the offspring derived from this nondominated solution will remain nondominated and thus act as the local optimizer that we pretend to obtain. The way in which these atoms are computed is described in Section 6.1.2.

Since the computational cost of managing the grid increases with the number of points used to create it, we will try to use just a few points of NS, DS, and IS. Moreover, such points must be as far from one another as possible, because the better the distribution the points have in the initial approximation the less points we need to build a reliable grid. On the other hand, in order to diversify the search we build several grids using different sets DS, IS and NS coming from the initial approximation. To ensure these sets are really disjoint we will mark each point as explored or non-explored (if it has been used or not to compute a grid) and we will not allow repetitions. Algorithm 6 describes a Rough Sets iteration. To maintain a good distribution along the solutions in all the sets mentioned before, we use the scheme called Pareto-adaptive ϵ-dominance grid, that we describe in detail in the Chapter 7.

Algorithm 6 *Rough sets iteration*

```
1: Input nondominated solutions from the first phase $NS$.
2: Input dominated solutions from the first phase $DS$.
3: Input Infeasible solutions from the first phase $IS$.
4: Output nondominated solutions found by the RS.
5: eval ← 0
6: repeat
7:     Choose $NumEff$ unexplored points of $NS$.
8:     Choose $NumDom$ unexplored points of $DS$.
9:     Choose $NumInfea$ unexplored points of $IS$.
10:    for $i = 0$ to $NumEff$ do                ▷ Generate $NumEff$ efficient atoms.
11:        for $j = 0$ to $Offspring$ do
12:            Generate (randomly) a new point in $atom_i$
13:            eval ← eval + 1
14:            if new is infeasible then
15:                Send new to $IS$
16:            else
17:                if new is efficient then
18:                    Include it in $NS$
19:                end if
20:                if A point old in $NS$ is dominated by new then
21:                    Send old to $DS$
22:                end if
23:            end if
24:        end for
25:    end for
26: until $MaxEval$ < eval
```

6.1.2 Atom construction in rough sets theory

As we have mentioned, the rough sets departs the search from the nondominated set NS generated by some other algorithm. This set is contained within the secondary population. We also need the dominated set (DS) and the set of infeasible solutions (IS).

From the set NS we choose $NumEff$ points previously unselected. If we do not have enough unselected points, we choose the rest randomly from the set NS. Next, we choose from the set DS, $NumDom$ points previously unselected (and in the same way if we do not have enough unselected points, we complete them in a random fashion) and $NumInfea$ unselected points from the set IS. These points will be used to approximate the boundary between the Pareto front and the rest of the feasible set in decision variable space. What we want to do now is to intensify the search in the area where the nondominated points reside, and avoid finding more points on the area where the dominated or infeasible points reside. For this purpose, we store these points on the set $Items$, where the set $Items$ will contain all the nondominated, dominated and infeasible points from NS, DS and IS respectively; and then perform a rough sets iteration:
A rough sets iteration is described next:

1. **Range initialization:** For each decision variable i, we compute and sort (from the smallest to the highest) the different values it takes in the set $Items$. Then, for each decision variable i, we have a set of $Range_i$ values (where $Range_i < |Items|$), and combining all these sets we have a (non-uniform) grid in decision variable space.

2. **Compute atoms:** We compute $NumEff$ rectangular atoms centered in the $NumEff$ nondominated solutions selected. To build a rectangular atom associated to a nondominated point $x^e \in Items$ we compute the following upper and lower bounds for each decision variable i:

   - Lower bound i: Middle point between x_i^e and the previous value in the set $Range_i$.
   - Upper bound i: Middle point between x_i^e and the following value in the set $Range_i$.

   In both cases, if there are no previous or subsequent values in $Range_i$, we consider the absolute lower or upper bound of variable i. This setting allows the method to explore close to the feasible set boundaries.

3. **Generate offspring:** Inside each atom we randomly generate Off-*spring* new points. Each of these points is sent to the set NS (that, as indicated before, can be a $pa\epsilon$-dominance grid) to check if it must be included as a new nondominated point or to IS in case of being infeasible. If any point in NS is dominated by this new point, it is sent to the set DS.

It should be clear that the purpose of using rough sets in our approach is to rebuild the Pareto front, departing from the solutions generated by the search engine adopted for its first phase (see Chapter 4). Thus, we use a local search engine based on rough sets theory.

6.1.3 Results of rough sets

In order to show that the inclusion of the local search algorithm based on rough sets theory helps our approach to obtain a better approximation of the true Pareto front with the same number of evaluations performed by each approach, we compared: 1) The ϵ-MyDE algorithm (see Section 4.1.3 in page 61) and 2) The DEMORS algorithm (that combines ϵ-MyDE and rough sets (RS)).

Our approach was validated using the **ZDT** set [173] which contains unconstrained and high dimensional problems. We also used 7 constrained test problems: Binh2 [12], Kita [85], Osyczka1 and Osyczka2 [116], Srinivas [149], Tanaka [155] and the Welded beam problem [124]. All of them, except for Kita problem, are minimization problems. The description of all the problems are contained in Appendix A. In all cases, the parameters of our approach were set as indicated in Table 6.1.

Three performance measures were adopted in order to allow a quantitative assessment of our results: (1) Two set coverage (**SC**), (2) Inverted generational distance (**IGD**), and (3) Spread (Δ). For each test problem, 30 independent runs were performed. A detailed description of the test functions are contained in Section 3.3.

Table 6.2 shows a summary of the results with the ZDT problems, whereas the rest are shown in Table 6.3. The results reported are the mean values for each of the three performance measures and the standard deviation of the 30 runs performed. The best mean values in each case are shown in **boldface**.

First, we will comment about the unconstrained problems. It can be clearly seen in Table 6.2 that DEMORS produced the best mean values in all cases for the convergence metrics (SC and IGD) for all the test functions adopted, and obtained better results in ZDT2, ZDT3 and ZDT4 for the metric Δ which measures the distribution of solutions along the Pareto front. The graphical results are shown in Figures 6.4 to 6.8, and they serve to reinforce our argument of the superiority of the results obtained by DEMORS against ϵ-MyDE with the same number of function evaluations. These plots correspond to the run in the mean value with respect to the inverted generational distance metric. In all the optimization problems, the true Pareto front is shown with a continuous line and the approximation obtained by each algorithm is shown with black circles. We can see from those plots, that in all the cases, ϵ-MyDE is not able to converge to the true Pareto front with $5,000$ evaluations. In contrast, DEMORS was able to converge to the true Pareto front in all cases.

With respect to the constrained MOPs, again DEMORS produced the best mean values in all the convergence metrics for all the seven test functions, and obtained the best results for the Δ metric in almost all the functions except for Osyczka. The graphical results are shown in Figures 6.9 to 6.15. These plots correspond to the run in the mean value with respect to the inverted generational distance performance measure. From the plots, it is also clear that DEMORS outperforms ϵ-MyDE in all cases, and is able to find all the true Pareto front with only $10,000$ evaluations, except for Osyczka's problem, in which DEMORS has some difficulties to find the extremes of the Pareto front and Welded Beam in which a portion of the Pareto front is lost.

6.2 Final remarks

The experiments performed in this chapter led us to conclude that rough sets is a suitable tool to be hybridized with a MOEA in order to improve the local exploration around the nondominated solutions found so far. If the search engine adopted to produce a coarse-grained approximation of the Pareto front is efficient (as in our case), then a good approximation of the true Pareto front can be achieved with a low computational cost.

What we try to do with the rough sets is to intensify the search over the nondominated solutions found by another algorithm. To this aim, we design a grid and decide which elements are nondominated (in which we build *atoms*, that are rectangular portions in the decision variable space) and which are not. Once we have the *efficient atoms* associated to the nondominated solutions, we intensify the search over these atoms, generating good offspring in the vicinity of the good solutions. The publication derived from this chapter is [67].

Parameter	**ϵ-MyDE**		**DEMORS**	
	unconstrained	constrained	unconstrained	constrained
P	25		25	
$P_{external}$	100		100	
P_m	$\frac{1}{nVariables}$		$\frac{1}{nVariables}$	
G_{max}	200	400	200	400
P_c	0.9	0.3	0.9	0.3
$elitism$	0.1	0.5	0.1	0.5
$Offspring$	–	–	1	1
$NumEff$	–	–	2	2
$NumDom$	–	–	10	5
$NumInfea$	–	–	–	5
Evaluations	5,000	10,000	5,000	10,000

Table 6.1: Parameters of both approaches: ϵ-MyDE and DEMORS.

Problem	Metric - Algorithm		mean	σ	best	worst
ZDT1	SC	DEMORS	**0.9313**	0.0739	0.6739	1.0000
		ϵ-MyDE	0.0174	0.0247	0.0000	0.0976
	IGD	DEMORS	**0.0010**	0.0008	0.0005	0.0044
		ϵ-MyDE	0.0030	0.0007	0.0019	0.0045
	Δ	DEMORS	0.5858	0.1751	0.2561	0.8335
		ϵ-MyDE	**0.5528**	0.0591	0.4472	0.7067

Problem	Metric - Algorithm		mean	σ	best	worst
ZDT2	SC	DEMORS	**0.8917**	0.2574	0.0000	1.0000
		ϵ-MyDE	0.0841	0.2363	0.0000	0.9630
	IGD	DEMORS	**0.0011**	0.0011	0.0003	0.0051
		ϵ-MyDE	0.0080	0.0062	0.0012	0.0201
	Δ	DEMORS	**0.4450**	0.1453	0.2695	0.8252
		ϵ-MyDE	0.7446	0.2140	0.4596	1.1893

Problem	Metric - Algorithm		mean	σ	best	worst
ZDT3	SC	DEMORS	**0.4522**	0.2825	0.0625	0.8889
		ϵ-MyDE	0.1309	0.1309	0.0000	0.3611
	IGD	DEMORS	**0.0062**	0.0026	0.0039	0.0148
		ϵ-MyDE	0.0075	0.0013	0.0054	0.0096
	Δ	DEMORS	**0.6718**	0.0741	0.5595	0.8543
		ϵ-MyDE	0.6963	0.0645	0.5933	0.8212

Problem	Metric - Algorithm		mean	σ	best	worst
ZDT4	SC	DEMORS	**0.7683**	0.3921	0.0000	1.0000
		ϵ-MyDE	0.1743	0.3615	0.0000	1.0000
	IGD	DEMORS	**0.1166**	0.1349	0.0006	0.4592
		ϵ-MyDE	0.3500	0.2453	0.0389	0.8202
	Δ	DEMORS	**0.8496**	0.1647	0.3222	1.0296
		ϵ-MyDE	0.8902	0.0891	0.7043	1.0004

Problem	Metric - Algorithm		mean	σ	best	worst
ZDT6	SC	DEMORS	**0.4318**	0.4777	0.0000	1.0000
		ϵ-MyDE	0.4163	0.4469	0.0000	1.0000
	IGD	DEMORS	**0.0183**	0.0170	0.0002	0.0535
		ϵ-MyDE	0.0198	0.0160	0.0001	0.0465
	Δ	DEMORS	0.8376	0.2433	0.2753	1.1790
		ϵ-MyDE	**0.7879**	0.2782	0.3018	1.0974

Table 6.2: Performance measure results with respect to the two set coverage (SC), inverted generational distance (IGD) and spread (Δ) for the ZDT problems.

	Metric - Algorithm		mean	σ	best	worst
Binh2	SC	DEMORS	**0.4044**	0.0680	0.2021	0.5647
		ϵ-MyDE	0.1320	0.0351	0.0426	0.1935
	IGD	DEMORS	**1.1942**	0.1050	1.0019	1.3766
		ϵ-MyDE	1.1988	0.1087	1.0020	1.3773
	Δ	DEMORS	**0.6600**	0.0206	0.6209	0.7071
		ϵ-MyDE	0.6808	0.0216	0.6037	0.6917

	Metric - Algorithm		mean	σ	best	worst
Kita	SC	DEMORS	**0.5853**	0.1136	0.2603	0.7882
		ϵ-MyDE	0.0327	0.0263	0.0000	0.1017
	IGD	DEMORS	**0.0038**	0.0008	0.0026	0.0052
		ϵ-MyDE	0.0103	0.0044	0.0057	0.0227
	Δ	DEMORS	**0.5337**	0.1644	0.3464	0.9646
		ϵ-MyDE	0.9640	0.1491	0.6442	1.2442

	Metric - Algorithm		mean	σ	best	worst
Osyczka	SC	DEMORS	**0.8710**	0.2241	0.0000	1.0000
		ϵ-MyDE	0.0019	0.0058	0.0000	0.0222
	IGD	DEMORS	**0.0152**	0.0139	0.0016	0.0552
		ϵ-MyDE	0.0556	0.0270	0.0155	0.1233
	Δ	DEMORS	0.8121	0.3112	0.3874	1.8086
		ϵ-MyDE	**0.3969**	0.2888	0.0311	1.1559

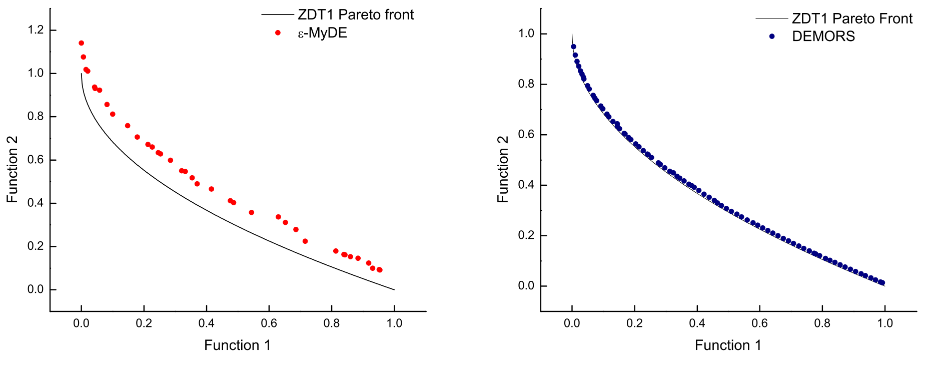
	Metric - Algorithm		mean	σ	best	worst
Osyczka2	SC	DEMORS	**0.8592**	0.1202	0.5660	1.0000
		ϵ-MyDE	0.0077	0.0137	0.0000	0.0526
	IGD	DEMORS	**0.1802**	0.2053	0.0526	1.2204
		ϵ-MyDE	0.3041	0.2701	0.0834	1.2878
	Δ	DEMORS	**0.6562**	0.1560	0.3586	1.0411
		ϵ-MyDE	1.0745	0.1362	0.7979	1.3300

	Metric - Algorithm		mean	σ	best	worst
Srinivas	SC	DEMORS	**0.5072**	0.0946	0.2447	0.6364
		ϵ-MyDE	0.0829	0.0293	0.0319	0.1474
	IGD	DEMORS	**0.0173**	0.0011	0.0151	0.0200
		ϵ-MyDE	0.0213	0.0025	0.0172	0.0315
	Δ	DEMORS	**0.2216**	0.0292	0.1739	0.2746
		ϵ-MyDE	0.2652	0.0377	0.2121	0.3974

	Metric - Algorithm		mean	σ	best	worst
Tanaka	SC	DEMORS	**0.6062**	0.2433	0.2295	1.0000
		ϵ-MyDE	0.0617	0.0581	0.0000	0.2889
	IGD	DEMORS	**0.0015**	0.0007	0.0007	0.0039
		ϵ-MyDE	0.0106	0.0138	0.0020	0.0589
	Δ	DEMORS	**0.5638**	0.1763	0.3546	1.0642
		ϵ-MyDE	0.9886	0.2833	0.3718	1.4192

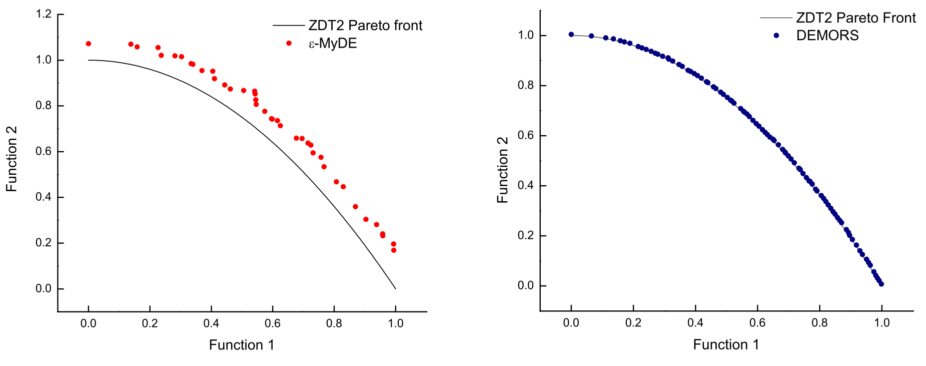
	Metric - Algorithm		mean	σ	best	worst
Welbeam	SC	DEMORS	**0.7982**	0.2048	0.2532	1.0000
		ϵ-MyDE	0.0063	0.0180	0.0000	0.0879
	IGD	DEMORS	**0.0242**	0.0193	0.0084	0.0765
		ϵ-MyDE	0.0705	0.0577	0.0189	0.3204
	Δ	DEMORS	**0.7782**	0.1933	0.4545	1.3390
		ϵ-MyDE	0.9692	0.1670	0.6784	1.3419

Table 6.3: Performance measure results with respect to two set coverage (SC), inverted generational distance (IGD) and spread (Δ) for the constrained problems.



(a) ϵ-MyDE (b) DEMORS

Figure 6.4: Pareto fronts obtained with ϵ-MyDE and DEMORS for ZDT1.



(a) ϵ-MyDE (b) DEMORS

Figure 6.5: Pareto fronts obtained with ϵ-MyDE and DEMORS for ZDT2.

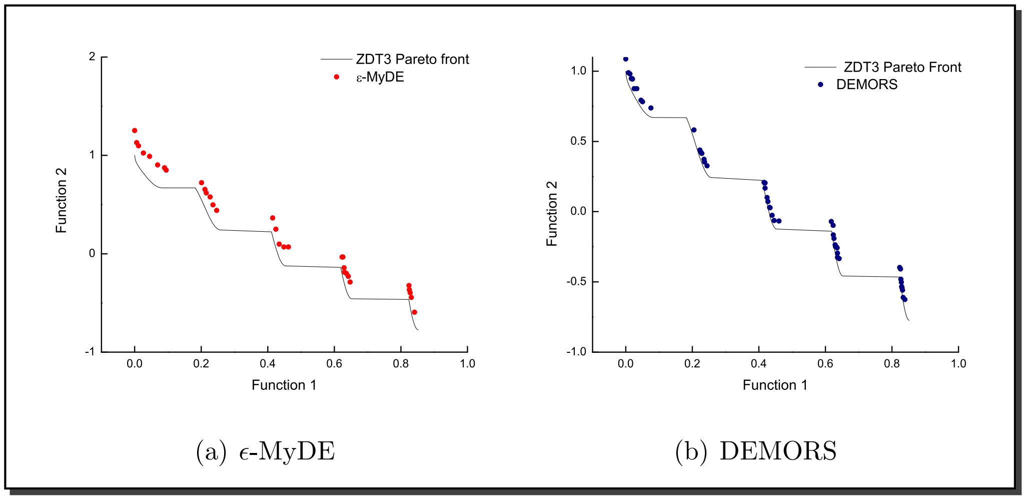


Figure 6.6: Pareto fronts obtained with ϵ-MyDE and DEMORS for ZDT3.

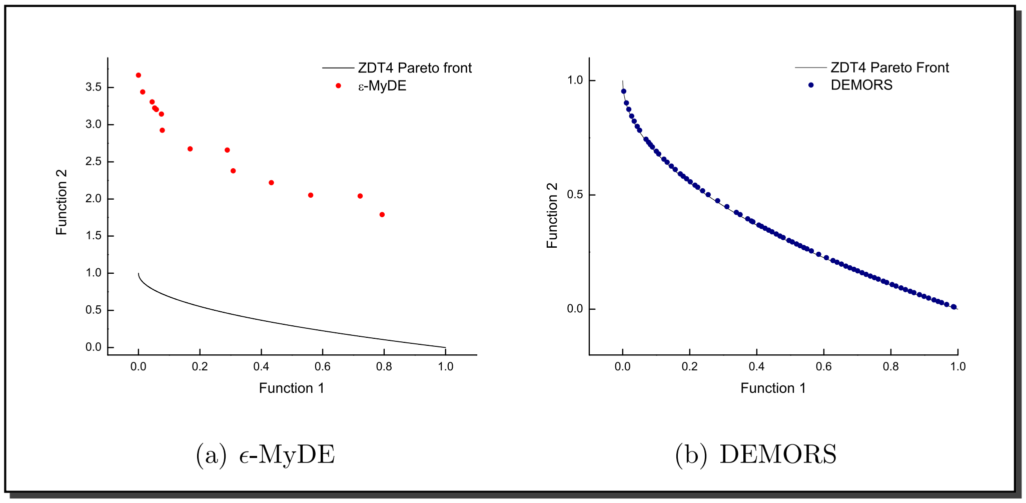


Figure 6.7: Pareto fronts obtained with ϵ-MyDE and DEMORS for ZDT4.

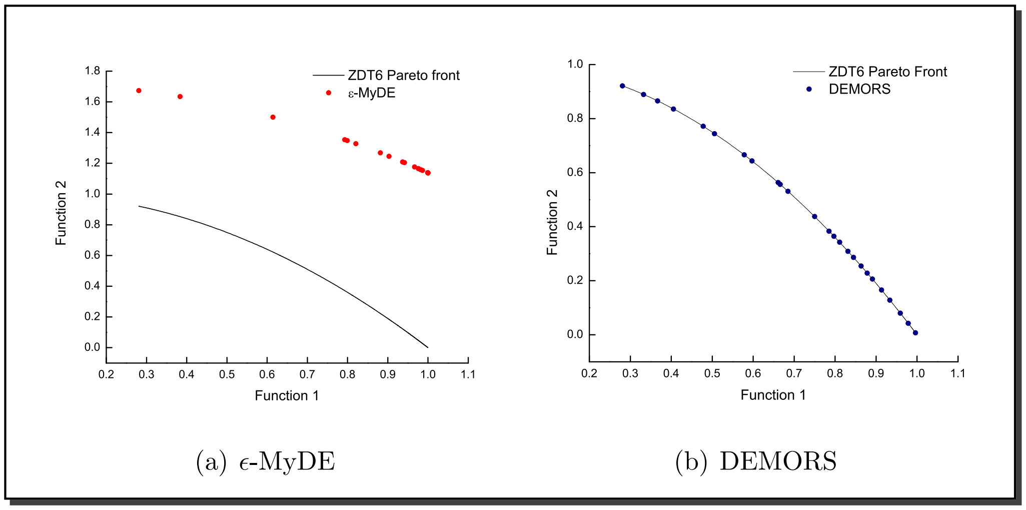


Figure 6.8: Pareto fronts obtained with ϵ-MyDE and DEMORS for ZDT6.

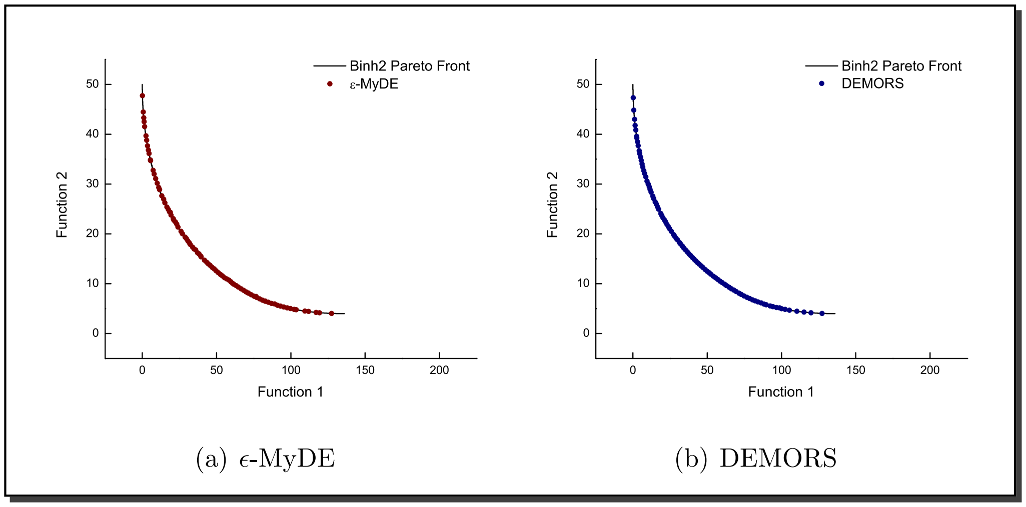
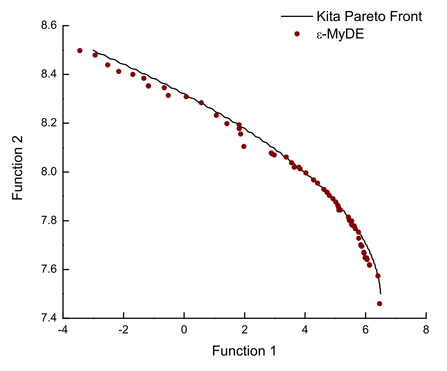
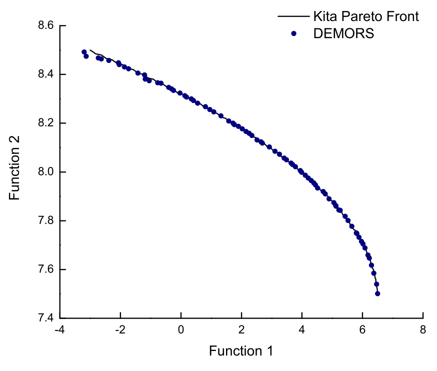


Figure 6.9: Pareto fronts obtained with ϵ-MyDE and DEMORS for Binh2.

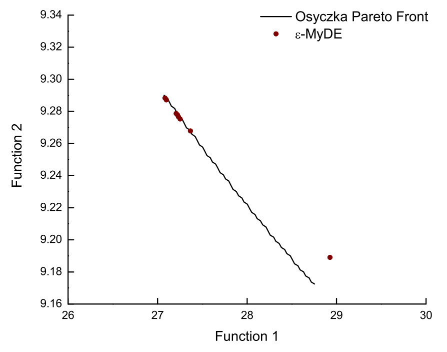


(a) ϵ-MyDE

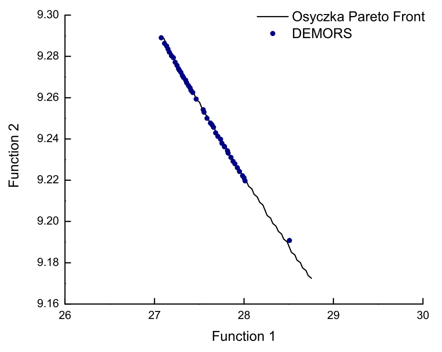


(b) DEMORS

Figure 6.10: Pareto fronts obtained with ϵ-MyDE and DEMORS for Kita.

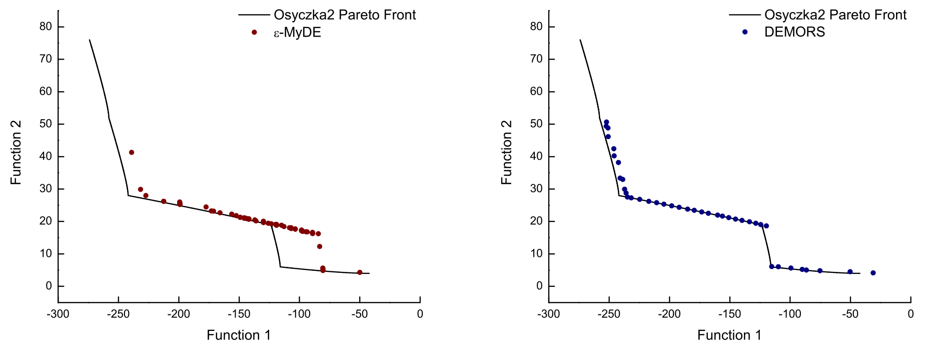


(a) ϵ-MyDE



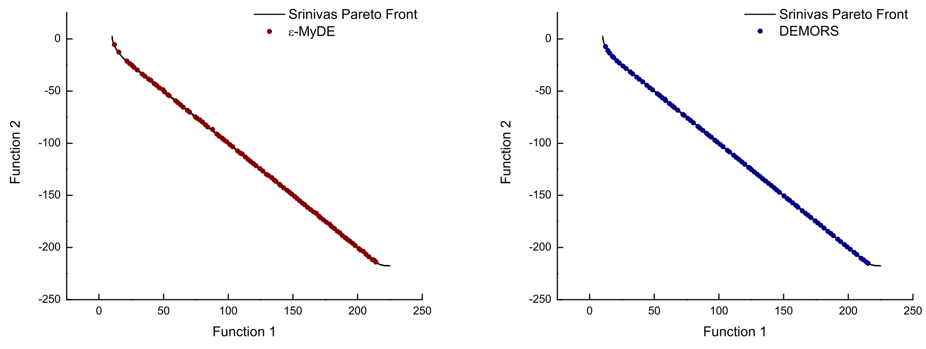
(b) DEMORS

Figure 6.11: Pareto fronts obtained with ϵ-MyDE and DEMORS for Osyczka.



(a) ϵ-MyDE (b) DEMORS

Figure 6.12: Pareto fronts obtained with ϵ-MyDE and DEMORS for Osyczka2.



(a) ϵ-MyDE (b) DEMORS

Figure 6.13: Pareto fronts obtained with ϵ-MyDE and DEMORS for Srinivas.

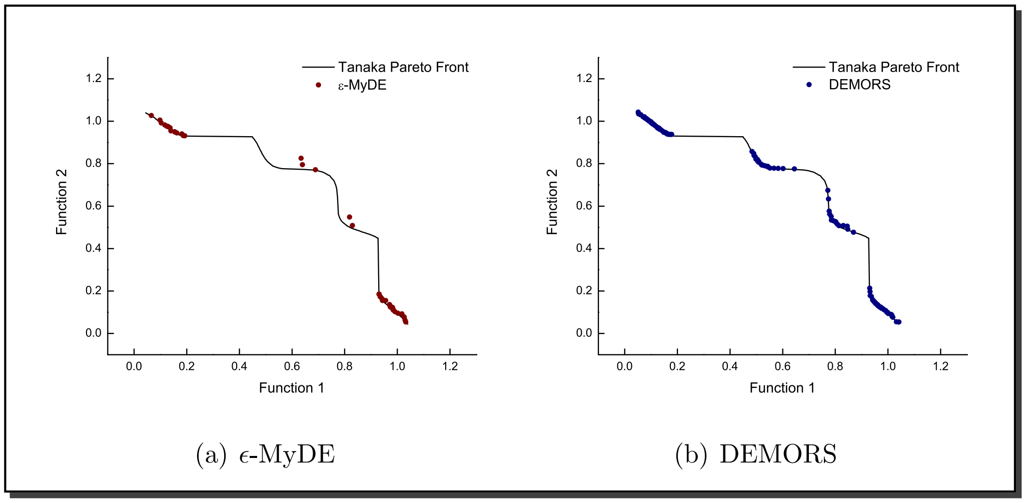


Figure 6.14: Pareto fronts obtained with ϵ-MyDE and DEMORS for Tanaka.

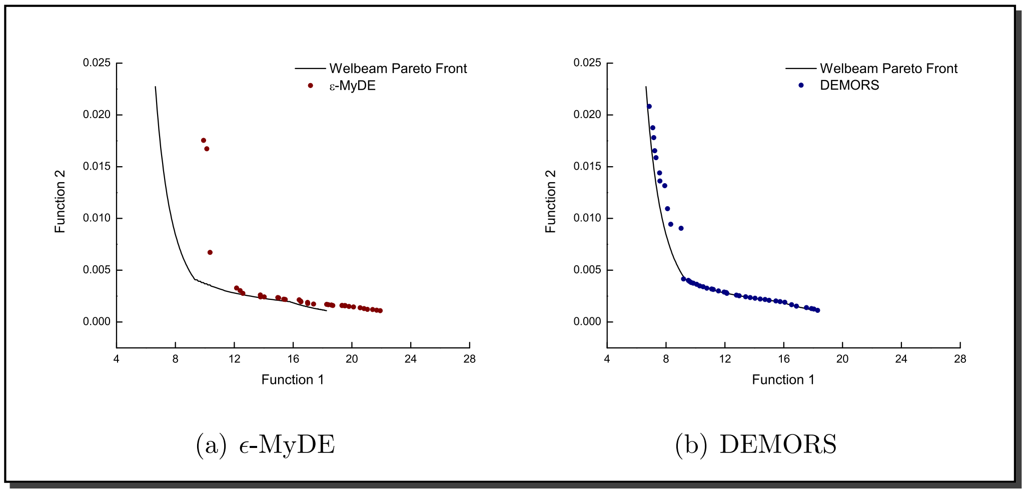


Figure 6.15: Pareto fronts obtained with ϵ-MyDE and DEMORS for Welded Beam.

7

Distribution: PAϵ-dominance

Researchers have proposed several mechanisms to reduce the number of nondominated solutions generated by a MOEA (most of them applicable to external archives): clusters [175], adaptive grids [88], crowding [39] and relaxed forms of Pareto dominance [98]. However, next we only mention those that need an external archive to store nondominated solutions, namely:

- **Adaptive grid**: Proposed by Knowles & Corne [88], the adaptive grid is really a space formed by hypercubes. Such hypercubes have as many components as objective functions has the problem to be solved. Each hypercube can be interpreted as a geographical region that contains an n number of individuals. The adaptive grid allows us to store nondominated solutions and to redistribute them when its maximum capacity is reached. A graphical representation of the adaptive grid is shown in Figure 7.1

- **ϵ-dominance**: This is a relaxed form of dominance proposed by Laumanns et al. [98]. The so-called ε-Pareto set is an archiving strategy that maintains a subset of generated solutions. It guarantees convergence and diversity according to well-defined criteria, namely the

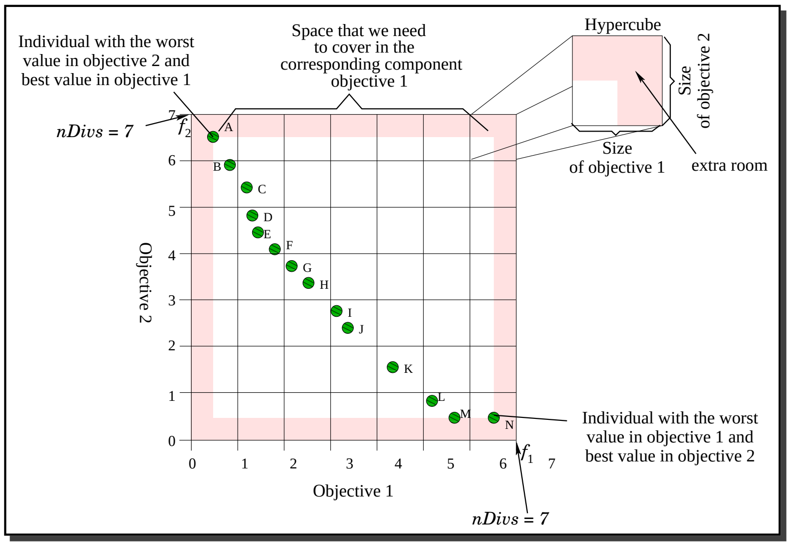

Individual with the worst value in objective 2 and best value in objective 1
Space that we need to cover in the corresponding component objective 1
Hypercube
Size of objective 2
Size of objective 1
extra room
nDivs = 7
f_2
Objective 2
Objective 1
A
B
C
D
E
F
G
H
I
J
K
L
M
N
Individual with the worst value in objective 1 and best value in objective 2
f_1
nDivs = 7
0
1
2
3
4
5
6
7

Figure 7.1: Graphic representation of the adaptive grid mechanism.

value of the ε parameter, which defines the resolution of the grid to be adopted for the secondary population. The general idea of this mechanism is to divide objective function space into boxes of size ε. Each box can be interpreted as a geographical region that contains a single solution. The approach accepts a new solution into the ε-Pareto set if any of the 3 cases holds: 1) it is the only solution in the box which it belongs to, 2) it dominates to other(s) solution(s) or 3) it competes against other nondominated solution inside the box, but it is closer to the origin vertex of the box. This algorithm is very attractive both from a theoretical and from a practical point of view. However, in order to achieve the best performance, it is necessary to provide the size of the box (the ε parameter) which is problem-dependent, and it's normally not known before executing a MOEA.

7.1 Pareto-adaptive ϵ-dominance

Laumanns et al. [98] proposed two different methods/schemes to implement ϵ-dominance: the additive and the multiplicative approaches. We assume all objectives are to be minimized. Then, given a vector $f \in \mathbb{R}^m$ and $\epsilon > 0$, for

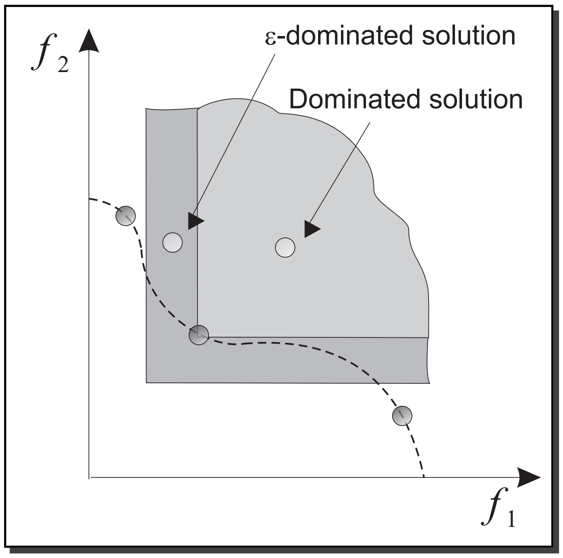


Figure 7.2: ϵ-dominance relation example.

the additive scheme f is said to ϵ-dominate all points in the set

$$\{g \in \mathbb{R}^m : f_i - \epsilon \leq g_i, \text{ for all } i = 1, ..., m\}$$

whereas for the multiplicative scheme f is said to ϵ-dominate all points in the set

$$\{g \in \mathbb{R}^m : f_i(1-\epsilon) \leq g_i, \text{ for all } i = 1, ..., m\}.$$

Although the above definitions assume the same ϵ value for all the objectives, they can be easily generalized to consider a different value for each objective. In order to do this, we only have to take an ϵ_i for each $i \in \{1, 2, ..., m\}$. Without loss of generality, we assume that $1 \leq f_i \leq K$, for all i.

Both schemes generate a hyper-grid in objective function space with $\left(\frac{K-1}{\epsilon}\right)^m$ boxes in the additive scheme and $\left(\frac{-\log K}{\log(1-\epsilon)}\right)^m$ for the multiplicative one. As ϵ-dominance only allows one point in each box, these grids could accommodate a maximum of $\left(\frac{K-1}{\epsilon}\right)^{m-1}$ non ϵ-dominated points for the additive scheme and $\left(\frac{-\log K}{\log(1-\epsilon)}\right)^{m-1}$ non ϵ-dominated points for the multiplicative scheme. Another possibility would be to ask the decision maker for the number of desired solutions and adjust the ϵ values in order to achieve that number. For example, if the decision maker wants T points in the Pareto front, for the additive scheme we can easily compute the value $\epsilon = (K-1)/T^{\frac{1}{m-1}}$,

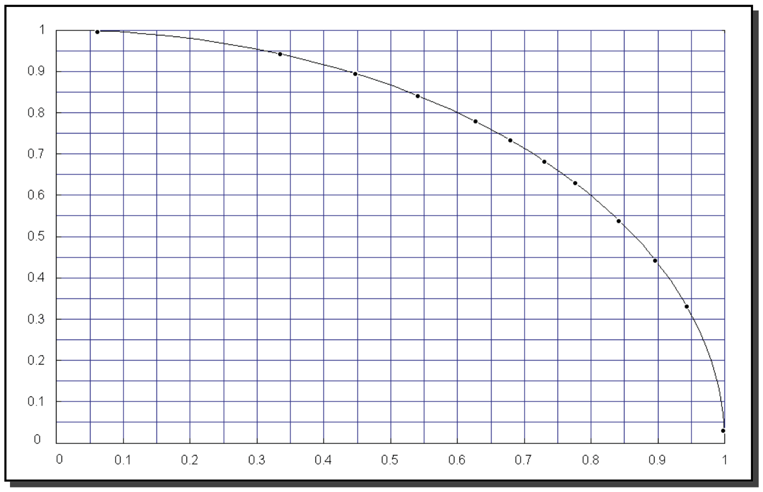


Figure 7.3: Uniform grid with 400 boxes (maximum capacity of 20 points) for the curve $x^2+y^2=1$. This grid allows a maximum of 12 points (the other 8 points are lost) because either the extreme points are easily ϵ-dominated or the precision of the grid is insufficient.

that will generate a hyper-grid with a maximum capacity of T points non ϵ-dominated. Similarly, this leads to an $\epsilon = 1 - K^{-T^{\frac{1}{1-m}}}$ for the multiplicative scheme.

7.1.1 ϵ-dominance

ϵ-dominance has been found to be an efficient mechanism to maintain diversity in multi-objective optimization problems without losing convergence properties towards the Pareto-optimal set [37, 38, 130]. Moreover, its implementation is quite easy and the decision maker can control the number of obtained solutions in a very intuitive way. As it is shown in Laumanns et al. [98], ϵ-dominance creates a hyper-grid in objective function space where each box uniquely contains one point. In order to do this, a two-level selection mechanism is implemented. The first checks a box-level dominance relation, so that the algorithm always maintains a set of nondominated boxes. In the second level, if two points share the same box, the traditional Pareto dominance relation is applied, so that the dominating point is retained. If none of these two points dominates the other, then the criterion normally adopted

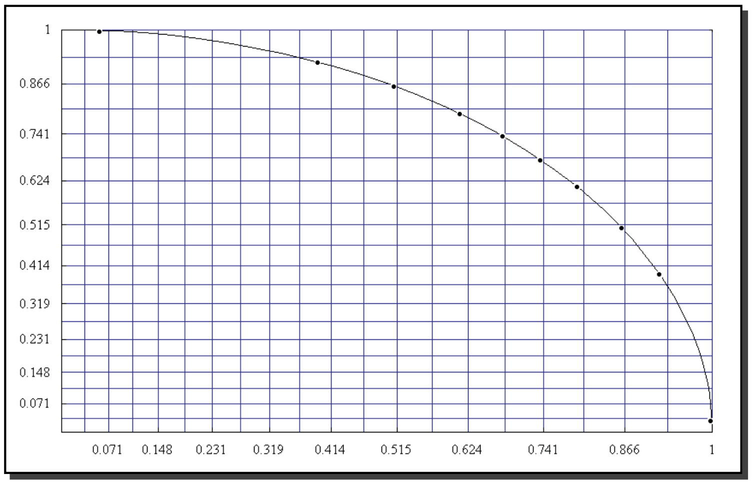


Figure 7.4: Non-uniform grid with 400 boxes (maximum capacity of 20 points) for the curve $x^2 + y^2 = 1$. In this case, because the front is concave, the grid only allows a maximum of 10 points, losing again both extreme points of the Pareto front.

is to keep the point closest to the lower lefthand corner of the box (or to the origin, for more than two objectives). Note that despite the use of this selection mechanism inside each box, this does not affect the convergence rate, because we are always maintaining the best solutions found so far (following the ideas of [134]). These mechanisms are also used in our paϵ-dominance. Thus, both our paϵ-dominance and ϵ-dominance guarantee convergence.

However, ϵ-dominance has some limitations such as the following:

1. We can lose a high number of efficient solutions if the decision maker does not take into account (or does not know beforehand) the geometrical characteristics of the true Pareto front of the problem to be solved.

2. It is normally the case that we lose the extreme points of the Pareto front, as well as points located in segments of the Pareto front that are almost horizontal or vertical, as shown in Figure 7.3.

3. The upper bound for the number of points allowed by a grid is not easy to achieve. For a non-adaptive grid, the upper bound is only achieved when the true Pareto front is linear.

4. When adopting a multiplicative scheme, the size of the region ϵ-dominated by the point $f \in \mathbb{R}^m$ depends on the f_i values. Then, the size of this region is larger in the cases where the f_i values increase. For the same reason, if the f_i values are close to zero, ϵ-dominance would be similar to the traditional Pareto-dominance. This kind of grid is not suitable, for instance, for concave Pareto fronts (see Figure 7.1).

In order to address some of the problems previously described, we propose an alternative scheme for the additive ϵ-dominance. Our proposal is called Pareto-adaptive-ϵ-dominance (paϵ-dominance). This scheme maintains the good properties of ϵ-dominance while overcoming its main limitations.

In our proposal, we consider not only a different ϵ value for each objective but also the vector $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_m)$ associated to each $f = (f_{1,}f_2, ..., f_m) \in \mathbb{R}^m$ depending on the geometrical characteristics of the Pareto-optimal front. In other words, we consider different intensities of dominance for each objective according to the position of each point along the Pareto front. Then, the size of the boxes will be adapted depending on the area in the objective functions space so that boxes will be smaller where needed (normally at the extremes of the Pareto front), and larger in other less problematic parts of the front.

To this aim, each Pareto front (that we will assume normalized: $0 \leq f_i \leq 1$ for any i) will be associated to one curve of the following family

$$\{x^p + y^p = 1 : 0 \leq x, y \leq 1, 0 < p < \infty\}.$$

for bi-objective optimization problems,

$$\{x^p + y^p + z^p = 1 : 0 \leq x, y, z \leq 1, 0 < p < \infty\}$$

for three dimensional problems, or

$$\{x_1^p + x_2^p + \cdots + x_n^p = 1 : 0 \leq x_1, x_2, ..., x_n \leq 1, 0 < p < \infty\}.$$

for N-dimensional problems. These families have the following property: for $p > 1$, the curve (or surface) is concave and the bigger the p value the longer the almost horizontal (and almost vertical) parts of the front; and, for $p < 1$, the curve (surface) is convex and the lower the p value the longer the almost horizontal (and almost vertical) stretches in the front. Finally, for $p = 1$ we get the linear front $x + y = 1$. For this last value, it will be

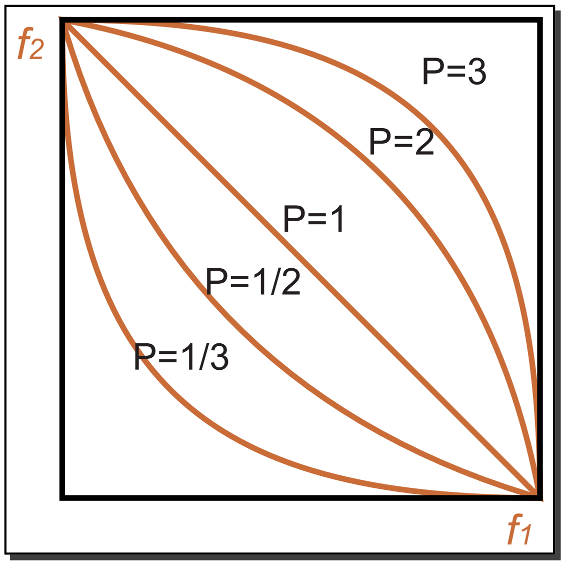


Figure 7.5: Curves in the reference set for $p = \frac{1}{3}, \frac{1}{2}, 1, 2, 3$. The ϵ values we have to consider for $p = 2$ and $p = 3$ have to be different because $x^3 + y^3 = 1$ has longer horizontal and vertical stretches than $x^2 + y^2 = 1$. The same happens for $p = \frac{1}{3}$ and $p = \frac{1}{2}$.

shown that our scheme coincides with the additive ϵ-dominance. Thus, our proposal generalizes the ϵ-dominance concept introduced by Laumanns et al. [98] (just taking $p = 1$ in Figure 7.3).

In Figure 7.1.1, we show five different curves of this family for $p \in \left\{\frac{1}{3}, \frac{1}{2}, 1, 2, 3\right\}$.

In order to decide the value of p, we need an initial Pareto front approximation, denoted by F, that will determine which value of p fits better to our front. This is, we will use F to be the model where the p-curve should fit. Then, the number of efficient points included in F can be critical for the final performance, because if the value of p is not appropriate, then the grid will not be appropriate neither. Obviously, the higher the number of efficient points in F the better the performance of the grid generated. On the other hand, if we want to maintain the diversity properties of ϵ-dominance, we should generate the first grid as soon as possible. For example, for a grid with a maximum capacity of 100 vectors, the different experiments performed by the authors indicated that the best results are obtained when the number of points in F is between 75 and 125 (we set it at 100 for our experiments).

So, we store all the nondominated points found in the archive (or secondary population) during the search process checking the Pareto dominance relation among them until reaching the number of desired nondominated points to generate the first grid.

To compute the value of p, we calculate the area (hypervolume) under the poligonal line (surface) formed by points in F (see Section 7.1.4 for further details). Once we know this area, we estimate the value of $p \in (0, +\infty)$ by means of an interpolation process. We choose p when the area under $x^p + y^p = 1$ is as similar to the F hypervolume as desired (this precision is set beforehand).

Although we are assuming that the Pareto front is symmetrical, this method could be generalized using the sets

$$\{x^p + y^q = 1 : 0 \leq x, y \leq 1, 0 < p, q < \infty\},$$

$$\{x^p + y^q + z^r = 1 : 0 \leq x, y, z \leq 1, 0 < p, q, r < \infty\}$$

or

$$\{x_1^{p_1} + x_2^{p_2} + \cdots + x_n^{p_n} = 1 : 0 \leq x_1, x_2, ..., x_n \leq 1, 0 < p_1, p_2, ..., p_n < \infty\}.$$

Nevertheless, the association procedure is more unstable, as it depends on F to a higher degree and the error for the estimated p and q values could be large.

Obviously, ϵ-dominance and paϵ-dominance work better for continuous fronts. In the case of disconnected fronts, both schemes have to be handled more carefully. The approximation of the p value could be less realistic depending on the number of parts that conform the Pareto front and the distance between them. Nevertheless, paϵ-dominance has been tested with Kursawe's problem with success, and this problem has a Pareto front consisting of four (disconnected) segments.

7.1.2 ϵ computation

Once the p value is estimated and the number T of points desired by the decision maker is known, we compute the sizes of the boxes for each objective $i \in \{1, 2, ..., m\}$, that is, the vector $\epsilon^i = (\epsilon^i_1, \epsilon^i_2, ..., \epsilon^i_T)$.

We use geometric sequences to do this[1]: we compute these values according to a geometric sequence depending on p, T and the size of the first box for each dimension, ϵ_1^i, so that, for $n \geq 2$,

$$\epsilon_n^i = \frac{\epsilon_{n-1}^i}{p^{v_i}} = \frac{\epsilon_{n-2}^i}{(p^{v_i})^2} = \cdots = \frac{\epsilon_1^i}{(p^{v_i})^{n-1}} \tag{7.1}$$

where v_i controls the *speed of variation* of the ϵ values in order to get a uniform distribution in the Pareto front.

Then, for each objective $i \in \{1, 2, ..., m\}$ we have to estimate the size of the first box, ϵ_1^i, and the speed v_i. To this end, we propose the following system of nonlinear equations, for each i,

$$\left.\begin{array}{l} \sum\limits_{n=1}^{T} \epsilon_n^i = 1 \\ \\ \sum\limits_{n=1}^{T/2} \epsilon_n^i = \frac{1}{2^{(\frac{1}{p})}} \end{array}\right\} . \tag{7.2}$$

The first equation represents the fact that the sum of the sizes of all boxes must be equal to the range of f_i. The second equation tries to spread the obtained nondominated points along the front and forces the accommodation of $T/2$ nondominated points in one half of the objective i, and the remaining $T/2$ points in the other half. Taking into account that $x^p + y^p = 1$ is symmetric, it is easy to obtain the middle point: $\left(\frac{1}{2^{1/p}}, \frac{1}{2^{1/p}}\right)$.

As both series in (7.2) are geometric, it follows that

$$\left.\begin{array}{l} \sum\limits_{n=1}^{T} \epsilon_n^i = \sum_{n=1}^{T} \frac{\epsilon_1^i}{(p^{v_i})^{n-1}} = \epsilon_1^i \frac{1-(\frac{1}{p^{v_i}})^T}{1-\frac{1}{p^{v_i}}} = \epsilon_1^i \frac{p^{Tv_i}-1}{(p^{v_i}-1)p^{(T-1)v_i}} = 1 \\ \\ \sum\limits_{n=1}^{T/2} \epsilon_n^i = \sum_{n=1}^{T/2} \frac{\epsilon_1^i}{(p^{v_i})^{n-1}} = \epsilon_1^i \frac{1-(\frac{1}{p^{v_i}})^{T/2}}{1-\frac{1}{p^{v_i}}} = \epsilon_1^i \frac{p^{\frac{T}{2}v_i}-1}{(p^{v_i}-1)p^{(\frac{T}{2}-1)v_i}} = \frac{1}{2^{1/p}} \end{array}\right\} . \tag{7.3}$$

[1]Geometric sequences allow us to easily control the size of the boxes either for increasing or for decreasing sizes (that is, for convex or concave problems). Also, they do not have to be explicitly added, since the expression to compute its summation is known beforehand.

Then, the solutions of (7.3) are

$$\left.\begin{array}{c} \epsilon_1^i = \frac{(p^{v_i}-1)p^{(T-1)v_i}}{p^{Tv_i}-1} \\ \left(1-2^{\frac{1}{p}}\right)p^{Tv_i} + 2^{\frac{1}{p}}p^{\frac{T}{2}v_i} - 1 = 0 \end{array}\right\}. \tag{7.4}$$

But ϵ_1^i is already calculated in the first equation and it does not appear in the second one. So, we only have to solve the second equation in (7.4). Due to its nonlinearity, we propose to solve it using a numerical method, for example, a dichotomy method [125]. Although this is not the fastest numerical method available, we decided to use it because of the simplicity of its implementation and the easy that results to control the precision required. Along our experiments, we applied a dichotomy method for v_i in the interval $[0.001, 0.1]$ because we set $T = 100$ and $\frac{1}{12} \leq p \leq 12$.

7.1.3 Box index vector

As in the original ϵ-dominance, the dominance relation is generalized among boxes. That is, at most one element is kept in each box and this representative vector can only be replaced by a dominating one. To this end, we associate with each vector $f \in \mathbb{R}^m$ a box index vector $b(f) = (b_1, ..., b_m) \in \mathbb{Z}^m$. So, in a first level, the algorithm always maintains a set of nondominated boxes (this is, a set of nondominated box index vectors). And in a second level, if two vectors share the same box, the representative vector is eliminated if the other one dominates it.

In order to calculate the box index vector of $f = (f_1, f_2, ..., f_m)$, we take b_i to be the only integer so that

$$\sum_{n=1}^{b_i} \epsilon_n^i \leq f_i < \sum_{n=1}^{b_i+1} \epsilon_n^i.$$

for all $i \in \{1, 2, ..., m\}$. Again, because both series are geometric, the above inequalities are equivalent to

$$\epsilon_1^i \frac{p^{v_i} - \left(\frac{1}{p^{v_i}}\right)^{b_i-1}}{p^{v_i}-1} \leq f_i < \epsilon_1^i \frac{p^{v_i} - \left(\frac{1}{p^{v_i}}\right)^{b_i}}{p^{v_i}-1}.$$

If we assume that $p^{v_i} - 1 > 0$, it is equivalent to

$$p^{v_i} - \left(\frac{1}{p^{v_i}}\right)^{b_i-1} \leq \frac{f_i\left(p^{v_i}-1\right)}{\epsilon_1^i} < p^{v_i} - \left(\frac{1}{p^{v_i}}\right)^{b_i}.$$

Then, by successive (elementary) operations, we have the following equivalent expressions

$$\ln\left(\frac{1}{p^{v_i}}\right)^{b_i-1} \geq \ln\left(p^{v_i} - \frac{f_i\left(p^{v_i}-1\right)}{\epsilon_1^i}\right) > \ln\left(\frac{1}{p^{v_i}}\right)^{b_i}.$$

$$(b_i-1)\ln\left(\frac{1}{p^{v_i}}\right) \geq \ln\left(p^{v_i} - \frac{f_i\left(p^{v_i}-1\right)}{\epsilon_1^i}\right) > b_i \ln\left(\frac{1}{p^{v_i}}\right)$$

$$b_i - 1 \leq \frac{\ln\left(p^{v_i} - \frac{f_i(p^{v_i}-1)}{\epsilon_1^i}\right)}{\ln\left(\frac{1}{p^{v_i}}\right)} < b_i.$$

Finally, we choose

$$b_i(f) = \left\lfloor \frac{\log\left(\frac{\epsilon_1^i p^{v_i} - (p^{v_i}-1)f_i}{\epsilon_1^i}\right)}{\log\left(\frac{1}{p^{v_i}}\right)} + 1 \right\rfloor.$$

It is easy to check that the same b_i is obtained if $p^{v_i} - 1 < 0$.

In that way, although the whole objective function space is discretized into boxes, the nondominated vectors are allocated into boxes whose box index vectors range from $(0, 0, ..., 0)$ to $(T-1, T-1, ..., T-1)$. Nevertheless, if vectors outside the above limits are found, we must include them in the grid (if their box index vectors are non-paϵ-dominated) in one of the two following ways:

1. Update the grid re-computing new box limits. In this case, a new p value would be also calculated. This does not ensure the convergence property of ϵ-dominance [98] and the behavior could be worse.

2. Do not change any of the box limits (the assignment of the elements to the boxes must remain the same). This guarantees the same convergence properties of ϵ-dominance but the number of nondominated points could be larger than T. In this case, a larger ϵ value can be chosen, but the grid would have to be updated again.

In our proposed approach we follow the second choice shown above, and, once the grid is generated, its boundaries are never modified. Note however, that the grid depends on the quality of the first set of nondominated points, F, as such set determines the value of p. The best performance is attained if there is no need to re-adjust this initial grid. We only update the grid when some of the coordinates of the new box index vector are sufficiently far to require it, this is, when $b_i < -3$ or $b_i > T + 3$ for some i. The best results have been obtained when the first grid is generated once F contains at least 100 nondominated points, as the hyper-grid is almost never re-adjusted with this setting. This minimum value has been empirically derived after numerous experiments for the curves $x^p + y^p = 1$. In all the cases that the authors empirically tested, with values close to 100, the value of p that was obtained was very close to the simulated curve.

Finally, if two vectors f and g share the same box (so, $b(f) = b(g)$) and neither dominates the other, we choose the one closer (using Euclidean distance, for example) to the lower lefthand corner of the box,[2] denoted by $c(b) = (c_1, ..., c_m)$. In order to calculate c_i, we sum the size of all the previous boxes, that is

$$c_i = \sum_{n=1}^{b_i} \epsilon_n^i = \epsilon_1^i \frac{p^{v_i b_i}}{(p^{v_i} - 1)p^{v_i(b_i-1)}}$$

for all $i = 1, 2, ..., m$.

In Figure 7.1.3 we can see the grid obtained for $x^2 + y^2 = 1$. The figure clearly indicates how the grid adapts the size of the boxes as needed.

7.1.4 Algorithm for the hypervolume

As we mentioned above, each Pareto front is associated to one curve in

$$\{x^p + y^p = 1 : 0 \leq x, y \leq 1, 0 < p < \infty\}$$

by estimating the area (hypervolume) under the polygonal line (surface) formed by the points in F in objective function space.

Let us assume a bi-objective optimization problem and $F = \{f^j = (f_1^j, f_2^j) : j = 1, 2, ..., |F|\}$ is the set of nondominated vectors obtained before

[2]For more than two objectives, a reference point could always be selected in each hypercube, bearing in mind the characteristics of the problem. As long as the mechanism used to select this reference point does not change, the convergence properties still hold.

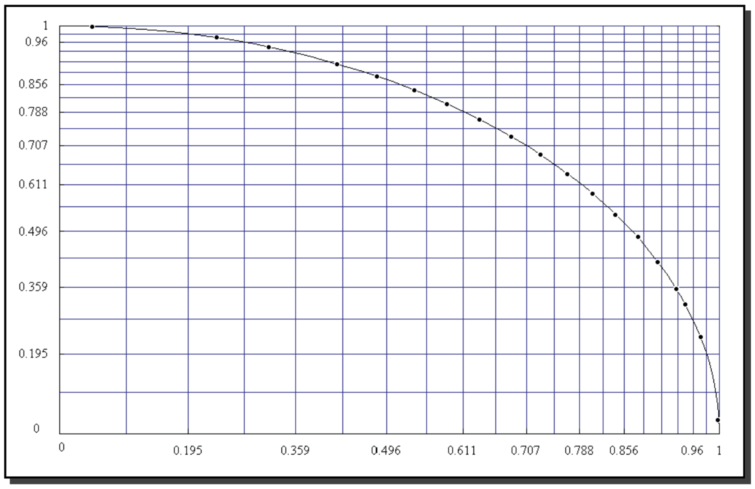


Figure 7.6: Alternative grid with 400 boxes (maximum capacity of 20 points) using paϵ-dominance for the curve $x^2 + y^2 = 1$. In this case the grid allows a maximum of 19 points.

generating the hyper-grid. Obviously, if we rank points in F in ascending order of magnitude in the first objective, f_2 values are ranked in descending order. Then, the area under the polygonal, $A(F)$, is calculated by the mean value of the following lower, $LA(F)$, and upper, $UA(F)$, approximation areas:

$$LA(F) = \sum_{i=1}^{|F|-1} \left(f_1^{i+1} - f_1^i\right) f_2^{i+1},$$

and

$$UA(F) = \sum_{i=1}^{|F|-1} \left(f_1^{i+1} - f_1^i\right) f_2^i.$$

From these areas, $A(F)$ is

$$A(F) = \frac{LA(F) + UA(F)}{2}.$$

For the three-dimensional case, the difficulty increases because points cannot be fully ranked. So, we propose the following procedure:

Initially, the *nPoints* points are sorted by their values in the third objective value. These values are then used to make *slices*. Each slice has a hypervolume in the first 2 objectives (Area). This area is calculated and it is multiplied by its depth in the third objective; then, the values obtained are summed up to obtain the total hypervolume of the $nPoints$ points.

Each slice in the hypervolume contains a different number of points, because we iteratively remove the lowest value point in the third objective. However, not all the points in each slice contribute to the Area in that slice. Some points may be dominated in the first two objective values and contribute nothing. So, it is important to re-check dominance (as a maximization problem) in the first two objective values (not including the third objective) by each slice to calculate the hypervolume (Area).

7.1.5 Acceptance scheme on paϵ-dominance grid

The identification array divides the whole objective space into hyper-boxes, each having ϵ_j size in the j-th objective. With the identification arrays calculated for the offspring c_1 and each archive member a, we use the procedure illustrated in Figure 7.7 and described next:

1. If the identification array B_a of any archive member a dominates that of the offspring c_i, then it means that the offspring is ϵ-dominated by this archive member and so the offspring *is not accepted*. Case (a) in Figure 7.7.

2. If B_{c_i} of the offspring dominates the B_a of any archive member a, the archive member is deleted and the offspring *is accepted*. Case (b) in Figure 7.7.

   If neither of the above two cases occur, then it means that the offspring is ϵ-non-dominated with respect to the archive contents. There are two further possibilities in this case:

   (a) If the offspring shares the same $\mathbf{B}$ vector with an archive member (meaning that they belong to the same hyper-box), then they are first checked for the usual non-domination. If the offspring dominates the archive member or the offspring is nondominated with respect to the archive member but is closer to the $\mathbf{B}$ vector (in terms of the Euclidian distance) than the archive member, then the offspring *is retained*. This is case (c) in Figure 7.7.

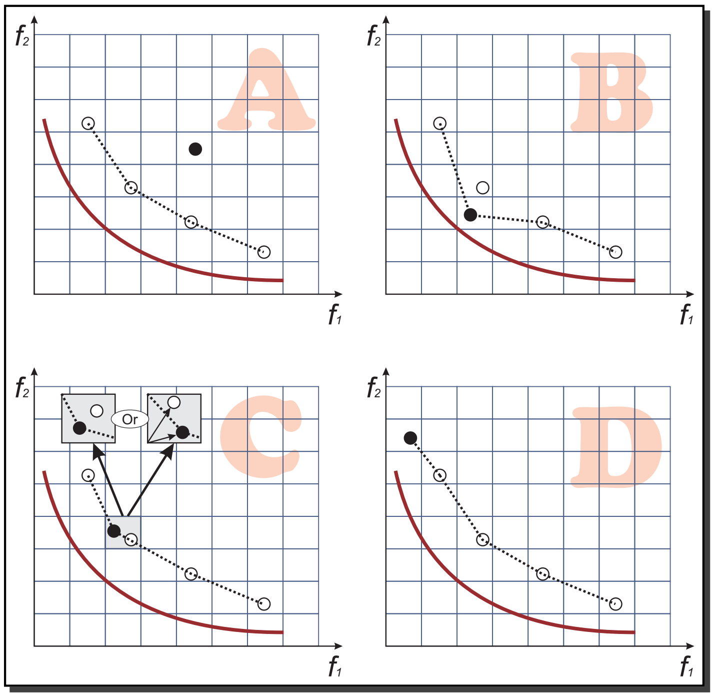


Figure 7.7: Four cases of accepting an offspring into the external archive

(b) In the event of an offspring not sharing the same **B** vector with any archive member, the offspring *is accepted*. This is case (d) in Figure 7.7.

Using the above procedure, we can guarantee the generation of a well-distributed set of nondominated solutions. Also, the value of ϵ adopted (defined by the user) regulates the size of the external archive. Thus, there is no need to pre-fix an upper limit on the size of the archive as done in most traditional MOEAs.

7.2 paϵ-Dominance results

In order to validate our proposed paϵ-dominance, we adopted three algorithms: Two of them use ϵ-dominance, and in one of them, such a mechanism is replaced by our paϵ-dominance to make the third algorithm. This will allow us to show also the performance of the same algorithm with and without paϵ-dominance. The three multi-objective evolutionary algorithms adopted for our experimental study are the following:

1. **ϵ-MyDE**: This approach was proposed earlier in this thesis (see Section 4.1.3 for further details), which uses the ϵ-dominance concept to retain the nondominated solutions obtained during the evolutionary process.

2. **ϵ-MOEA**: This approach was proposed by Deb et al. [37, 38], and it consists of a steady-state genetic algorithm which maintains an archive of nondominated individuals. Note however, that this algorithm does not use the Pareto dominance relation when updating the archive. Instead, it uses the ϵ-dominance relation. One parent is selected from the main population and other from the archive. Then, an offspring is produced and it is allowed to enter into the archive if ϵ-dominates at least one element of the archive, and if no archive member ϵ-dominates it.

3. **paϵ-MyDE**: This is a modification of the ϵ-MyDE approach indicated above, in which we include paϵ-dominance instead of the regular ϵ-dominance concept.

Table 7.1 summarizes the parameter settings adopted for all the algorithms compared. In Table 7.1, P refers to the population at each generation, G_{max} is the total number of generations (or iterations) to be performed. Note that all the algorithms perform the same number of objective function evaluations: 7,500 for all test problems. NP is the number of solutions expected by each algorithm; this parameter is controlled by the value of $\vec{\epsilon}$ ($\vec{\epsilon}$ values for ϵ-MyDE and ϵ-MOEA have been selected to this end following the guidelines provided by Laumanns et al. [98]). F is a parameter applicable only to differential evolution. P_c and P_m are the crossover and mutation rates, respectively.

7.2.1 Test functions and metrics

We chose five continuous (unconstrained) test problems with different geometrical characteristics for our experimental study. Note that our choice of problems was directed by the geometrical characteristics of the Pareto fronts rather than by the difficulty of solving each test problem, since our

Parameter	**ϵ-MyDE**	**ϵ-MOEA**	**paϵ-MyDE**
P	100	100	100
NP	100 (approx)	100(approx)	100 (approx)
G_{max}	75	75	75
P_c	0.95	1.0	0.95
P_m	1/n	1/n	1/n
F	0.5	–	0.5

Table 7.1: Parameters used by the algorithms compared.

Function	**k**	**n**	**Type**	**Characteristics**
Deb11	2	2	$min(f_1, f_2)$	Convex, bimodal
Deb52	2	2	$min(f_1, f_2)$	Concave
Kursawe	2	3	$min.(f_1, f_2)$	Disconnected
ZDT1	2	30	$min.(f_1, f_2)$	Multimodal
DTLZ2	3	12	$min(f_1, f_2, f_3)$	Concave, three-dimensional

Table 7.2: **k** denotes the number of objectives, **n** the number of decision variables, **Type** specifies the type of optimization problem (maximization or minimization) and **Characteristics** provides a summary of the geometrical characteristics of the Pareto front.

goal is to show the advantages of our paϵ-dominance scheme over the original ϵ-dominance.

The problems selected are the following: Deb11 (convex and bimodal) and Deb52 (the Pareto front is concave) from [34]; Kursawe's problem [94] (the Pareto front is disconnected); ZDT1 (multimodal problem) from [173]; and DTLZ2 from [41] (a three-objective problem). Tables 7.2 and A.2 show further details of these problems.

7.2.2 Results

In this section, we compare the performance of our proposed paϵ-dominance using the aforementioned algorithms and five different test functions. Tables 7.3, 7.4, 7.5, 7.6 and 7.7 show, for each performance measure considered,

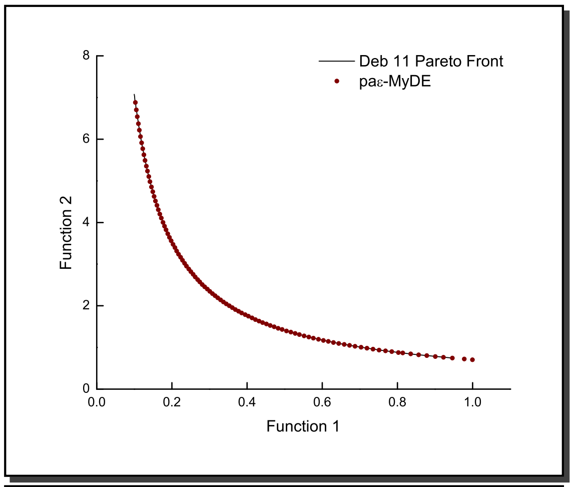
Algorithm		No. of points	Chi-Square	Spread	Crowding
paϵ-MyDE	Mean	**99.833**	**7.714**	**0.230**	**0.009**
	SDev	1.003	0.209	0.016	0.001
	Max	101	8.255	0.273	0.011
	Min	98	7.280	0.191	0.009
ϵ-MyDE	Mean	46.433	8.125	0.490	0.042
	SDev	1.086	0.216	0.016	0.001
	Max	50	8.668	0.526	0.047
	Min	45	7.832	0.464	0.038
ϵ-MOEA	Mean	49.900	8.581	0.559	0.038
	SDev	6.730	1.680	0.120	0.003
	Max	60	11.010	0.734	0.041
	Min	45	7.317	0.461	0.033

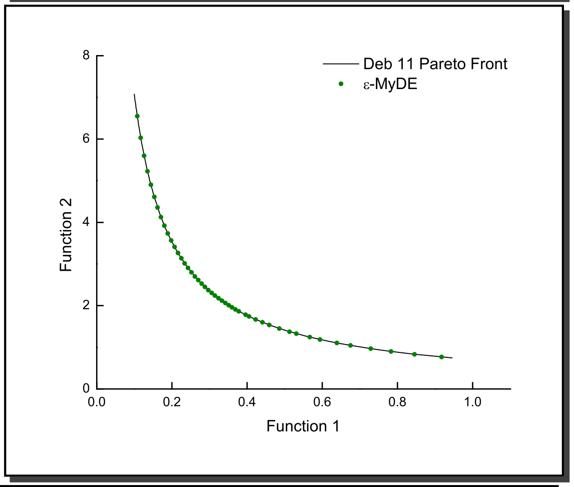
Table 7.3: Mean, standard deviation, maximum and minimum values over 30 runs for the first test problem (Deb11).

its mean value, standard deviation and the maximum and minimum value over 30 independent runs. We emphasize the best values using **boldface**.

Table 7.3 shows the results for the first problem considered (Deb11). It is worth mentioning that paϵ-MyDE achieved the best results in this case, not only regarding the distribution of solutions, but also with respect to the number of solutions retained (its average was 99.8 from a maximum of 100). In fact, the paϵ-MyDE obtained the best results with respect to all the performance measures considered. In Figure 7.8, we show the Pareto fronts obtained by the three algorithms. This figure graphically shows that our approach (paϵ-MyDE) has benefited from adopting paϵ-dominance instead of ϵ-dominance.

Table 7.4 shows the results for the second test problem (Deb52). In this case, due to an almost horizontal region in the Pareto front, ϵ-dominance loses a big number of points. Although the number of points generated by paϵ-MyDE is also far from 100, it finds more than twice the number of points obtained by the two other algorithms adopting ϵ-dominance. Again, the paϵ-MyDE obtained the best results with respect to all the performance measures considered. Regarding the Chi-square metric, paϵ-MyDE is significantly better than ϵ-MyDE and ϵ-MOEA, but there are no differences between these





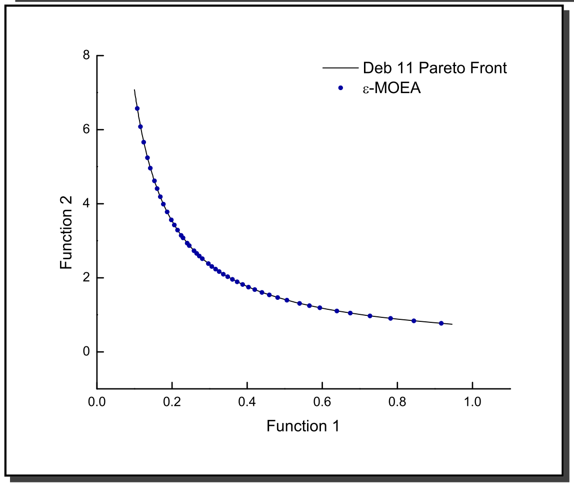


Figure 7.8: Nondominated solutions generated by paϵ-MyDE (top), ϵ-MyDE (middle) and ϵ-MOEA (bottom) for the first test problem (Deb11).

Algorithm		No. of points	Chi-Square	Spread	Crowding
paϵ-MyDE	Mean	**75.033**	**6.875**	**0.377**	**0.056**
	SDev	1.494	0.1942	0.046	0.006
	Max	78	7.461	0.474	0.068
	Min	72	6.510	0.328	0.048
ϵ-MyDE	Mean	34.800	8.385	0.502	0.073
	SDev	0.833	0.100	0.031	0.007
	Max	37	8.646	0.585	0.083
	Min	33	8.165	0.452	0.058
ϵ-MOEA	Mean	33.600	8.494	0.534	0.105
	SDev	0.663	0.188	0.049	0.001
	Max	35	8.799	0.580	0.106
	Min	33	8.105	0.390	0.101

Table 7.4: Mean, standard deviation, maximum and minimum values over 30 runs for the second test problem (Deb52).

two. In Figure 7.9, we show the Pareto fronts obtained by the three algorithms. Notice that paϵ-MyDE and ϵ-MOEA were both able to find the extreme points despite the difficult geometrical characteristics of this Pareto front. Although paϵ-dominance presents the best distribution, there is a gap in the horizontal part of the front due to the strong asymmetry of the Pareto front.

Table 7.5 shows the results for the third test problem (Kursawe). In this case, the performance measures are very similar for the three approaches compared, although our paϵ-MyDE outperformed the others with respect to two of them. The reason for this similar performance is that the p value associated to this problem is close to 1 and, as previously mentioned, ϵ-dominance and paϵ-dominance are almost the same as the p value gets close to 1. The number of points found is around 60 because this front is disconnected. In Figure 7.10 we show the Pareto fronts obtained by the three methods. In this case, paϵ-dominance and ϵ-dominance generate similar grids.

Table 7.6 shows the results for the fourth test problem (ZDT1). Also, our paϵ-MyDE obtained the best results with respect to all the performance measures considered. Again, the number of points retained is close to 100 and their distribution is quite good. In Figure 7.11 we show the Pareto fronts

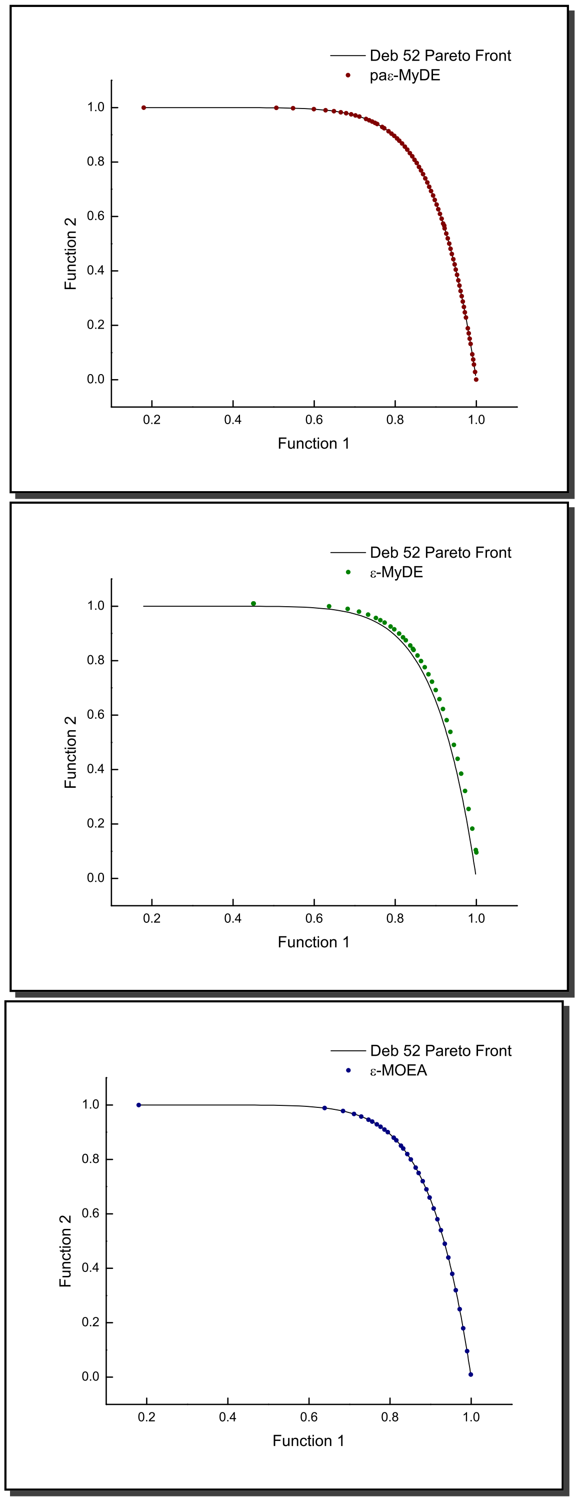


Figure 7.9: Efficient solutions generated by paϵ-MyDE (top), ϵ-MyDE (middle) and ϵ-MOEA (bottom) for the second test problem (Deb52).

Algorithm		No. of points	Chi-Square	Spread	Crowding
paϵ-MyDE	Mean	**63.733**	**6.113**	0.351	0.036
	SDev	1.711	0.275	0.025	0.001
	Max	68	6.739	0.404	0.040
	Min	60	5.645	0.303	0.034
ϵ-MyDE	Mean	60.933	6.682	**0.300**	**0.035**
	SDev	1.031	0.207	0.022	0.001
	Max	63	7.303	0.339	0.039
	Min	59	6.401	0.241	0.033
ϵ-MOEA	Mean	57.433	6.653	0.327	**0.035**
	SDev	0.761	0.224	0.016	0.001
	Max	59	7.273	0.358	0.037
	Min	56	6.298	0.287	0.033

Table 7.5: Mean, standard deviation, maximum and minimum values over 30 runs for the third test problem (Kursawe).

obtained by the three methods. Notice that paϵ-MyDE was able to find the extreme points despite the almost vertical region of this Pareto front.

Finally, Table 7.7 shows the results for the fifth test problem (DTLZ2). Again, best mean values are obtained by paϵ-dominance except for the Chi-square metric. In Figure 7.12 we show the Pareto fronts obtained. Note that the graphical results may be misleading in this case, since the distribution obtained by ϵ-MOEA may appear to have the best spread. However, that seems to be due to the lower number of points that it obtains. Nevertheless, the new grid finds more points specially on the extreme areas of the Pareto front.

7.3 Final remarks

In our proposed scheme, we considered different ϵ-dominance regions depending on the geometrical characteristics of the Pareto-optimal front. In order to do this, each Pareto front is associated to one curve of the family

$$\{x^p + y^p = 1 : 0 \leq x, y \leq 1, 0 < p < \infty\}.$$

Algorithm		No. of points	Chi-Square	Spread	Crowding
paϵ-MyDE	Mean	**93.366**	**4.684**	**0.158**	**0.009**
	SDev	2.057	0.650	0.013	0.001
	Max	98	5.788	0.200	0.011
	Min	90	3.401	0.141	0.006
ϵ-MyDE	Mean	77.233	5.611	0.211	0.014
	SDev	1.605	0.317	0.015	0.002
	Max	81	6.392	0.240	0.023
	Min	74	5.067	0.181	0.010
ϵ-MOEA	Mean	75.266	5.975	0.220	0.012
	SDev	0.512	0.189	0.007	0.0005
	Max	77	6.505	0.242	0.014
	Min	75	5.675	0.205	0.011

Table 7.6: Mean, standard deviation, maximum and minimum values over 30 runs for the fourth problem (ZDT1).

Algorithm		No. of points	Chi-Square	Spread	Crowding
paϵ-MyDE	Mean	**82.633**	11.592	**0.461**	**0.043**
	SDev	16.113	0.442	0.060	0.011
	Max	123	12.673	0.598	0.076
	Min	61	10.775	0.367	0.026
ϵ-MyDE	Mean	69.433	10.944	0.560	0.060
	SDev	3.008	0.127	0.065	0.010
	Max	77	11.300	0.664	0.081
	Min	64	10.786	0.444	0.046
ϵ-MOEA	Mean	60.033	**10.838**	0.523	0.067
	SDev	2.168	0.092	0.049	0.007
	Max	64	11.199	0.633	0.086
	Min	56	10.688	0.426	0.057

Table 7.7: Mean, standard deviation, maximum and minimum values over 30 runs for the fifth test problem (DTLZ2).

for bi-objective optimization problems, or

$$\{x^p + y^p + z^p = 1 : 0 \leq x, y, z \leq 1, 0 < p < \infty\}$$

for three dimensional problems. This way, we take advantage of the positive aspects of ϵ-dominance (already shown), while addressing some of its limitations.

On the one hand, paϵ-dominance finds a higher number of efficient points because the size of the boxes are adjusted specially in those areas where the Pareto front needs less solutions in any of its dimensions (almost horizontal or vertical regions of the Pareto front). Also, these solutions are better uniformly distributed along the Pareto front because the new grid balances the size of the boxes being more precise in those areas of the objective function space in which more solutions are needed. The publication derived from this chapter is [68].

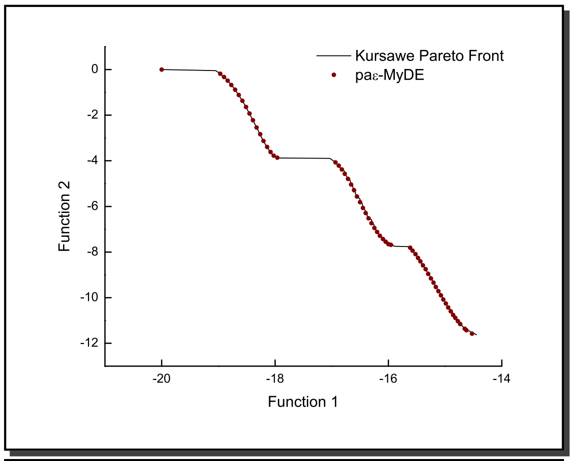
Algorithm 7 The paϵ-dominance algorithm - Part 1

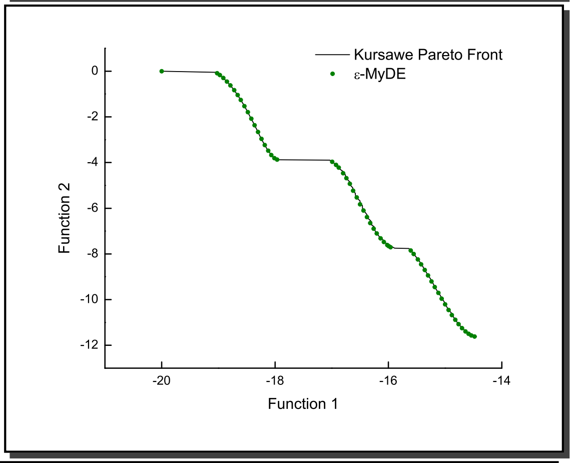
```
Require: T ← number of solutions given by user
 1: procedure PAϵ-DOMINANCE GRID ($P_{nPoints}$, $nPoints$, $nObjectives$)
 2:     $maxValue_f[i] \leftarrow P_{nPoints}$        ▷ maximum values per each nObjectives
 3:     $minValue_f[i] \leftarrow P_{nPoints}$        ▷ minimum values per each nObjectives
 4:     if nObjectives == 2 then
 5:         $area \leftarrow$ Area($P_{nPoints}, nPoints$)        ▷ Calculate area
 6:         p ← get_P($area$, $nObjectives$)
 7:     else
 8:         $hyper \leftarrow$ HyperVolume($P_{nPoints}, nPoints, nObjectives$)        ▷
    Hypervolume
 9:         p ← get_P($hyper$, $nObjectives$)
10:     end if
11:     v ← Speed_variation(p, T)        ▷ Calculate Speed of Variation
12:     aux ← $\{(p^v - 1) \cdot p^{(T-1)\cdot v}\}/\{p^{(T\cdot v)} - 1\}$
13:     for $i \leftarrow 0, nObjectives$ do        ▷ Calculate First $\epsilon_1$ in each dimension
14:         $\epsilon_1^i \leftarrow$ abs $\|maxValue_f[i] - minValue_f[i]\| \cdot$ aux;
15:     end for
16: end procedure

17: procedure GET_P($hyper$, $nObjectives$)
18:     if nObjectives == 2 then
19:         fp ← openFile p.txt        ▷ The file p.txt associates the correct value
    of p that corresponds to a specific value of hyper, only for 2 objectives
20:     else
21:         fp ← openFile 3p.txt        ▷ The file p.txt associates the correct value
    of p that corresponds to a specific value of hyper, only for 3 objectives
22:     end if
23:     repeat        ▷
24:         $line \leftarrow$ ReadNextLine (fp)        ▷ read line. eg: {0.5, 0.1667}
25:         if $hyper < line_2$ then        ▷ $line_2$ = Second value of line. eg: 0.1667
26:             return $lastline_1 + \frac{(hyper - lastline_2)\cdot(line_1 - lastline_1)}{(line_2 - lastline_2)}$
27:         end if
28:         $lastline \leftarrow line$
29:     until ¬ (endofFile $fp$)
30: end procedure
```

Algorithm 8 The paϵ-dominance algorithm - Part 2

```
31: procedure SPEED_VARIATION($p$, $T$)
32:     low ← 0.001
33:     up ← 1.0
34:     lowsign ← Dichotomy_fun ($p$, $T$, low);
35:     repeat
36:         medium ← ((up + low) / 2.0)
37:         auxsign ← Dichotomy_fun ($p$, $T$, medium)
38:         if (auxsign == lowsign) then
39:             low ← medium
40:         else
41:             up ← medium
42:         end if
43:     until ((up - low) $< 10^{-4}$)
44:     return ((up + low) / 2.0);
45: end procedure
46: procedure DICHOTOMY_FUN($p$, $T$, $x$)
47:     fun ← $((1 - 2^{(1/p)}) \cdot p^{(T*x)}) + 2^{(1/p)} \cdot p^{(T*x/2)} - 1$
48:     if ($0 <$ fun) then
49:         return 1
50:     else
51:         return 0
52:     end if
53: end procedure
```





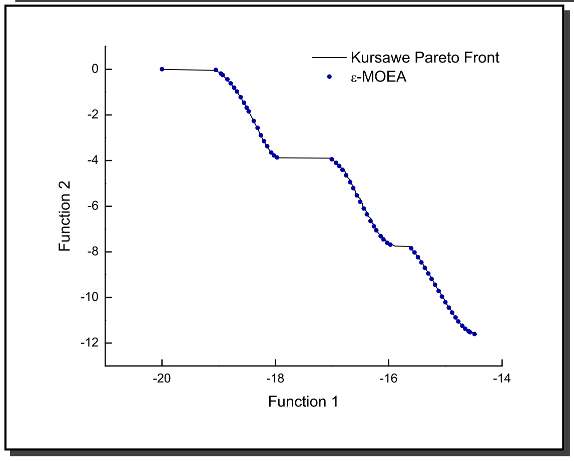


Figure 7.10: Efficient solutions generated by paϵ-MyDE (top), ϵ-MyDE (middle) and ϵ-MOEA (bottom) for the third test problem (Kursawe).

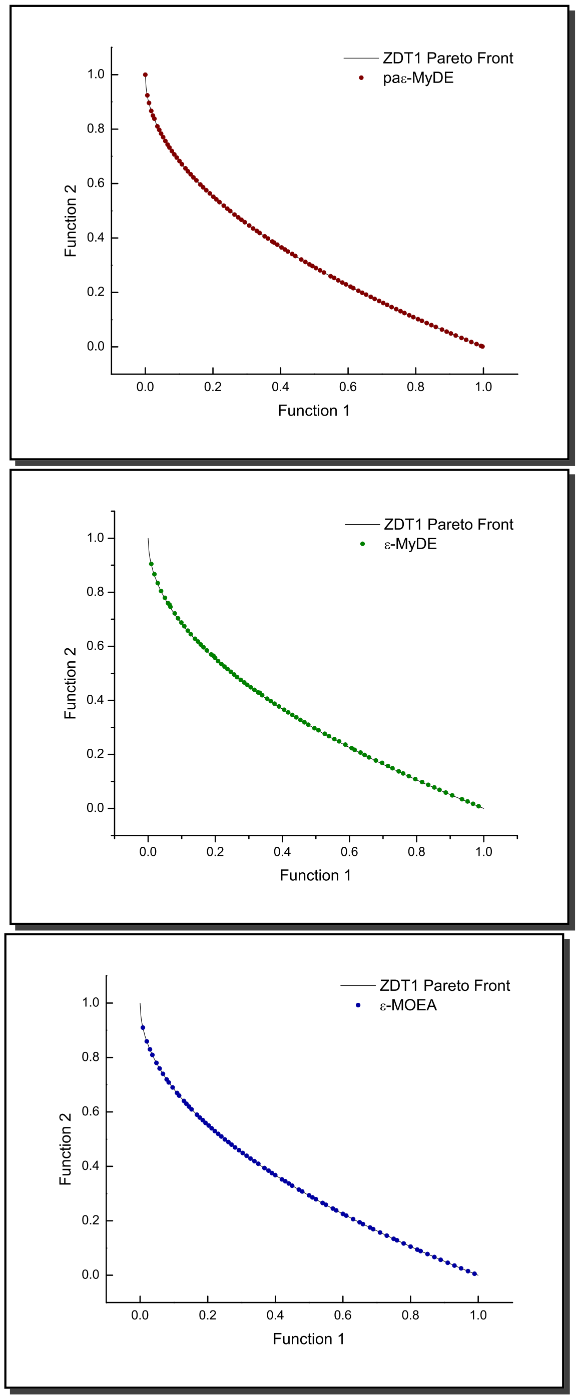
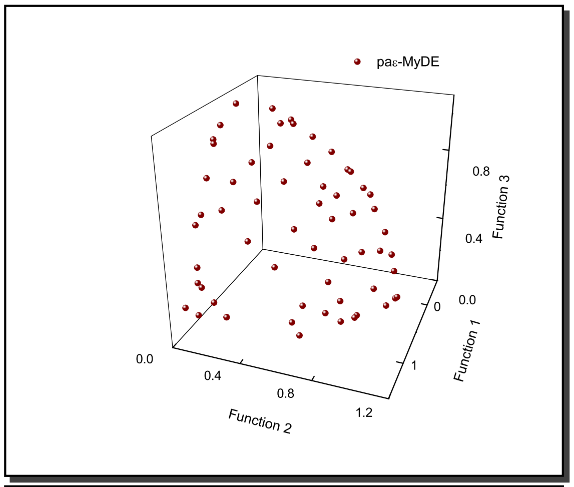
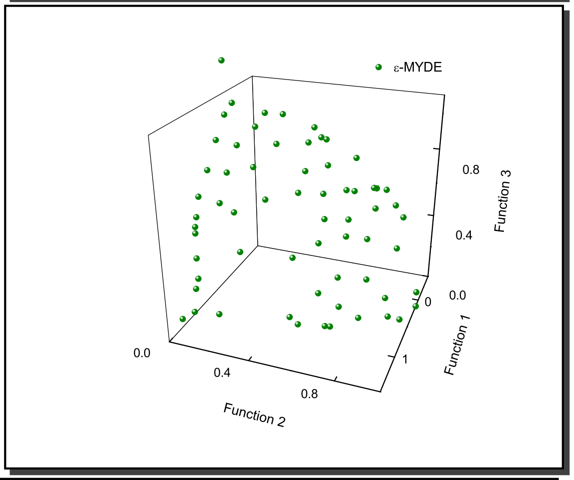


Figure 7.11: Efficient solutions generated by paϵ-MyDE (top), ϵ-MyDE (middle) and ϵ-MOEA (bottom) for the fourth test problem (ZDT1).





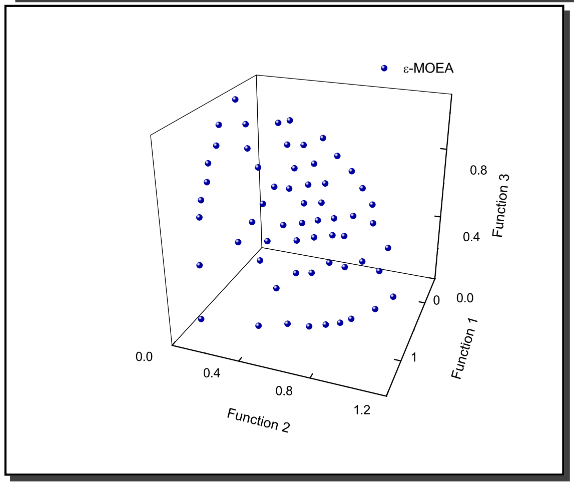


Figure 7.12: Efficient solutions generated by paϵ-MyDE (top), ϵ-MyDE (middle) and ϵ-MOEA (bottom) for the fifth test problem (DTLZ2).

8 Results

This chapter presents the results that we have obtained using the different algorithms that we have discussed in previous chapters. First, we present the comparison of results of our first proposal (the DEMORS algorithm) using both unconstrained and constrained functions. After that, we present the results that we get using the SVM-based multi-objective evolutionary algorithm (MOEA) combined with our proposed MOPSO. All the comparisons are made against a MOEA representative of the state-of-the-art: NSGA-II [39].

8.1 DEMORS: unconstrained problems

8.1.1 Methodology

Our proposed approach was validated using 9 test problems: five bi-objective problems from the **ZDT** set [173] and four problems with three objectives from the **DTLZ** set [41]. The definition and characteristics of these test functions are provided in Appendix A. Our proposed approach is compared with respect to those generated by NSGA-II[1] [39] and SPEA2[2] [174], which are MOEAs representative of the state-of-the-art in the area. In order to allow a quantitative comparison of results, we adopted the three following performance measures (a detailed description of these performance measures is provided in Section 3.3):

- **Unary additive epsilon indicator ($I^1_{\varepsilon+}$).**
- **Inverted generational distance (IGD).**
- **Spread (Δ).**

8.1.2 Parameters

In all cases, the parameters are shown in Table 8.1. In order to allow a fair comparison of results, all the approaches adopted real-numbers encoding and performed 5000 fitness function evaluations per run.

Parameter	DEMORS	NSGA-II	SPEA2
P	25	100	100
$P_{secondary}$	100	–	100
G_{max}	200	50	50
P_c	0.7	0.9	0.9
P_m	1/n	1/n	1/n
$elitism$	0.2	–	–
$Offspring$	1	–	–
$NumEff$	2	–	–
$NumDom$	10	–	–

Table 8.1: Parameters used by DEMORS, NSGA-II and SPEA2 for the unconstrained problems.

[1]The NSGA-II was obtained from http://www.iitk.ac.in/kangal/codes.shtml and the parameters suggested in the source code where the same used for these test functions.

[2]The SPEA2 was obtained from http://www.tik.ee.ethz.ch/sop/pisa/selectors/spea2/

8.1.3 Results

Table 8.2 shows a summary of our results for all the bi-objective problems and Table 8.3 shows the results for the problems with three objectives. For each test problem, we performed 30 independent runs per algorithm. The results reported in Tables 8.2 and 8.3 are the mean values for each of the three performance measures, the standard deviation, the best and the worst value of the 30 runs performed; the best mean values in each case are shown in **boldface**. The graphical results are shown in Figures 8.1, 8.2, 8.3, 8.4 and 8.5 for the ZDT set problems and in Figures 8.6, 8.7, 8.8 and 8.9 for the DTLZ set problems. These plots correspond to the run in the mean value with respect to the unary additive epsilon indicator. In all the bi-objective optimization problems, the true Pareto front is shown with a continuous line and the approximation obtained by each algorithm is shown with circles.

8.1.3.1 ZDT set problems

It can be clearly seen in Table 8.2 that the performance measures show that our DEMORS produced the best mean values in most cases in the bi-objective problems. Regarding IGD, there were only two cases in which the SPEA2 outperformed our approach. With respect to the unary additive epsilon indicator and spread metrics, our DEMORS outperformed the NSGA-II and SPEA2 in all cases.

The graphical results shown in Figures 8.1, 8.2, 8.3, 8.4 and 8.5. We can clearly see that in problems ZDT2, ZDT3, ZDT4 and ZDT6, the NSGA-II is very far from the true Pareto front; and in ZDT3, ZDT4 and ZDT6, SPEA2 obtains results that are not close to the true Pareto front, whereas our DEMORS has already converged to the true Pareto front after only 5000 fitness function evaluations in all the test problems.

Conclusion: ZDT set

If we were asked to sort the MOEAs compared based on their performance when using the ZDT set problems the order would be the following:

1	2	3
DEMORS	SPEA2	NSGA-II

	Metric - Algorithm		mean	Std. Desv.	best	worst
ZDT1	$I^1_{\varepsilon+}$	DEMORS	**0.3180**	0.1900	0.0102	0.5980
		NSGA-II	0.5968	0.4634	0.1376	1.8928
		SPEA2	0.8180	0.4330	0.1457	2.0039
	IGD	DEMORS	0.0010	0.00082	0.0005	0.0044
		NSGA-II	0.0003	0.0003	0.0002	0.0004
		SPEA2	**0.0002**	0.0001	0.0002	0.0003
	Δ	DEMORS	**0.5857**	0.1750	0.2561	0.8334
		NSGA-II	0.6566	0.1835	0.4143	1.0673
		SPEA2	0.6352	0.1723	0.2835	0.8877

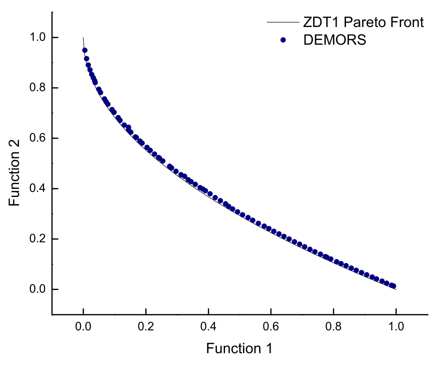
	Metric - Algorithm		mean	Std. Desv.	best	worst
ZDT2	$I^1_{\varepsilon+}$	DEMORS	**0.0297**	0.0377	0.0025	0.1538
		NSGA-II	0.2044	0.2644	0.0030	0.7833
		SPEA2	0.5430	0.3843	0.0101	1.3142
	IGD	DEMORS	**0.0119**	0.0008	0.0113	0.0141
		NSGA-II	0.0305	0.0140	0.0114	0.0408
		SPEA2	0.0217	0.0140	0.0114	0.0410
	Δ	DEMORS	**0.4461**	0.1448	0.2713	0.8255
		NSGA-II	0.8472	0.2205	0.4104	1.0331
		SPEA2	0.7605	0.2084	0.4185	1.0270

	Metric - Algorithm		mean	Std. Desv.	best	worst
ZDT3	$I^1_{\varepsilon+}$	DEMORS	**0.1337**	0.0383	0.0869	0.2377
		NSGA-II	1.6298	0.2871	1.1293	2.1879
		SPEA2	1.9444	0.5866	1.1597	3.8751
	IGD	DEMORS	**0.0084**	0.0007	0.0072	0.0101
		NSGA-II	0.0138	0.0001	0.0136	0.0140
		SPEA2	0.0139	0.0001	0.0136	0.0142
	Δ	DEMORS	**0.7734**	0.0469	0.6866	0.8599
		NSGA-II	0.8003	0.0716	0.6987	0.9178
		SPEA2	0.8278	0.1118	0.6012	1.0789

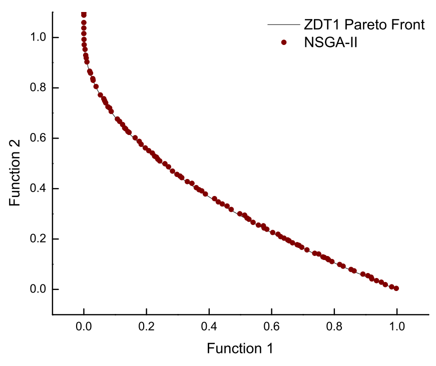
	Metric - Algorithm		mean	Std. Desv.	best	worst
ZDT4	$I^1_{\varepsilon+}$	DEMORS	**2.6054**	2.6306	0.0135	9.1621
		NSGA-II	4.4173	4.1008	0.0350	15.8153
		SPEA2	15.2087	15.7190	1.0714	50.7055
	IGD	DEMORS	**0.0119**	0.0008	0.0113	0.0141
		NSGA-II	0.0305	0.0140	0.0114	0.0408
		SPEA2	0.0217	0.0140	0.0114	0.0410
	Δ	DEMORS	**0.4461**	0.1448	0.2713	0.8255
		NSGA-II	0.8472	0.2205	0.4104	1.0331
		SPEA2	0.7605	0.2084	0.4185	1.0270

	Metric - Algorithm		mean	Std. Desv.	best	worst
ZDT6	$I^1_{\varepsilon+}$	DEMORS	**2.1209**	2.2593	0.0020	6.3006
		NSGA-II	5.4218	1.8146	0.8527	7.7676
		SPEA2	5.8388	1.7322	0.1736	8.2088
	IGD	DEMORS	0.0463	0.0348	0.0116	0.1242
		NSGA-II	0.0166	0.0013	0.0140	0.0187
		SPEA2	**0.0156**	0.0010	0.0142	0.0172
	Δ	DEMORS	**0.8748**	0.1787	0.4688	1.1747
		NSGA-II	1.0880	0.1113	0.7711	1.3080
		SPEA2	1.1091	0.1256	0.6700	1.2521

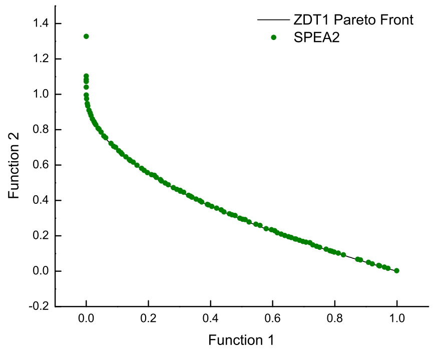
Table 8.2: Performance measures: $I^1_{\varepsilon+}$, IGD and Spread for the ZDT test problems, comparing: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

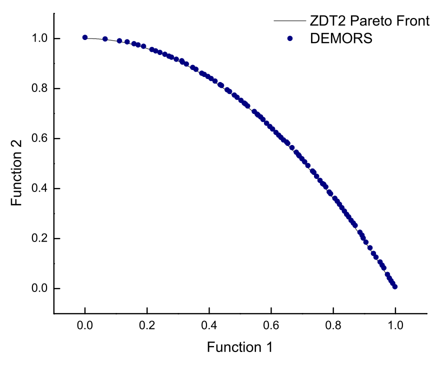


(b) NSGA-II

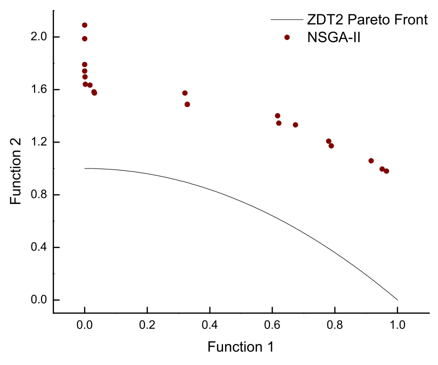


(c) SPEA2

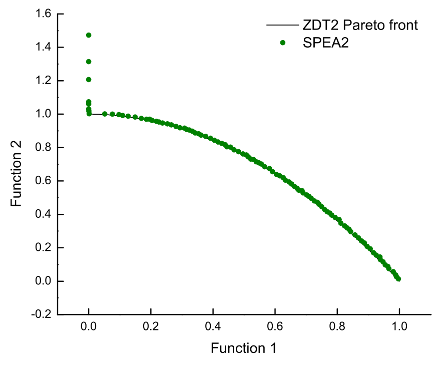
Figure 8.1: Pareto fronts obtained for ZDT1: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

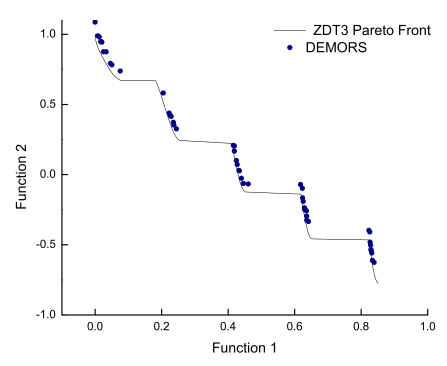


(b) NSGA-II

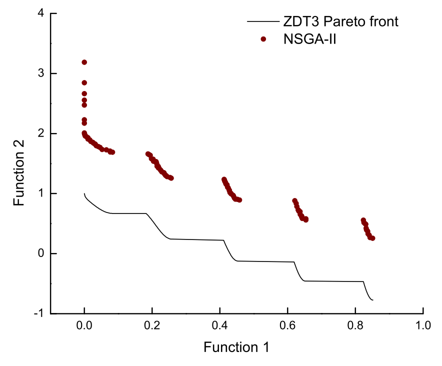


(c) SPEA2

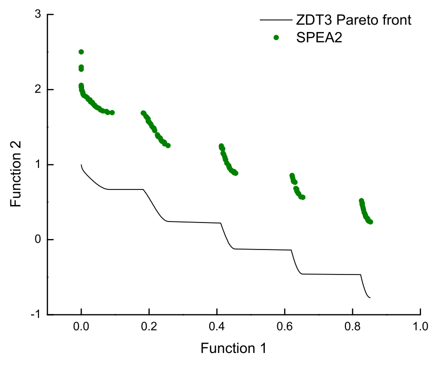
Figure 8.2: Pareto fronts obtained for ZDT2: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

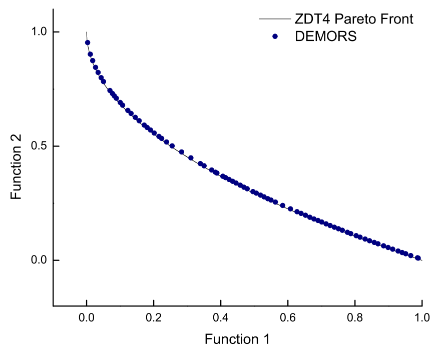


(b) NSGA-II

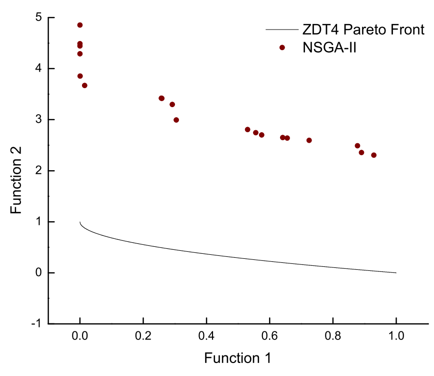


(c) SPEA2

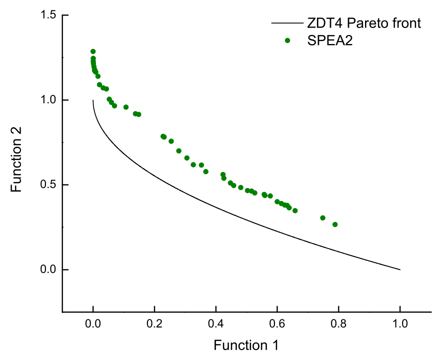
Figure 8.3: Pareto fronts obtained for ZDT3: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

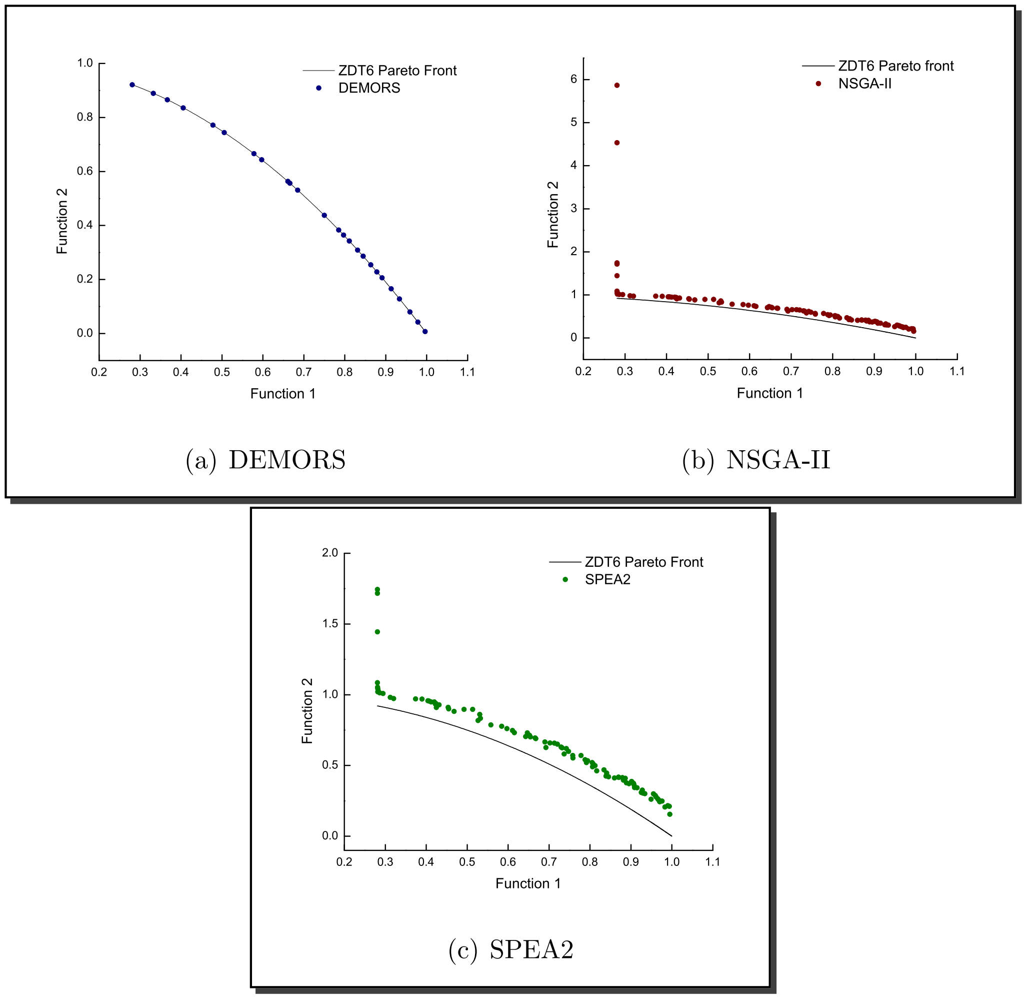


(b) NSGA-II



(c) SPEA2

Figure 8.4: Pareto fronts obtained for ZDT4: DEMORS, NSGA-II and SPEA2.



(a) DEMORS (b) NSGA-II

(c) SPEA2

Figure 8.5: Pareto fronts obtained for ZDT6: DEMORS, NSGA-II and SPEA2.

8.1.3.2 DTLZ set problems

With respect to the performance measures (see Table 8.3), DEMORS obtained the best results in all the test functions, showing that the distance that DEMORS needs to get the Pareto front is closer than the other algorithms. With respect to the IGD metric, DEMORS obtained the best results in most of the test functions, except for DTLZ1, in which SPEA2 obtains the best result in DTLZ1. With respect to the diversity metric, DEMORS is outperformed only in DTLZ2 and DTLZ4. It is clear, that when the algorithms deal with these problems with three objectives, the difficulty is increased, and none of them could reach the true Pareto front in DTLZ1, in which is much harder to reach the true Pareto front with only 5,000 function evaluations.

Graphically, we can see in Figures 8.6, 8.7, 8.8 and 8.9 the different plots obtained from the three different algorithms. We can see that none of the algorithms could reach the true Pareto front in DTLZ1, but DEMORS gets closer than the others. In DTLZ2 and DTLZ3, DEMORS is also closer to the true Pareto front. In DTLZ4, SPEA2 shows a better distribution of solutions.

Conclusion: DTLZ set
The spread of solutions of our DEMORS is evidently not the best possible, but we argue that this is a good trade-off (and the performance measures back up this statement) if we consider the low computational cost achieved. Evidently, quality of the spread of solutions is sacrificed at the expense of reducing the computational cost required to obtain a good approximation of the Pareto front.

If we were asked to sort the MOEAs compared based on their performance when using the DTLZ set problems the order would be the following:

1	2	3
DEMORS	SPEA2	NSGA-II

8.2 DEMORS: constrained problems

8.2.1 Methodology

Our approach was validated using seven constrained test problems: Binh2 [12], Kita [85], Osyczka1 and Osyczka2 [116], Srinivas [149], Tanaka [155] and

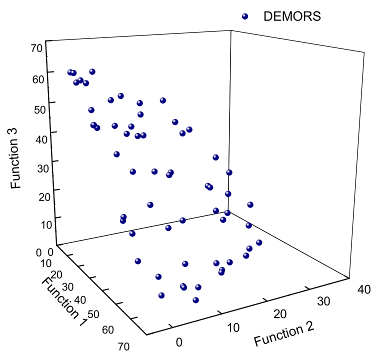
	Metric - Algorithm		mean	Std. Desv.	best	worst
DTLZ1	$I^1_{\varepsilon+}$	DEMORS	**80.59**	32.4032	61.02	189.43
		NSGA-II	407.31	30.8137	346.37	487.66
		SPEA2	383.30	34.8352	316.01	433.37
	IGD	DEMORS	0.6629	0.0245	0.6219	0.7046
		NSGA-II	0.5255	0.1185	0.2161	0.6955
		SPEA2	**0.4873**	0.0983	0.2726	0.6786
	Δ	DEMORS	**0.6077**	0.0409	0.5406	0.7058
		NSGA-II	0.9131	0.0448	0.8275	1.0169
		SPEA2	0.9096	0.0384	0.8331	0.9777

	Metric - Algorithm		mean	Std. Desv.	best	worst
DTLZ2	$I^1_{\varepsilon+}$	DEMORS	**0.1672**	0.1357	0.0489	0.5430
		NSGA-II	0.9580	0.2109	0.6322	1.4049
		SPEA2	0.7255	0.1648	0.4733	0.9976
	IGD	DEMORS	**0.0130**	0.0001	0.0129	0.0132
		NSGA-II	0.0138	0.0002	0.0136	0.0142
		SPEA2	0.0135	0.0001	0.0133	0.0136
	Δ	DEMORS	0.4803	0.0302	0.4108	0.5254
		NSGA-II	0.6127	0.0365	0.5447	0.6794
		SPEA2	**0.2951**	0.0224	0.2643	0.3571

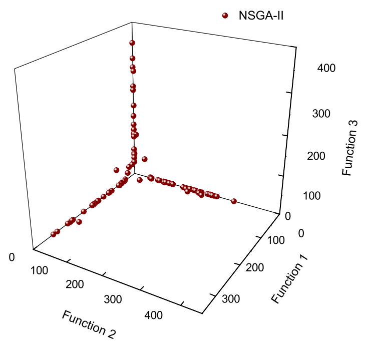
	Metric - Algorithm		mean	Std. Desv.	best	worst
DTLZ3	$I^1_{\varepsilon+}$	DEMORS	**166.89**	29.49	117.41	225.73
		NSGA-II	1380.38	103.77	1168.54	1540.6
		SPEA2	1306.95	134.65	972.52	1590.6
	IGD	DEMORS	**2.5107**	0.2001	2.1526	3.0404
		NSGA-II	4.6846	0.5199	3.9746	5.7531
		SPEA2	4.8363	0.7413	3.2458	6.2143
	Δ	DEMORS	**0.7469**	0.0752	0.5755	0.8904
		NSGA-II	0.9347	0.0433	0.8187	0.9858
		SPEA2	0.8941	0.0666	0.7453	1.0493

	Metric - Algorithm		mean	Std. Desv.	best	worst
DTLZ4	$I^1_{\varepsilon+}$	DEMORS	**0.1078**	0.0954	0.0497	0.4796
		NSGA-II	0.4504	0.1995	0.1172	0.8690
		SPEA2	0.5767	0.2501	0.0517	1.0716
	IGD	DEMORS	**0.0132**	0.0004	0.0128	0.0143
		NSGA-II	0.0141	0.0011	0.0134	0.0174
		SPEA2	0.0136	0.0009	0.0132	0.0174
	Δ	DEMORS	0.7107	0.0645	0.6094	0.8238
		NSGA-II	0.5753	0.1119	0.4821	0.8028
		SPEA2	**0.3178**	0.1286	0.2170	0.7283

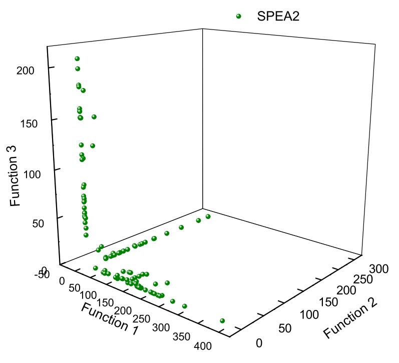
Table 8.3: Performance measures: $I^1_{\varepsilon+}$, IGD and Spread for the DTLZ test problems, comparing: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

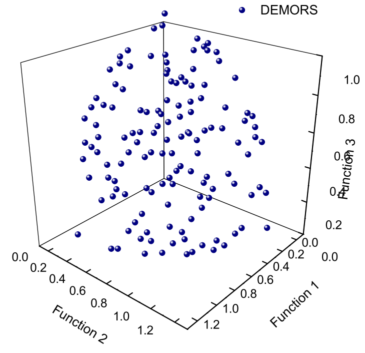


(b) NSGA-II

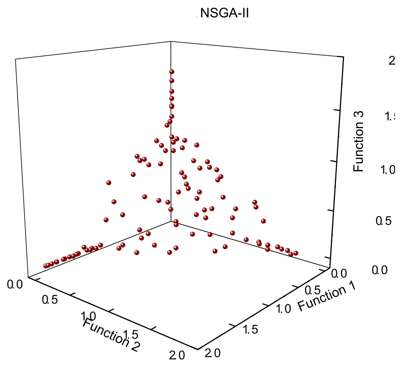


(c) SPEA2

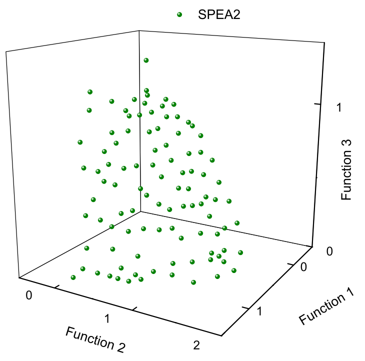
Figure 8.6: Pareto fronts obtained for DTLZ1: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

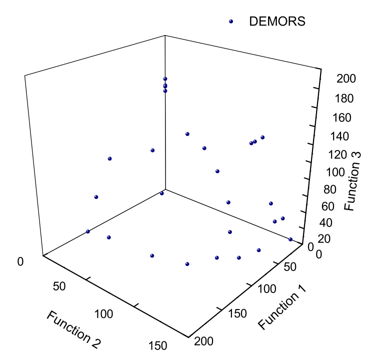


(b) NSGA-II

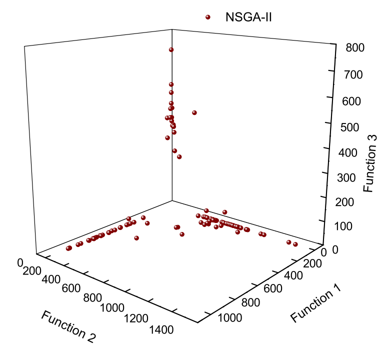


(c) SPEA2

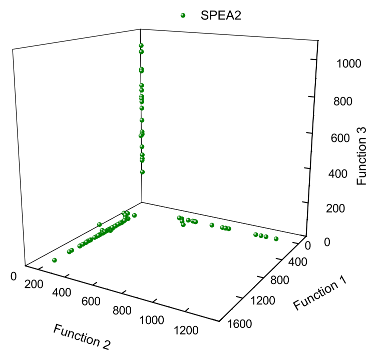
Figure 8.7: Pareto fronts obtained for DTLZ2: DEMORS, NSGA-II and SPEA2.



(a) DEMORS

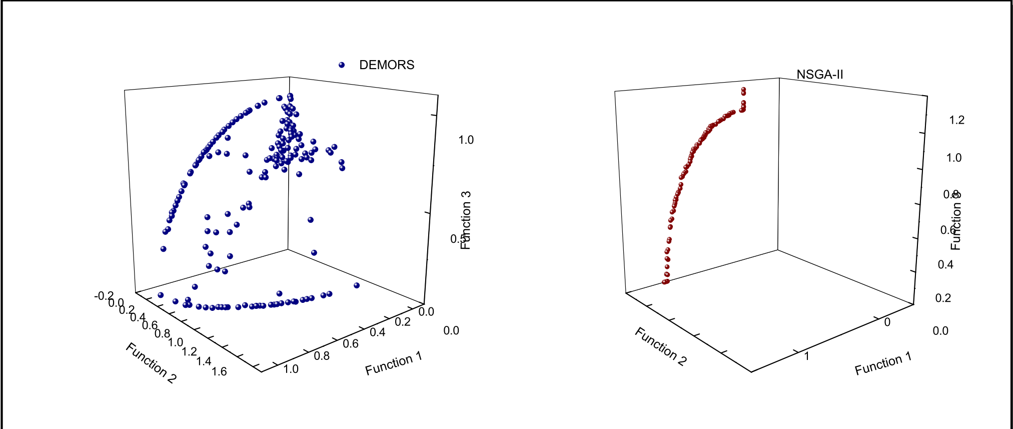


(b) NSGA-II

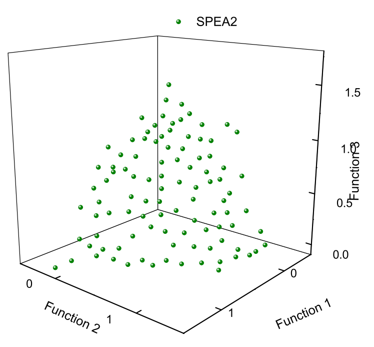


(c) SPEA2

Figure 8.8: Pareto fronts obtained for DTLZ3: DEMORS, NSGA-II and SPEA2.



(a) DEMORS (b) NSGA-II



(c) SPEA2

Figure 8.9: Pareto fronts obtained for DTLZ4: DEMORS, NSGA-II and SPEA2.

the Welded beam problem [124]. All of them, except for Kita, are minimization problems. The definitions of these problems can be found in Appendix A. The problems have been selected trying to cover all different complexities: convex, non-convex and disconnected Pareto fronts, linear and nonlinear objective functions and constraints and different number of decision variables (from 2 to 6).

In order to allow a quantitative comparison of results, we adopted the three following performance measures (a detailed description of the performance measures can be seen in Section 3.3):

- **Size of the space covered (SSC).**
- **Unary additive epsilon indicator ($I^1_{\varepsilon+}$).**
- **Spread (Δ).**

8.2.2 Parameters

In all cases, the parameters are shown in Table 8.4. In order to allow a fair comparison of results, both approaches adopted real-numbers encoding and performed 10,000 fitness function evaluations per run. It is important to mention that the parameter *elitism* used for DEMORS to handle constraint problems is increased (from 0.2 to 0.5) due the necessity to find more non-dominated solutions during the random selection in the first phase of the algorithm.

8.2.3 Results

Table 8.5 shows a summary of the performance measures results for all the constrained problems. For each test problem, we performed 30 independent runs per algorithm. The results reported in Table 8.5 are the mean values for each of the three performance measures and the standard deviation of the 30 runs performed; the best mean values in each case are shown in **boldface**. The graphical results are shown in Figures 8.10, 8.11, 8.12 and 8.13. These plots correspond to the run in the mean value with respect to the unary additive epsilon indicator. In all the optimization problems, the true Pareto front is shown with a continuous line and the approximation obtained by each algorithm is shown with circles.

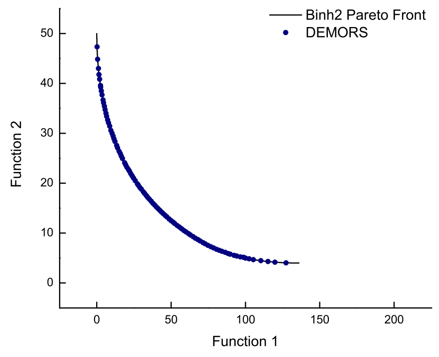
Parameter	**DEMORS**	**NSGA-II**
P	25	100
$P_{secondary}$	100	–
G_{max}	400	100
P_c	0.8	0.9
P_m	1/n	1/n
$elitism$	0.5	–
$Offspring$	1	–
$NumEff$	2	–
$NumDom$	5	–
$NumInfea$	5	–

Table 8.4: Parameters used by DEMORS and NSGA-II for the constrained problems.

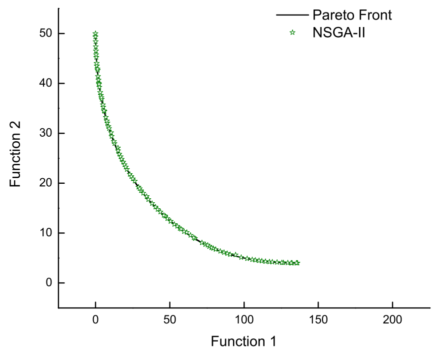
It can be clearly seen in Table 8.5 that our DEMORS produced the best mean values in 3 cases (Binh2, Kita and Srinivas) for all the performance measures, although NSGA-II obtained similar results. For the other 4 problems (Osyczka1, Osyczka2, Tanaka and Welded beam), NSGA-II obtained the best mean values for the SSC and $I^1_{\varepsilon+}$ performance measures but DEMORS improved all its Δ values, while results on the first two performance measures are very similar, too. This is, the results with respect to the SSC and $I^1_{\varepsilon+}$ performance measures are very similar for all the problems, but DEMORS performed better according to the distribution performance measure Δ for all the problems. These results led us to conclude that the hybridization of Differential Evolution and Rough Sets Theory is a competitive alternative to deal with constrained problems. While Differential Evolution rapidly converges to the Pareto optimum, Rough Sets Theory demonstrated to be a good local optimizer able to improve the convergence and spread of the initial approximation obtained in constrained problems.

Conclusion: Constrained Problems

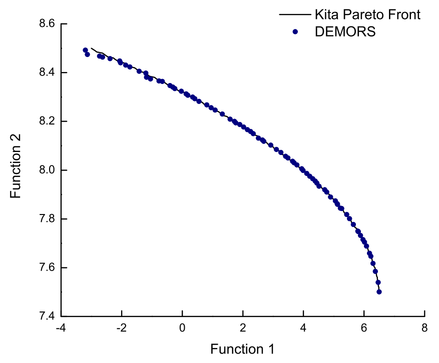
Our results indicate that DEMORS is a competitive MOEA for constrained multi-objective optimization problems when performing only 10,000 fitness function evaluations, and it is also able to produce a good spread of solutions within such a reduced number of evaluations.



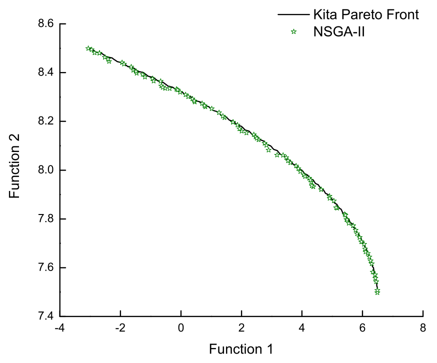
(a) DEMORS



(b) NSGA-II

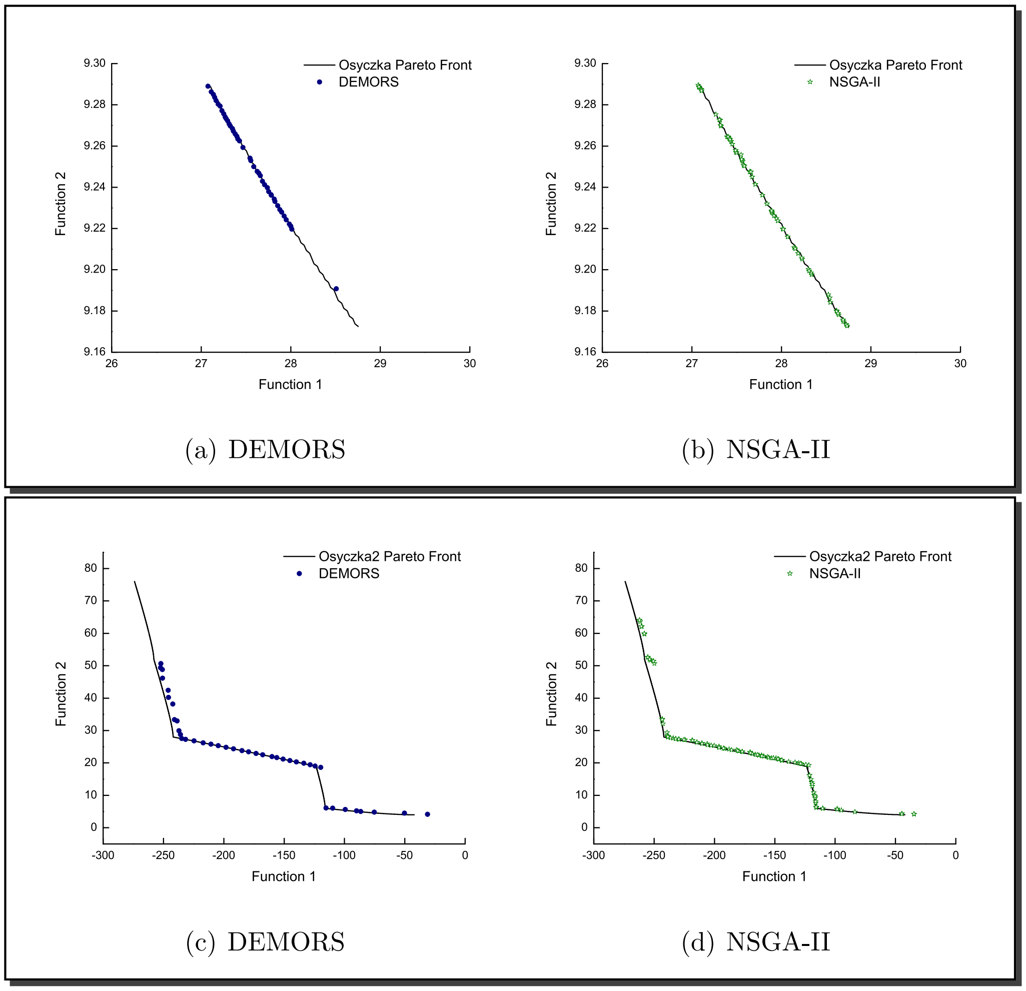


(c) DEMORS



(d) NSGA-II

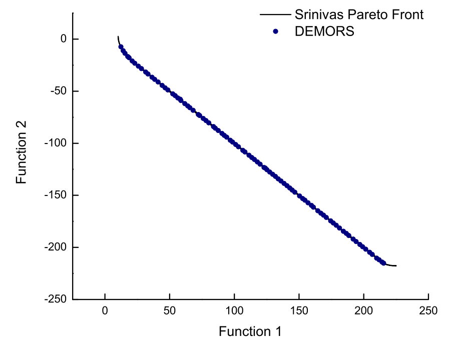
Figure 8.10: Pareto fronts for Binh2 and Kita: DEMORS and NSGA-II.



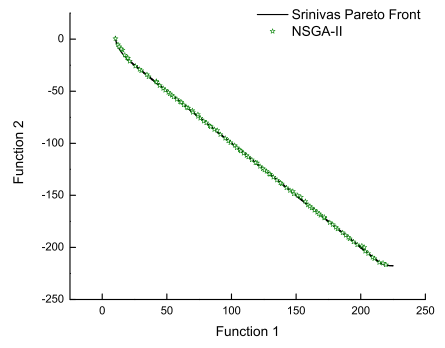
(a) DEMORS (b) NSGA-II

(c) DEMORS (d) NSGA-II

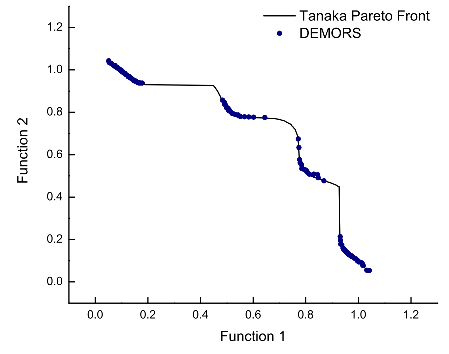
Figure 8.11: Pareto fronts for Osyczka 1 and 2: DEMORS and NSGA-II.



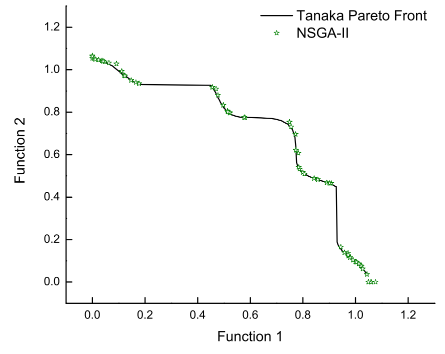
(a) DEMORS



(b) NSGA-II



(c) DEMORS



(d) NSGA-II

Figure 8.12: Pareto fronts for Srinivas and Tanaka: DEMORS and NSGA-II.

Function	SSC				$I^1_{\varepsilon+}$			
	DEMORS		NSGA-II		DEMORS		NSGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ
Binh2	**0.808**	0.000	0.806	0.000	**0.007**	0.001	0.013	0.002
Kita	**0.879**	0.002	**0.879**	0.001	**0.013**	0.006	**0.013**	0.003
Osyczka1	0.821	0.084	**0.875**	0.021	0.107	0.099	**0.051**	0.027
Osyczka2	0.946	0.012	**0.963**	0.000	0.034	0.022	**0.009**	0.003
Srinivas	**0.557**	0.000	0.556	0.000	**0.012**	0.003	0.013	0.002
Tanaka	0.718	0.008	**0.749**	0.001	0.021	0.012	**0.012**	0.002
Welded Beam	0.953	0.020	**0.974**	0.003	0.030	0.021	**0.009**	0.003

Function	Δ			
	DEMORS		NSGA-II	
	Mean	σ	Mean	σ
Binh2	**0.452**	0.018	0.492	0.036
Kita	**0.964**	0.068	1.038	0.060
Osyczka	**0.716**	0.164	0.816	0.062
Osyczka2	**0.771**	0.287	1.210	0.068
Srinivas	**0.165**	0.015	0.277	0.015
Tanaka	**0.918**	0.115	1.140	0.045
Welded Beam	**0.732**	0.188	0.898	0.133

Table 8.5: Comparison of results between DEMORS and the NSGA-II for constrained problems adopted. The best values are in **boldface**. σ refers to the standard deviation over the 30 runs performed.

8.3 SVM+RS: unconstrained bi-objective problems

After our first experiments with the surrogate methods (see Section 5.4) and the election of the appropriate leader in PSO (see Section 4.2.3), it became clear that despite the fact that a good convergence was achieved, this was obtained at the expense of a poor distribution of the results. Thus, we decided to use rough sets as a local search engine (see Section 6.1), in order to find the solutions that are missing in the approximations obtained during the first phase of our approach performing only 600 evaluations. So, we decided to perform another 1,400 fitness function evaluations, aiming to fill up the gaps along the Pareto front.

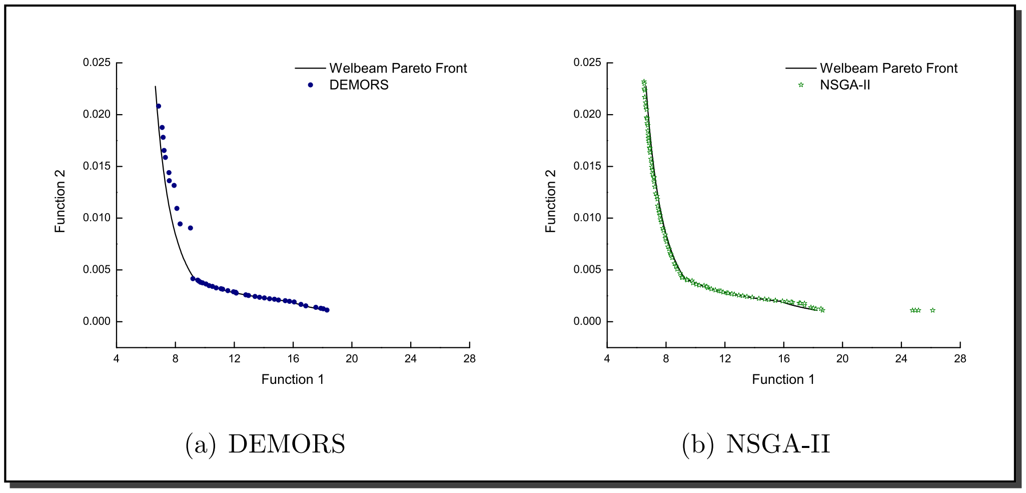


Figure 8.13: Pareto fronts for Welded Beam: DEMORS and NSGA-II.

8.3.1 Methodology

Our approach was validated using five bi-objective problems from the **ZDT** set [173]. The definition and characteristics of these test functions are provided in Appendix A. Our proposed approach is compared with respect to those generated by NSGA-II, which is a MOEA representative of the state-of-the-art in the area. In order to allow a quantitative comparison of results, we adopted the three following performance measures and (a detailed description of the performance measures can be seen in Section 3.3):

- **Two Set coverage (SC).**
- **Inverted generational distance (IGD).**
- **Spread (Δ).**

8.3.2 Parameters

In all cases, the parameters are shown in Table 8.6. In order to allow a fair comparison of results, both approaches adopted real-numbers encoding and performed 2,028 fitness function evaluations per run in total[3]. We de-

[3]We performed 2,028 fitness function evaluations because the NSGA-II requires a population size which is multiple of 4. Using $P = 52$ and $G_{max} = 39$, we get 2,028 evaluations.

cided to use a population of 52 for the NSGA-II to give an opportunity to develop te solutions for more generations, in this case 39, and get a better approximation.

Parameter	SVM+RS	NSGA-II
P	100	52
G_{max}	–	39
Internal P_{mopso}	100	–
Internal G_{mopso}	100	–
PSO Flight w, c_1, c_2	0.1, 0.1, 1.4	–
P_c	–	0.9
P_m	1/n	1/n
$Offspring$	1	–
$NumEff$	2	–
$NumDom$	10	–
Evaluations	2,028	2,028

Table 8.6: Parameters used by SVM+RS and NSGA-II for the ZDT test problems.

8.3.3 Results

The results reported in Table 8.7 are the mean values for each of the three performance measures adopted (SC, IGD and Δ) and the standard deviation of the 10 runs performed with 2028 evaluations in total for both approaches (SVM+RS and NSGA-II). The best mean values in each case are shown in boldface in Table 8.7. The graphical results are shown in Figures 8.14, 8.15 and 8.16. These plots correspond to the run in the mean value with respect to the inverted generational distance metric. In all the optimization problems, the true Pareto front is shown with a continuous line and the approximation obtained by each algorithm is shown with circles.

It can be observed that in the ZDT test problems, our approach produced the best results with respect to the SC performance measure in all cases except for ZDT4. The same applies for the IGD performance measure. Also, our approach outperformed the NSGA-II with respect to the spread performance measure in three cases (ZDT2, ZDT3 and ZDT4). Graphically, it can be seen that our approach gets closer than the NSGA-II to the true Pareto front in ZDT1, ZDT2, ZDT3 and ZDT6, but not in ZDT4. The poor performance of both approaches in ZDT4 might be attributed to the bad scalability presented by both approaches.

Conclusion: Surrogate Methods
Our results indicate that the NSGA-II, despite being a highly competitive MOEA, is not able to converge to the true Pareto front in most of the test problems adopted when performing only 2,000 fitness function evaluations. This hybrid algorithm (SVM+RS) provides competitive results in most of the test problems adopted. We believe that our approach can be a good choice in real-world problems in which each evaluation of the fitness function is very expensive (computationally speaking).

8.4 Final remarks

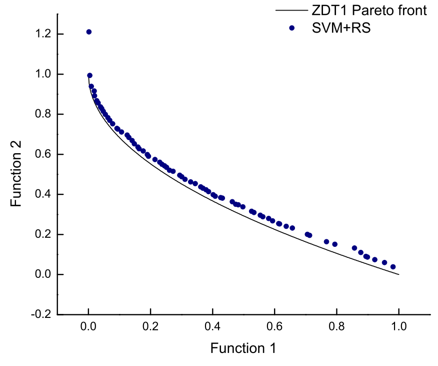
These results seem to indicate that our approaches could be a viable alternative for real-world applications in which each evaluation of the fitness function is very expensive (computationally speaking). In such applications, we can afford sacrificing a good distribution of solutions for the sake of obtaining a reasonably good approximation of the Pareto front with a low number of evaluations. However, there exists a well-known theorem, the "NO FREE LUNCH THEOREM" (NFL) proposed by Wolpert and Macready [168], which states that: "any two optimization algorithms are equivalent when their performance is averaged across all possible problems." If the condition for NFL holds approximately, then all algorithms yield approximately the same results over all the test functions.

So, we can conclude that our algorithms are suitable to deal with the multi-objective problems in which a low number of fitness evaluations are required. But, we can not conclude that our algorithms will perform always better than those representative of the state-of-the-art, as eventually the other algorithms will get a comparable or better performance than ours when the number of fitness function evaluations is increased.

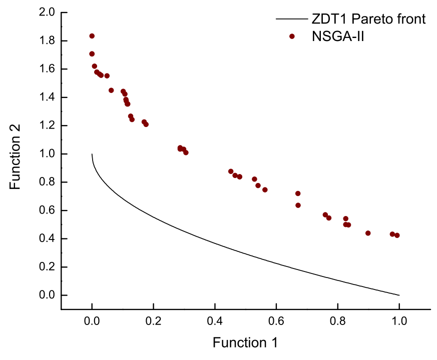
Function	SC				IGD			
	SVM+RS		NSGA-II		SVM+RS		NSGA-II	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ
ZDT1	**0.8755**	0.1785	0.0725	0.1676	**0.0061**	0.0046	0.0168	0.0021
ZDT2	**0.9690**	0.0505	0.0000	0.0000	**0.0026**	0.0027	0.0382	0.0062
ZDT3	**0.6453**	0.3343	0.0372	0.0668	**0.0145**	0.0071	0.1190	0.0113
ZDT4	0.0167	0.0500	**0.7937**	0.2512	0.7397	0.1554	**0.1511**	0.0405
ZDT6	**0.9208**	0.1886	0.0000	0.0000	**0.0135**	0.0025	0.0548	0.0092

Function	Δ			
	DEMORS		NSGA-II	
	Mean	σ	Mean	σ
ZDT1	0.8308	0.1987	**0.8194**	0.0324
ZDT2	**0.5684**	0.2338	0.9711	0.0696
ZDT3	**0.8684**	0.0943	0.9475	0.0296
ZDT4	**0.9572**	0.0411	1.0795	0.1026
ZDT6	1.1000	0.1297	**0.9588**	0.0274

Table 8.7: Comparison of results between the SVM+RS algorithm and the NSGA-II for the five test problems adopted (2000 evaluations). The best values are in **boldface**. σ refers to the standard deviation over the 30 performed runs.

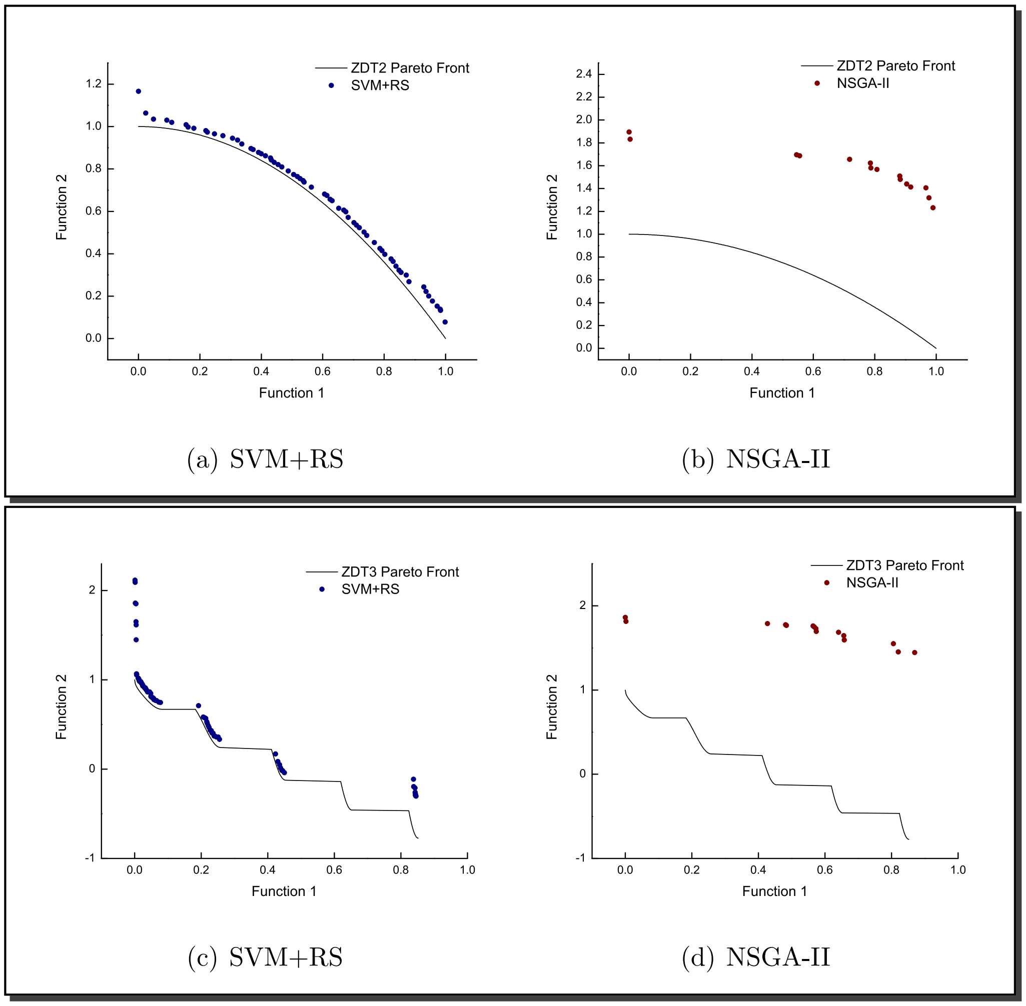


(a) SVM+RS



(b) NSGA-II

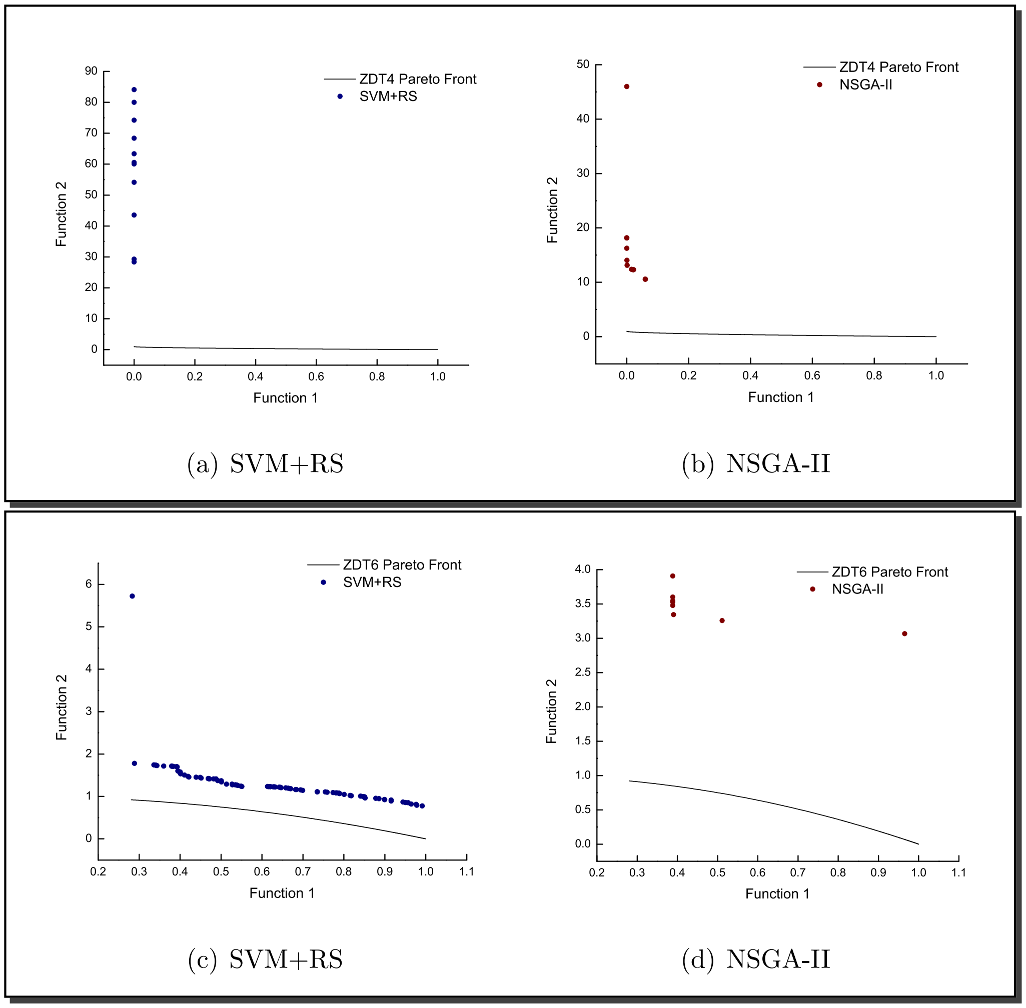
Figure 8.14: Pareto fronts obtained for ZDT1: SVM+RS and NSGA-II.



(a) SVM+RS (b) NSGA-II

(c) SVM+RS (d) NSGA-II

Figure 8.15: Pareto fronts obtained for ZDT2 and ZDT3: SVM+RS and NSGA-II.



(a) SVM+RS (b) NSGA-II (c) SVM+RS (d) NSGA-II

Figure 8.16: Pareto fronts obtained for ZDT4 and ZDT6: SVM+RS and NSGA-II.

9 Final Remarks

9.1 Summary

The main goal of this research has been to design techniques that improve the efficiency (defined in terms of the number of fitness function evaluations performed) of a multi-objective evolutionary algorithm when solving problems of relatively high dimensionality (up to 30 decision variables). When solving a multi-objective problem, it is very important to do a good exploration at the beginning of the search and find quickly. After that, an exploitation of those good solutions is needed to accelerate the convergence and finally get the true Pareto front of the multi-objective problem. Additionally, a mechanism to keep the nondominated solutions is normally used to retain a well-distributed set of solutions along the Pareto front. We have presented different mechanisms that tackle each of these problems: exploration, exploitation and distribution. In Figure 9.1, we show the different approaches that were proposed in this work, and a brief description of them is provided next:

- We have introduced an approach that uses differential evolution to solve both unconstrained and constrained multi-objective optimization

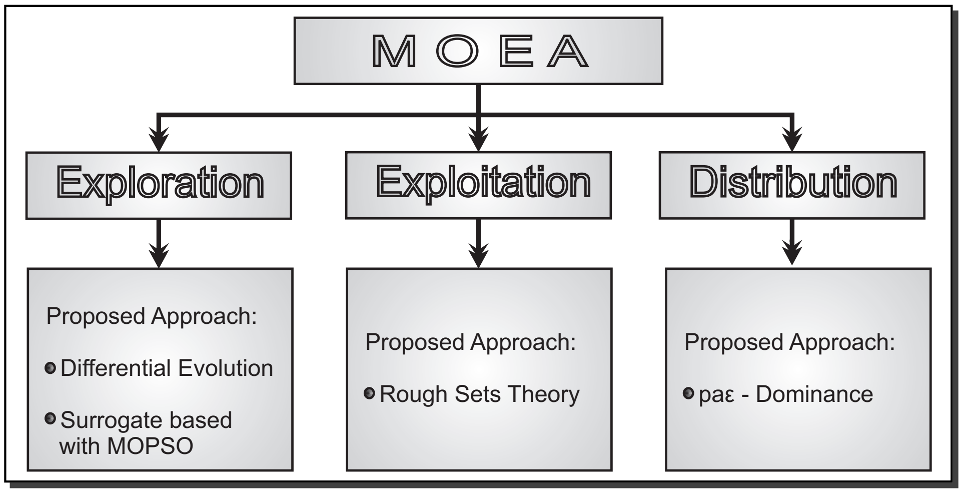


Figure 9.1: Different mechanisms proposed to deal with efficiency on MOEAs.

problems. The high convergence rate that characterizes the differential evolution algorithm was controlled using two elitist selection schemes. Our proposed approach was able to produce good results and was able to find good solutions with a high convergence rate in a low number of fitness function evaluations. The constraint-handling scheme adopted in our algorithm allowed a successful exploration of the search space even when the optimum lies on the boundary between the feasible and infeasible regions (refer to Section 4.1.3 in page 61 for further details).

- We used four different schemes to select a leader in the PSO flight equation in combination with the surrogate model to approximate the functions using supervised learning. From the initial comparison, we concluded that the PSO scheme based on selecting a leader that dominates the particle was the most appropriate model to use as a search engine, because it provides a good (although insufficient) approximation of the Pareto front. (refer to Section 4.2.2 in page 71 for further details).

- We have presented a study of surrogate methods to solve multi-objective problems with high dimensionality. Three different methods were used in our comparative study: artificial neural networks (ANNs), radial basis function (RBFs) and support vector machines (SVMs), all of them in their regression form, in order to approximate the function using supervised learning. From this study, we concluded that the SVMs were the most appropriate model for dealing with the type of problems of our interest, because they provide a good approximation of the true Pareto front, although they are unable to produce a good spread of the solutions. (refer to Section 5.3 in page 96 for further details).

- We have introduced a local search mechanism based on rough sets theory in order to intensify the search around the solutions obtained by the previous phase that is able to find good solutions near the Pareto front. We did an hybridization with both approaches: 1) differential evolution and 2) surrogate based MOEA, in order to improve the local exploration around the nondominated solutions found so far. If the search engine adopted to produce a coarse-grained approximation of the Pareto front is efficient (as in our case), then a good approximation of the true Pareto front can be achieved with a low computational cost. (refer to Section 6.1.1 in page 107 for further details).

- We have proposed an alternative approach for the ϵ-dominance (which we call paϵ-dominance). In our proposed scheme, we considered different ϵ-dominance regions depending on the geometrical characteristics of the Pareto-optimal front. We took advantage of the positive aspects of ϵ-dominance, while addressing some of its limitations. On the one hand, paϵ-dominance finds a higher number of nondominated solutions because the size of the boxes are adjusted to the shape of the Pareto front. Also, these solutions are better uniformly distributed along the Pareto front because the new grid balances the size of the boxes being more precise in those areas of the objective function space in which more solutions are needed. (refer to Chapter 7 in page 123 for further details).

9.2 Conclusions

The high number of fitness evaluations performed by EAs are often computationally expensive, time-consuming and problematic in many real-world applications. Thus, our main purpose was to develop different techniques aimed to improve efficiency, trying to reduce the number of fitness function evaluations performed by a MOEA. We observed that the use of a hybrid algorithm was a suitable way to obtain a good approximation of the unconstrained and constrained MOP problems with a few number of fitness evaluations. We used an algorithm with a fast convergence rate to find quickly promising solutions. After that, a local optimizer was used to exploit those good solutions to accelerate the convergence and finally get to the true Pareto front of the multi-objective problem.

The following conclusions were obtained during the development process of the approach, its experimental validation, as well as its application in multi-objective optimization:

- We have presented a new hybrid multi-objective optimization algorithm for unconstrained and constrained MOP problems based on the use of a fast differential evolution and a local search engine based on rough sets. The proposed approach was found to provide very competitive results with respect to other previous works (NSGA-II and SPEA2) in a variety of test problems, in spite of the fact that it performed only 5,000 and 10,000 fitness function evaluations for unconstrained and constrained MOP problems, respectively. Our comparison of results indicated that our approach clearly outperforms the NSGA-II and SPEA2, which are two of the most competitive MOEAs known to date.

- The hybridization of a fast MOEA with a local search engine is a suitable and powerful tool. If the search engine adopted to produce a coarse-grained approximation of the Pareto front is efficient (as in our case), then a good approximation of the true Pareto front can be finally achieved with an small extra additional cost by using the rough sets based local optimizer.

- We used the surrogate models in combination with a local search procedure based on rough sets to solve multi-objective problems. These results seem to indicate that our hybrid approach SVM+RS can be a viable alternative for real-world applications in which each evaluation of the fitness function is very expensive (computationally speaking). In such applications, we can afford sacrificing a good distribution of solutions for the sake of obtaining a reasonably good approximation of the Pareto front with a low number of evaluations.

- In order to assess the performance of our proposed paϵ-dominance, we solved five test problems with different geometrical characteristics and used three standard metrics designed to measure diversity properties and one more measure related to the number of points found. In all cases, paϵ-dominance has been found to be more efficient in getting a higher number of nondominated solutions with a better spread. Thus, we conclude that paϵ-dominance is an advantageous alternative to ϵ-dominance, particularly when the Pareto front has geometrical characteristics (such as convex, concave, connected, or/and disconnected) that cause difficulties to ϵ-dominance.

9.3 Future Work

As part of our future work, we intend to improve the performance of the differential evolution algorithm adopted, by exploring alternative differential evolution models and operators. Additionally, we are also interested in coupling the local search mechanisms to different search engines.

We will intend to improve the performance of the surrogate-based algorithm adopted, refining the interaction mechanism between the surrogate method and the MOEA, such that the interleaving of these two approaches maximizes performance. We are also interested in including another scheme to retain solutions in the training set, helping the surrogate models to get a better approximation of the real functions.

We are interested in validating our proposed scheme with other MOPs (including more real-world applications). We also want to refine the hybridization scheme, such that a larger reduction in the total number of evaluations can be achieved.

With respect to the $pa\epsilon$-dominance method, we plan to generalize our proposal, so that we can drop our symmetry hypothesis assumed in the curves of the form $x^p + y^p = 1$, considering $x^p + y^q = 1$ or $x^p + y^q + z^r = 1$ for two and three objective problems respectively. This would certainly be more unstable than the current proposal, but we believe that such instability can be controlled using a different way of determining the values of p, and q (and r, if dealing with a three-objective problem), for a given Pareto front. Disconnected Pareto front also require a more in-depth analysis, since they deserve a special treatment when using relaxed forms of Pareto dominance such as ϵ-dominance or our proposed scheme.

9.4 Publications

The different papers that have been derived from this thesis are the following:

1. **Luis V. Santana-Quintero**, Carlos A. Coello Coello, Alfredo G. Hernández-Díaz and Jesús Moisés Osorio Velázquez, Surrogate-based Multi-Objective Particle Swarm Optimization, *in Proceedings of the 2008 IEEE Swarm Intelligence Symposium (SIS 2008)*, IEEE Press, St. Louis, Missouri, USA, September 2008.

2. **Luis V. Santana-Quintero**, Carlos Coello Coello and Alfredo G. Hernández-Díaz, Hybridizing Surrogate Techniques, Rough Sets and Evolutionary Algorithms to Efficiently Solve Multi-Objective Optimization Problems, in Maarten Keijzer et al. (editors), *Genetic and Evolutionary Computation Conference (GECCO'2008)*, pp. 763-764, ACM Press, Atlanta, Georgia, USA, July 2008.

3. Ricardo Landa-Becerra, **Luis V. Santana-Quintero** and Carlos Coello Coello, Knowledge Incorporation in Multi-Objective Evolutionary Algorithms, in Ashish Ghosh et al. (editors), *Multi-objective Evolutionary Algorithms for Knowledge Discovery from Data Bases*, pp. 23–46, Springer, Berlin, 2008.

4. Alfredo G. Hernández-Díaz, **Luis V. Santana-Quintero**, Carlos A. Coello Coello, Rafael Caballero, and Julián Molina, Rough Sets Theory for Muli-Objective Optimization Problems, in Carlos Cotta, Simeon Reich, Antoni Ligeza and Robert Schaefer (editors), *Knowledge-Driven Computing*, pp. 81 – 98, ISBN 978-3-540-77474-7 Springer, 2008.

5. Alfredo G. Hernández-Díaz, **Luis V. Santana-Quintero**, Carlos A. Coello Coello and Julián Molina, Pareto-adaptive ϵ-dominance, *Evolutionary Computation*, Vol. 15, No. 4, pp. 493–517, Winter, MIT Press, 2007.

6. **Luis V. Santana-Quintero**, Víctor A. Serrano-Hernández, Carlos A. Coello Coello, Alfredo G. Hernández-Díaz and Julián Molina, Use of Radial Basis Functions and Rough Sets for Evolutionary Multi-Objective Optimization, *in Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)*, pp. 107–114, IEEE Press, Honolulu, Hawaii, USA, April 2007.

7. **Luis V. Santana-Quintero**, Noel Ramírez-Santiago, Carlos A. Coello Coello, Julián Molina Luque and Alfredo Garcia Hernández-Díaz, A New Proposal for Multiobjective Optimization using Particle Swarm Optimization and Rough Sets Theory, *Parallel Problem Solving from Nature (PPSN IX)*. 9th International Conference, Springer, pp. 483-492, LNCS Vol. 4193, Reykjavik, Iceland, September 2006

8. Alfredo G. Hernández-Díaz, **Luis V. Santana-Quintero**, Carlos Coello Coello, Rafael Caballero and Julián Molina, A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory, in Maarten Keijzer et al. (editors), *Genetic and Evolutionary Computation Conference (GECCO'2006)*, pp. 675-682, Vol. 1, ACM Press, Seattle, Washington, USA, July 2006, ISBN 1-59593-186-4.

9. **Luis Vicente Santana-Quintero** and Carlos A. Coello Coello, An Algorithm Based on Differential Evolution for Multi-Objective Problems, *International Journal of Computational Intelligence Research*, Vol. 1, No. 2, pp. 151-169, 2005 ISSN 0973-1873.

10. **Luis Vicente Santana-Quintero** and Carlos A. Coello Coello, An Algorithm Based on Differential Evolution for MultiObjective Problems, *ANNIE Artificial Neural Networks in Engineering*, Vol. 15, pp. 211-220, ASME Press, St. Louis, 2005.

A

Test Functions Adopted: Definitions

Function	Objectives	Bounds	Characteristics
ZDT1	$f_1(\vec{x}) = x_1$ and $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	convex, connected
ZDT2	$f_1(\vec{x}) = x_1$ and $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	nonconvex, connected
ZDT3	$f_1(\vec{x}) = x_1$ and $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})} - (f_1/g(\vec{x}))\sin(10\pi f_1)$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	discontinuos
ZDT4	$f_1(\vec{x}) = x_1$ $g(\vec{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$	n = 10 $0 \leq x_1 \leq 1$, $-5 \leq x_i \leq 5$, $i = 2, 3 \ldots, 10$	nonconvex, exists 21^9 local Pareto-optimal fronts
ZDT6	$f_1(\vec{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $g(\vec{x}) = 1 + 9[(\sum_{i=2}^{10} x_i)/9]$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$	n = 10 $0 \leq x_i \leq 1$, $i = 2, 3 \ldots, 10$	nonconvex
DTLZ1	$f_1(\vec{x}) = \frac{1}{2}x_1x_2(1 + g(\vec{x}))$ $f_2(\vec{x}) = \frac{1}{2}x_1(1 - x_2)(1 + g(\vec{x}))$ $f_3(\vec{x}) = \frac{1}{2}(1 - x_1)(1 + g(\vec{x}))$ $g(\vec{x}) = 100[5 + \sum_{i=3}^{n}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]$	n = 7 $0 \leq x_i \leq 1$ $i = 1, \ldots, 7$	linear Pareto front, contains $11^5 - 1$ local Pareto-optimal fronts
DTLZ2	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1 + g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1 + g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1)(1 + g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{n}(x_i - 0.5)^2$	n = 12 $0 \leq x_i \leq 1$ $i = 1, \ldots, 12$	convex, connected
DTLZ3	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1 + g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1 + g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1)(1 + g(\vec{x}))$ $g(\vec{x}) = 100[10 + \sum_{i=3}^{n}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]$	$0 \leq x_i \leq 1$ $i = 1, \ldots, 12$	nonconvex contains $3^{10} - 1$ local Pareto-optimal fronts
DTLZ4	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1^{\alpha})\cos(\frac{\pi}{2}x_2^{\alpha})(1 + g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1^{\alpha})\sin(\frac{\pi}{2}x_2^{\alpha})(1 + g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1^{\alpha})(1 + g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{12}(x_i - 0.5)^2$	n = 12; $\alpha = 100$ $0 \leq x_i \leq 1$ $i = 1, \ldots, 12$	convex, connected

Table A.1: Definition and description of each of the 9 unconstrained test problems adopted in this thesis.

Function	**Objectives**	**Bounds**
Deb11	$f_1(x_1) = x_1$ $f_2(x_1, x_2) = \frac{1}{x_1}\left(2 - e^{-\left(\frac{x_2-0.2}{0.004}\right)^2} - 0.8e^{-\left(\frac{x_2-0.6}{0.4}\right)^2}\right)$	$0.1 \leq x_i \leq 1$ $i = 1, 2$
Deb52	$f_1(x_1) = 1 - e^{-4x_1} \sin^4(10\pi x_1)$ $f_2(x_1, x_2) = g(x_2) * h(x_1)$, where $g(x_2) = 1 + x_2^2$ and $h(x_1) = \begin{cases} 1 - \left(\frac{f_1(x_1)}{g(x_2)}\right)^{10} & if\ f_1(x_1) \leq g(x_2) \\ 0 & \text{otherwise.} \end{cases}$	$0 \leq x_i \leq 1$ $i = 1, 2$
Kursawe	$f_1(x_1, x_2) = \sum_{i=1}^{2} -10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}$ $f_2(x_1, x_2) = \sum_{i=1}^{2} \left(\lvert x_i\rvert^{0.8} + 5\sin(x_i^3)\right)$	$-5 \leq x_i \leq 5$ $i = 1, 2, 3$

Table A.2: Objective functions and bounds of the decision variables for each of the test problems adopted for our experimental study in paϵ-dominance.

Function	Objectives and Constraints	Bounds	Characteristics
Binh (2) [12]	$f_1(x,y) = 4x^2 + 4y^2$, $f_2(x,y) = (x-5)^2 + (y-5)^2$ subject to: $0 \geq (x-5)^2 + y^2 - 25$ $0 \geq -(x-8)^2 - (y+3)^2 + 7.7$	$0 \leq x \leq 5$, $0 \leq y \leq 3$	connected, convex
Kita [85]	$\text{Max} f_1(x,y) = -x^2 + y$, $\text{Max} f_2(x,y) = \frac{1}{2}x + y + 1$ subject to: $0 \geq \frac{1}{6}x + y - \frac{13}{2}$, $0 \geq \frac{1}{2}x + y - \frac{15}{2}$, $0 \geq 5x + y - 30$	$0 \leq x \leq 7$, $0 \leq y \leq 7$	connected, concave
Osyckza [116]	$f_1(x,y) = x + y^2$, $f_2(x,y) = x^2 + y$ subject to: $0 \leq 12 - x - y$, $0 \leq x^2 + 10x - y^2 + 16y - 80$	$2 \leq x \leq 7$, $5 \leq y \leq 10$	disconnected, convex
Osyckza (2) [116]	$f_1(\vec{x}) = -25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2$ $+ (x_4-4)^2 + (x_5-1)^2$, $f_2(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$ subject to: $0 \leq x_1 + x_2 - 2$, $0 \leq 6 - x_1 - x_2$, $0 \leq 2 - x_2 + x_1$, $0 \leq 2 - x_1 + 3x_2$, $0 \leq 4 - (x_3-3)^2 - x_4$, $0 \leq (x_5-3)^2 + x_6 - 4$	$0 \leq x_1, x_2, x_6 \leq 10$, $1 \leq x_3, x_5 \leq 5$, $0 \leq x_4 \leq 6$	disconnected,
Srinivas [149]	$f_1(x,y) = (x-2)^2 + (y-1)^2 + 2$, $f_2(x,y) = 9x - (y-1)^2$ subject to: $0 \geq x^2 + y^2 - 225$, $0 \geq x - 3y + 10$	$-20 \leq x \leq 20$, $-20 \leq y \leq 20$	connected
Tanaka [155]	$f_1(x,y) = x$, $f_2(x,y) = y$ subject to: $0 \geq -x^2 - y^2 + 1 + 0.1 * \cos(16 \arctan \frac{x}{y})$ $\frac{1}{2} \geq (x-\frac{1}{2})^2 + (y-\frac{1}{2})^2$	$0 \leq x \leq \pi$, $0 \leq y \leq \pi$	disconnected
Welded beam [124]	$f_1(\vec{x}) = 1.10471h^2l + 0.04811tb(14.0+l)$, $f_2(\vec{x}) = \frac{4 \cdot F \cdot L^3}{Et^3b}$ subject to: $g_1(\vec{x}) \equiv \tau_{max} - \tau \geq 0$, $g_2(\vec{x}) \equiv \rho_{max} - \rho \geq 0$, $g_3(\vec{x}) \equiv b - h \geq 0$, $g_4(\vec{x}) \equiv 0.125 - h \geq 0$ $g_5(\vec{x}) \equiv P_c - F \geq 0$ where: $F = 6{,}000\ lb$ $\quad L = 14\ in$ $E = 30e6\ psi$ $\quad G = 12e6\ psi$ $\tau_{max} = 13{,}600\ psi$ $\quad \rho_{max} = 30{,}000\ psi$ $\alpha = \frac{1}{(3G \cdot t \cdot b^3)}$ $\quad I = \frac{1}{(12 \cdot t \cdot b^3)}$ $P_c = (4013\sqrt{EI\alpha}/L^2) \cdot (1 - (t/2L) \cdot \sqrt{EI/\alpha})$ $\rho = (6FL)/(bl^2)$ $J = 2 \cdot (0.707hl \cdot (l^2/12 + ((h+b)/2)^2)$ $R = \sqrt{l^2/4 + ((h+l)/2)^2}$ $M = F(L + l/2)$ $cost = l/2R$ $\tau'' = MR/J$ $\tau' = F/(\sqrt{2} + hl)$ $\tau = \sqrt{(\tau'^2 + 2\tau'\tau''cost + \tau''^2)}$	$0 \leq h, b \leq 2.0$, $0 \leq l, t \leq 10.0$	convex,

Table A.3: Definition and description of the constraint test problems adopted in this thesis.

B

Test Functions Adopted: Figures

The following figures present the real Pareto front for each function in Appendix A.

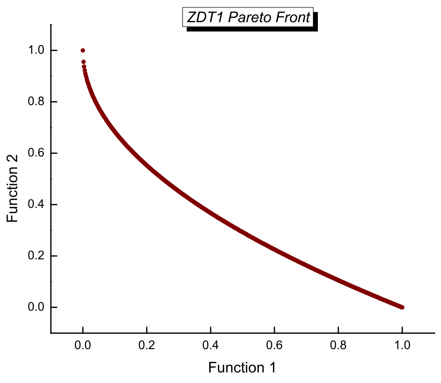


Figure B.1: ZDT1 Pareto front

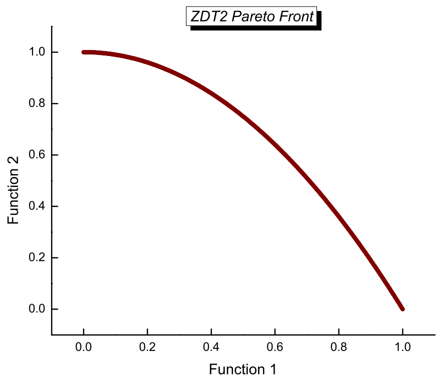


Figure B.2: ZDT2 Pareto front

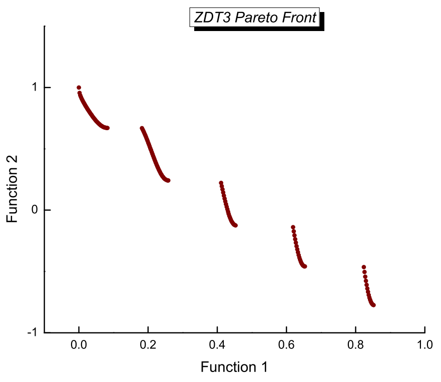


Figure B.3: ZDT3 Pareto front

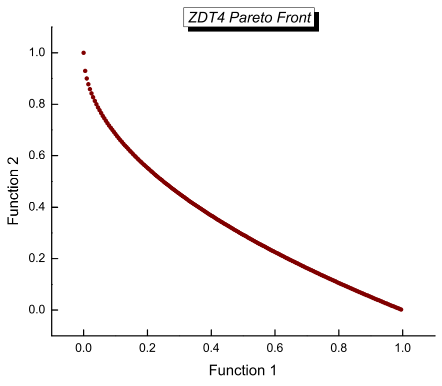


Figure B.4: ZDT4 Pareto front

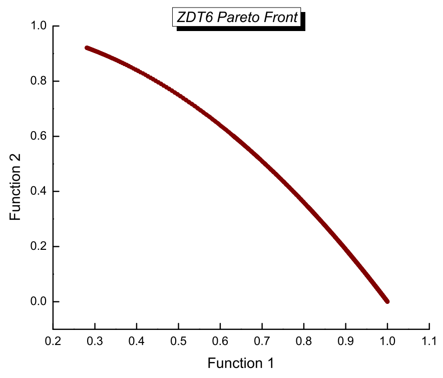


Figure B.5: ZDT6 Pareto front

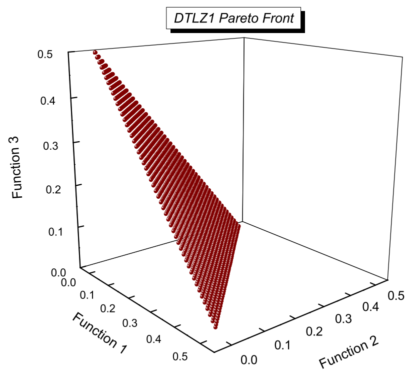


Figure B.6: DTLZ1 Pareto front

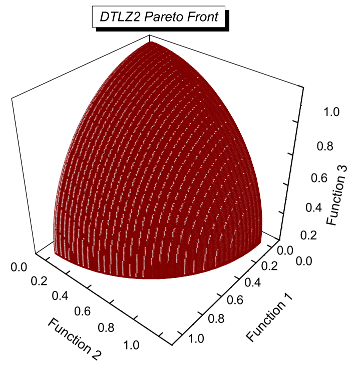


Figure B.7: DTLZ2 Pareto front

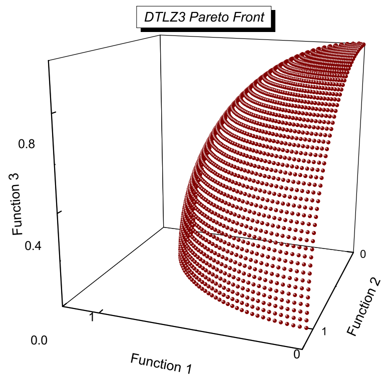


Figure B.8: DTLZ3 Pareto front

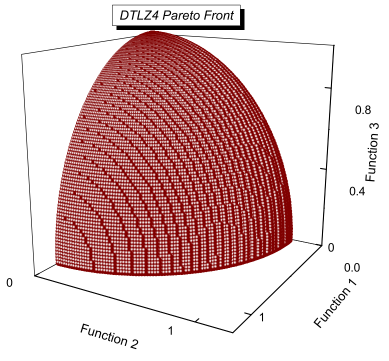


Figure B.9: DTLZ4 Pareto front

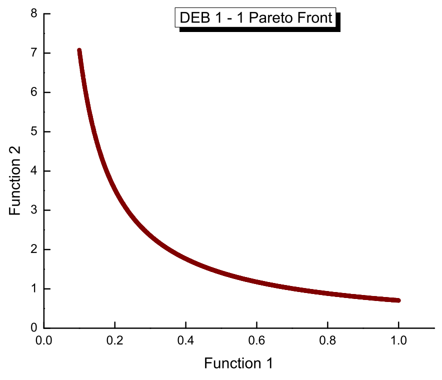


Figure B.10: Deb 1 - 1 Pareto front

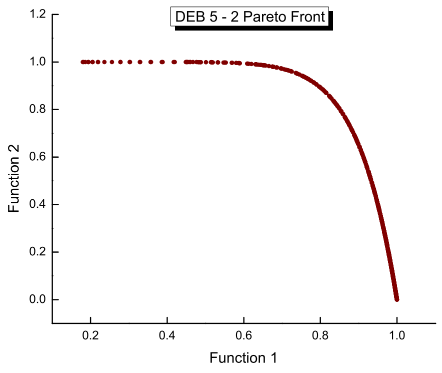


Figure B.11: Deb 5 - 2 Pareto front

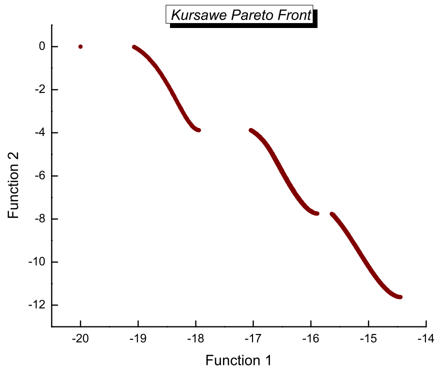


Figure B.12: Kursawe Pareto front

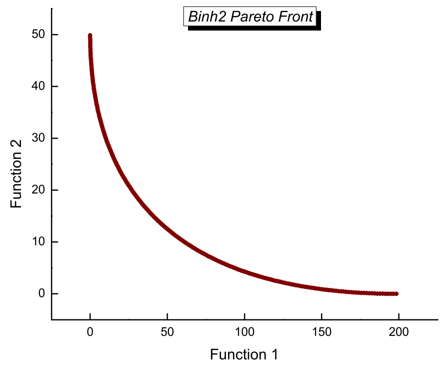


Figure B.13: Binh (2) Pareto front

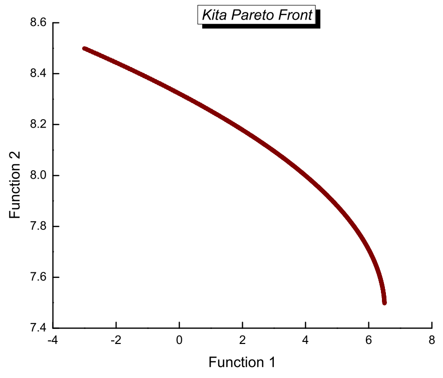


Figure B.14: Kita Pareto front

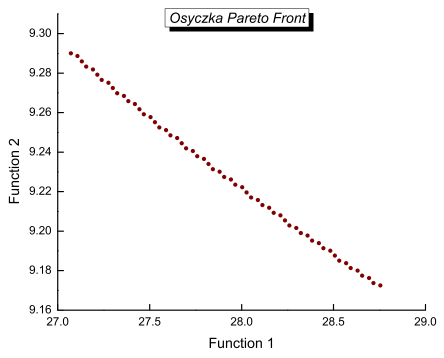


Figure B.15: Osyczka Pareto front

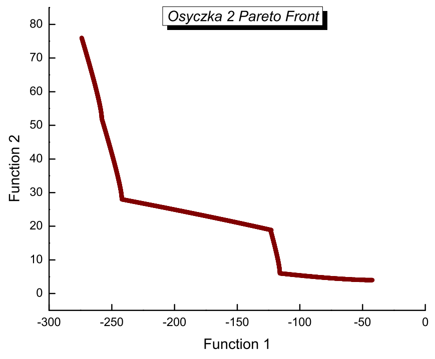


Figure B.16: Osyczka (2) Pareto front

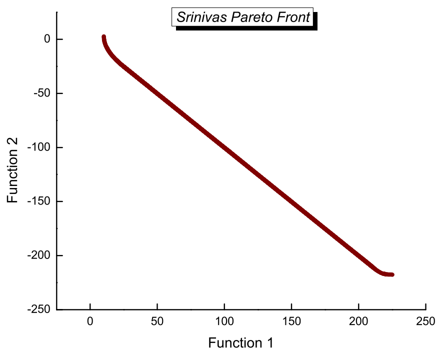


Figure B.17: Srinivas Pareto front

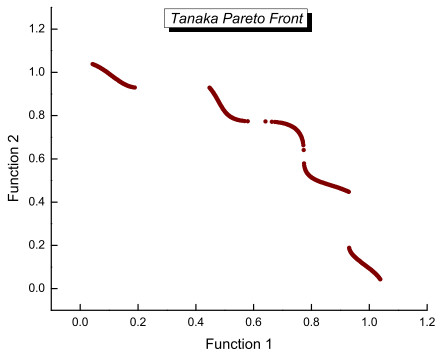


Figure B.18: Tanaka Pareto front

Bibliography

[1] ABBASS, H. A. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 1, IEEE Service Center, pp. 831–836.

[2] ABBASS, H. A., AND SARKER, R. The Pareto Differential Evolution Algorithm. *International Journal on Artificial Intelligence Tools 11*, 4 (2002), 531–552.

[3] ABIDO, M. Two-Level of Nondominated Solutions Approach to Multiobjective Particle Swarm Optimization. In *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)* (London, UK, July 2007), D. Thierens, Ed., vol. 1, ACM Press, pp. 726–733.

[4] ALVAREZ-BENITEZ, J. E., EVERSON, R. M., AND FIELDSEND, J. E. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005* (Guanajuato, México, March 2005), C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., Springer. LNCS Vol. 3410, pp. 459–473.

[5] BABU, B., AND JEHAN, M. M. L. Differential Evolution for Multi-Objective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)* (Canberra, Australia, December 2003), vol. 4, IEEE Press, pp. 2696–2703.

[6] Bäck, T., Fogel, D. B., and Michalewicz, Z., Eds. *Handbook of Evolutionary Computation.* Institute of Physics Publishing and Oxford University Press, 1997.

[7] Bäck, T. A. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, 1996.

[8] Baker, J. E. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application* (Mahwah, NJ, USA, 1987), Lawrence Erlbaum Associates, Inc., pp. 14–21.

[9] Bartz-Beielstein, T., Limbourg, P., Parsopoulos, K. E., Vrahatis, M. N., Mehnen, J., and Schmitt, K. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)* (Canberra, Australia, December 2003), vol. 3, IEEE Press, pp. 1780–1787.

[10] Baumgartner, U., Magele, C., and Renhart, W. Pareto Optimality and Particle Swarm Optimization. *IEEE Transactions on Magnetics 40*, 2 (March 2004), 1172–1175.

[11] Bhattacharya, M., and Lu, G. A dynamic approximate fitness based hybrid ea for optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation* (2003), pp. 1879–1886.

[12] Binh, T. T., and Korn, U. MOBES: A multiobjective evolution strategy for constrained optimization problems. In *The Third International Conference on Genetic Algorithms (Mendel 97)* (Brno, Czech Republic, 1997), pp. 176–182.

[13] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, UK, 1995.

[14] Bramanti, A., Barba, P. D., Farina, M., and Savini, A. Combining response surfaces and evolutionary strategies for multiobjective pareto-optimization in electromagnetics. In *International Journal of Applied Electromagnetics and Mechanics* (2001), vol. 15, pp. 231 – 236.

[15] Bueche, D., Schraudolph, N., and Koumoutsakos, P. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Trans. on Systems, Man, and Cybernetics: Part C* (2004).

[16] Cai, Z., Gong, W., and Huang, Y. A novel differential evolution algorithm based on ϵ-domination and orthogonal design method for multiobjective problem. In *Evolutionary Multi-Criterion Optimization. Fourth International Conference, EMO 2007* (Matsushima, Japan, March 2007), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., Springer. Lecture Notes in Computer Science Vol. 4403, pp. 286–301.

[17] Chafekar, D., Shi, L., Rasheed, K., and Xuan, J. Multi-objective ga optimization using reduced models. *IEEE Trans. on Systems, Man, and Cybernetics: Part C* (2004).

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

[19] Coello Coello, C. A., Christiansen, A. D., and Hernández Aguirre, A. Using a New GA-Based Multiobjective Optimization Technique for the Design of Robot Arms. *Robotica 16*, 4 (July–August 1998), 401–414.

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

[21] Coello Coello, C. A., and Lamont, G. B., Eds. *Applications of Multi-Objective Evolutionary Algorithms.* World Scientific, Singapore, 2004. ISBN 981-256-106-4.

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

[23] Coello Coello, C. A., and Salazar Lechuga, M. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 2, IEEE Service Center, pp. 1051–1056.

[24] Coello Coello, C. A., and Toscano Pulido, G. A Micro-Genetic Algorithm for Multiobjective Optimization. In *First International Conference on Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, Eds. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001, pp. 126–140.

[25] Coello Coello, C. A., Toscano Pulido, G., and Salazar Lechuga, M. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation 8*, 3 (June 2004), 256–279.

[26] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[27] Cohon, J., and Marks, D. A review and evaluation of multiobjective programming techniques. *Water Resources Research 11*, 2 (1975), 208–220.

[28] Dantzig, G. B. *Linear Programming and Extensions.* Princeton University Press, New Jersey, 1963.

[29] Das, I., and Dennis, J. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization 14*, 1 (1997), 63–69.

[30] Dawkins, R. *The Selfish Gene.* Oxford University Press, September 1990.

[31] de Castro, L. N., and Timmis, J. *Artificial Immune Systems: A new Computational Intelligence Approach.* Springer Verlag, 2002.

[32] De Jong, K. *An Analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[33] Deb, K. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Tech. Rep. CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

[34] Deb, K. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation 7*, 3 (Fall 1999), 205–230.

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

[36] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference* (Paris, France, 2000), pp. 849–858.

[37] Deb, K., Mohan, M., and Mishra, S. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003* (Faro, Portugal, April 2003), pp. 222–236.

[38] Deb, K., Mohan, M., and Mishra, S. Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation 13*, 4 (Winter 2005), 501–525.

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

[40] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 1, IEEE Service Center, pp. 825–830.

[41] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable Test Problems for Evolutionary Multiobjective Optimization. In *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. Springer, USA, 2005, pp. 105–145.

[42] Dhingra, A., and Lee, B. A genetic algorithm approach to single and multiobjective structural optimization with discrete-continuous variables. *International Journal for Numerical Methods in Engineering 37* (1994), 4059–4080.

[43] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B 26* (1996), 29–41.

[44] Eberhart, R. C., and Shi, Y. Comparison between Genetic Algorithms and Particle Swarm Optimization. In *Proceedings of the Seventh Annual Conference on Evolutionary Programming* (March 1998), V. W. Porto, N. Saravanan, D. Waagen, and A. Eibe, Eds., Springer-Verlag, pp. 611–619.

[45] Emmerich, M., Giotis, A., Özdenir, M., Bäck, T., and Giannakoglou, K. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature* (2002), no. 2439 in Lecture Notes in Computer Science, Springer, pp. 371–380.

[46] Emmerich, M. T., Giannakoglou, K. C., and Naujoks, B. Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation 10*, 4 (August 2006), 421–439.

[47] Evans, G. W. An overview of techniques for solving multiobjective mathematical programs. *Management Science 30*, 11 (November 1984), 1268 – 1282.

[48] Farina, M. A neural network based generalized response surface multiobjective evolutionary algorithms. In *Congress on Evolutionary Computation* (2002), IEEE Press, pp. 956–961.

[49] Fieldsend, J. E., and Singh, S. A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence* (Birmingham, UK, September 2002), pp. 37–44.

[50] Fishburn, P. C. Lexicographic orders, utilities and decision rules: A survey. *Management Science 20*, 11 (1974), 1442 – 1471.

[51] Fleming, P. J., and Pashkevich, A. P. Computer aided control system design using a multiobjective optimization approach. In *Proc. of the IEE Control '85 Conference* (1985), pp. 174 – 179.

[52] Fletcher, R. *Practical Methods of Optimization.* John Wiley and Sons, New York, 1989.

[53] Fogel, D. B. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks 5*, 1 (January 1994), 3–14.

[54] Fonseca, C. M., and Fleming, P. J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), S. Forrest, Ed., University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, pp. 416–423.

[55] Fu, J., Liu, Q., Zhou, X., Xiang, K., and Zeng, Z. An adaptive variable strategy pareto differential evolution algorithm for multi-objective optimization. In *2008 IEEE Congress on Evolutionary Computation (CEC'2008)* (Hong Kong, June 2008), IEEE Press, pp. 648–652.

[56] Giunta, A., and Watson, L. A comparison of approximation modeling techniques: Polynomial versus interpolating models. Tech. Rep. 98-4758, AIAA, 1998.

[57] Glover, F., and Laguna, M. *Tabu Search.* Kluwer Academic Publishers, Norwell, Massachusetts, 1998.

[58] Goel, T., Vaidyanathan, R., Haftka, R., Shyy, W., Queipo, N., and Tucker, K. Response surface approximation of pareto optimal front in multiobjective optimization. Tech. Rep. 2004-4501, AIAA, 2004.

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

[60] Goldberg, D. E., and Deb, K. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms* (1991), Morgan Kaufmann, pp. 69–93.

[61] Goldberg, D. E., Deb, K., and Korb, B. Don't worry. be messy. In *Proceedings of the Fourth International Conference on Genetic Algorithms* (1991), R. K. Belew and L. B. Booker, Eds., Morgan Kaufmann, pp. 24–30.

[62] Grefenstette, J. J. Genesis: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines* (1984), pp. 161 – 165.

[63] Haimes, Y. Y., Lasdon, L. S., and Wismer, D. A. On a bicriterion formulation of the problems integrated system identification and system optimization. *IEEE Transactions on Systems, Man. and Cybernetics 1*, 3 (July 1971), 296 – 297.

[64] Hardy, R. L. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. res 76* (1971), 1905–1915.

[65] Hedayat, A. S., Sloane, N. J. A., and Stufken, J. *Orthogonal Arrays: Theory and Applications*. Springer - Verlag, New York, 1999.

[66] Hernández-Díaz, A. G., Santana-Quintero, L. V., Coello Coello, C., Caballero, R., and Molina, J. A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory. In *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)* (Seattle, Washington, USA, July 2006), M. K. et al., Ed., vol. 1, ACM Press. ISBN 1-59593-186-4, pp. 675–682.

[67] Hernández-Díaz, A. G., Santana-Quintero, L. V., Coello Coello, C. A., Caballero, R., , and Molina, J. Rough Sets Theory for Multi-Objective Optimization Problems. In *Knowledge-Driven Computing*, C. Cotta, S. Reich, R. Schaefer, and A. Ligęza, Eds. Springer-Verlag, Berlin, 2008, pp. 81–98. ISBN 978-3-540-77474-7.

[68] Hernández-Díaz, A. G., Santana-Quintero, L. V., Coello Coello, C. A., and Molina, J. Pareto-adaptive ϵ-dominance. *Evolutionary Computation 15*, 4 (Winter 2007), 493–517.

[69] Holland, J. H. *Evolutionstrategie: Optimierung technisher nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart, Alemania, 1973.

[70] Holland, J. H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[71] Hong, Y.-S., H.Lee, and Tahk, M.-J. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization 35*, 1 (2003), 91–102.

[72] Horn, J., Nafpliotis, N., and Goldberg, D. E. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* (Piscataway, New Jersey, June 1994), vol. 1, IEEE Service Center, pp. 82–87.

[73] Hüscken, M., Jin, Y., and Sendhoff, B. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal 9*, 1 (2005), 21–28.

[74] I., V., and A.J., K. Multiobjective Optimization using Surrogates. In *Proceedings of the 7th International Conference on Adaptive Computing in Design and Manufacture* (Holland, 2006), pp. 167 – 175.

[75] Ijiri, Y. *Management Goals and Accounting for Control.* North Holland Publishing Company, Amsterdam, 1965.

[76] Iorio, A. W., and Li, X. Solving rotated multi-objective optimization problems using differential evolution. In *AI 2004: Advances in Artificial Intelligence, Proceedings* (2004), Springer-Verlag, Lecture Notes in Artificial Intelligence Vol. 3339, pp. 861–872.

[77] Iorio, A. W., and Li, X. Incorporating Directional Information within a Differential Evolution Algorithm for Multi-objective Optimization. In *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)* (Seattle, Washington, USA, July 2006), M. K. et al., Ed., vol. 1, ACM Press. ISBN 1-59593-186-4, pp. 691–697.

[78] Jakob, W., Gorges-Shleuter, M., and Blume, C. Application of genetic algorithms to task planning and learning. In *Parallel*

Problem Solving from Nature—PPSN 2 (Amsterdam, North-Holland, 1992), R. Männer and B. Manderick, Eds., pp. 291–300.

[79] Jensen, M. T. Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *IEEE Transactions on Evolutionary Computation 7*, 5 (October 2003), 503–515.

[80] Jin, R., Chen, W., and Simpson, T. Comparative studies of metamodeling techniques under miltiple modeling criteria. Tech. Rep. 2000-4801, AIAA, 2000.

[81] Jin, Y. Fitness approximation in evolutionary computation - bibliography. http://www.soft-computing.de/amec.html, July 2005.

[82] Karakasis, M. K., and Giannakoglou, K. C. Metamodel-Assisted Multi-Objective Evolutionary Optimization. In *EUROGEN 2005. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems* (Munich, Germany, 2005), R. Schilling, W. Haase, J. Periaux, H. Baier, and G. Bugeda, Eds.

[83] Karush, W. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[84] Kennedy, J., and Eberhart, R. C. *Swarm Intelligence*. Morgan Kaufmann Publishers, California, USA, 2001.

[85] Kita, H., Yabumoto, Y., Mori, N., and Nishikawa, Y. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In *Parallel Problem Solving from Nature—PPSN IV* (Berlin, Germany, September 1996), H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., Lecture Notes in Computer Science, Springer-Verlag, pp. 504–512.

[86] Knowles, J. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation 10*, 1 (January 2006), 50–66.

[87] Knowles, J. D. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization.* PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

[88] Knowles, J. D., and Corne, D. W. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation 8*, 2 (2000), 149–172.

[89] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[90] Kuhn, H. W., and Tucker, A. W. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium* (1951), J. Neyman, Ed., University of California Press, Berkeley, pp. 481 – 492.

[91] Kukkonen, S., and Lampinen, J. An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints. In *Parallel Problem Solving from Nature - PPSN VIII* (Birmingham, UK, September 2004), Springer-Verlag. Lecture Notes in Computer Science Vol. 3242, pp. 752–761.

[92] Kukkonen, S., and Lampinen, J. GDE3: The third Evolution Step of Generalized Differential Evolution. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)* (Edinburgh, Scotland, September 2005), vol. 1, IEEE Service Center, pp. 443–450.

[93] Kung, H., Luccio, F., and Preparata, F. On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery 22*, 4 (1975), 469–476.

[94] Kursawe, F. A Variant of Evolution Strategies for Vector Optimization. In *Parallel Problem Solving from Nature. 1st Workshop, PPSN I* (Berlin, Germany, October 1991), H. P. Schwefel and R. Männer, Eds., vol. 496 of *Lecture Notes in Computer Science Vol. 496*, Springer-Verlag, pp. 193–197.

[95] Laguna, M., and Martí, R. *Scatter Search: Methodology and Implementations in C.* Kluwer Academic Publishers, 2003.

[96] LAND, A. H., AND DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica 28*, 3 (1960), 497–520.

[97] LANDA-BECERRA, R., SANTANA-QUINTERO, L. V., AND COELLO COELLO, C. A. Knowledge Incorporation in Multi-Objective Evolutionary Algorithms. In *Multi-objective Evolutionary Algorithms for Knowledge Discovery from Data Bases*, A. Ghosh, S. Dehuri, and S. Ghosh, Eds. Springer, Berlin, 2008, pp. 23–46.

[98] LAUMANNS, M., THIELE, L., DEB, K., AND ZITZLER, E. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation 10*, 3 (Fall 2002), 263–282.

[99] LIN, T. Special issue on rough sets. *Journal of the Intelligent Automation and Soft Computing 2/2* (1996).

[100] LUENBERGER, D. G. *Linear and Nonlinear Programming*, second ed. Springer, 1984.

[101] M. MAXFIELD, A. C., AND FOGEL, L., Eds. *Artificial Intelligence through a Simulation of Evolution* (Washington D.C., 1965), Biophysics and Cybernetics Systems: Proceedings of the Second Cybernetics Sciences, Spartan Books.

[102] MADAVAN, N. K. Multiobjective Optimization Using a Pareto Differential Evolution Approach. In *Congress on Evolutionary Computation (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 2, IEEE Service Center, pp. 1145–1150.

[103] MAHFOUF, M., CHEN, M.-Y., AND LINKENS, D. A. Adaptive Weighted Particle Swarm Optimisation for Multi-objective Optimal Design of Alloy Steels. In *Parallel Problem Solving from Nature - PPSN VIII* (Birmingham, UK, September 2004), Springer-Verlag. LNCS Vol. 3242, pp. 762–771.

[104] MCKAY, M., BECKMAN, R., AND CONOVER, W. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[105] Mezura-Montes, E., and Coello, C. A. C. A Simple Evolution Strategy to Solve Constrained Optimization Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)* (Heidelberg, Germany, July 2003), E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, U.-M. O. Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, and J. Miller, Eds., Chicago, Illinois, Springer Verlag, pp. 640–641. Lecture Notes in Computer Science Vol. 2723.

[106] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, third ed. Springer-Verlag, 1996.

[107] Michalewicz, Z., and Fogel, D. B. *How to Solve It: Modern Heuristics*, second ed. Springer, 2004.

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

[109] Monarchi, D., Kisiel, C. C., and Duckstein, L. Interactive multiobjective programming in water resources: A case study. *Water Resources Research 9*, 4 (August 1973), 837 – 850.

[110] Moore, J., and Chapman, R. Application of Particle Swarm to Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University. (Unpublished manuscript), 1999.

[111] Morse, J. N. Reducing the size of the nondominated set: Pruning by clustering. *Computers and Operations Research 7*, 1-2 (1980), 55–66.

[112] Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts. Tech. Rep. C3P Report 826, Caltech, Pasadena, California, 1989.

[113] Nain, P. K. S., and Deb, K. Computationally effective search and optimization procedure using coarse to fine approximation. In *Congress on Evolutionary Computation* (2003), pp. 2081–2088.

[114] Ong, Y., Nair, P., and Keane, A. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal 41*, 4 (2003), 687–696.

[115] Ong, Y. S., Nair, P. B., Keane, A. J., and Wong, K. W. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In *Knowledge Incorporation in Evolutionary Computation*, Y. Jin, Ed., Studies in Fuzziness and Soft Computing. Springer, 2004, pp. 307–332.

[116] Osyczka, A., and Kundu, S. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization 10* (1995), 94–99.

[117] Parsopoulos, K., Taoulis, D., Pavlidis, N., Plagianakos, V., and Vrahatis, M. Vector Evaluated Differential Evolution for Multiobjective Optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)* (Portland, Oregon, USA, June 2004), vol. 1, IEEE Service Center, pp. 204–211.

[118] Parsopoulos, K., Tasoulis, D., and Vrahatis, M. Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)* (Innsbruck, Austria, February 2004), vol. 2, ACTA Press, pp. 823–828.

[119] Pawlak, Z. Rough sets. *International Journal of Computer and Information Sciences 11*, 1 (Summer 1982), 341–356.

[120] Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.

[121] Pierret, S. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *ASME Journal of Turbomachinery 121*, 3 (1999), 326–332.

[122] Price, K., and Storn, R. Differential evolution web site. http://www.icsi.berkeley.edu/storn/code.html, 2004.

[123] Purshouse, R. C., and Fleming, P. J. Evolutionary Multi-Objective Optimisation: An Exploratory Analysis. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)* (Canberra, Australia, December 2003), vol. 3, IEEE Press, pp. 2066–2073.

[124] Ragsdell, K. M., and Phillips, D. T. Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry Series B B*, 98 (1975), 1021–1025.

[125] Rao, S. S. *Engineering Optimization: Theory and Practice*, third ed. John Wiley & Sons, 1996.

[126] Rasheed, K., Ni, X., and Vattam, S. Comparison of methods for developing dynamic reduced models for design optimization. *Soft Computing Journal* (2003).

[127] Ratle, A. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Parallel Problem Solving from Nature* (1998), A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds., vol. V, pp. 87–96.

[128] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. *Engineering Optimization: Methods and Applications*. John Wiley & Sons, 1983.

[129] Reyes Sierra, M., and Coello Coello, C. A. Fitness Inheritance in Multiobjective Particle Swarm Optimization. In *2005 IEEE Swarm Intelligence Symposium (SIS'05)* (Pasadena, California, June 2005), IEEE Press, pp. 146–123.

[130] Reyes Sierra, M., and Coello Coello, C. A. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005* (Guanajuato, México, March 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., Springer. Lecture Notes in Computer Science Vol. 3410, pp. 505–519.

[131] Reyes-Sierra, M., and Coello Coello, C. A. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research 2*, 3 (2006), 287–308.

[132] Reynolds, R. G., Michalewicz, Z., and Cavaretta, M. Using cultural algorithms for constraint handling in GENOCOP. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. MIT Press, Cambridge, Massachusetts, 1995, pp. 298–305.

[133] Robič, T., and Filipič, B. DEMO: Differential Evolution for Multiobjective Optimization. In *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005* (Guanajuato, México, March 2005), C. A. C. Coello, A. Hernández, and E. Zitzler, Eds., Springer. Lecture Notes in Computer Science Vol. 3410, pp. 520–533.

[134] Rudolph, G., and Agapie, A. Convergences Properties of Some Multi-Objective Evolutionary Algorithms. In *Congress on Evolutionary Computation (CEC 2000)* (Piscataway, NJ, 2000), A. Zalzala and R. Eberhart, Eds., IEEE Press, Vol. 2, pp. 1010–1016.

[135] Sacks, J., Welch, W., Mitchell, T., and Wynn, H. Design and analysis of computer experiments (with discussion). In *Statistical Science* (1989), vol. 4, pp. 409 – 435.

[136] Santana-Quintero, L. V., and Coello, C. A. C. An Algorithm Based on Differential Evolution for Multiobjective Problems. In *Smart Engineering System Design: Neural Networks, Evolutionary Programming and Artificial Life* (St. Louis Missouri, USA, November 2005), C. H. Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, Eds., vol. 15, pp. 211–220.

[137] Santana Quintero, L. V., Coello Coello, C., Hernández-Díaz, A. G., and Osorio Velázquez, J. M. Use of Particle Swarm to accelerate convergence in a Surrogate-based algorithm to solve Multi-objective Optimization Problems. In *IEEE Swarm Intelligence Symposium 2008* (St. Louis, Missouri, USA, September 2008), IEEE Press.

[138] Santana-Quintero, L. V., and Coello Coello, C. A. An Algorithm Based on Differential Evolution for Multi-Objective Problems. *International Journal of Computational Intelligence Research 1*, 2 (2005), 151–169.

[139] Santana-Quintero, L. V., Coello Coello, C. A., and Hernández-Díaz, A. G. Hybridizing Surrogate Techniques, Rough Sets and Evolutionary Algorithms to Efficiently Solve Multi-Objective Optimization Problems. In *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)* (Atlanta, USA, July 2008), ACM Press, pp. 763–764. ISBN 978-1-60558-131-6.

[140] Santana-Quintero, L. V., Ramírez-Santiago, N., Coello Coello, C. A., Molina Luque, J., and Hernández-Díaz, A. G. A New Proposal for Multiobjective Optimization Using Particle Swarm Optimization and Rough Sets Theory. In *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds. Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland, September 2006, pp. 483–492.

[141] Santana-Quintero, L. V., Serrano-Hernandez, V. A., Coello, C. A. C., Hernández-Díaz, A. G., and Molina, J. Use of Radial Basis Functions and Rough Sets for Evolutionary Multi-Objective Optimization. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM'2007)* (Honolulu, Hawaii, USA, April 2007), IEEE Press, pp. 107–114.

[142] Schaffer, J. D. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms* (1985), Lawrence Erlbaum, pp. 93–100.

[143] Schaffer, J. D., and Grefenstette, J. J. Multiobjective Learning via Genetic Algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)* (Los Angeles, California, 1985), AAAI, pp. 593–595.

[144] Schoenauer, K. A. M. Surrogate deterministic mutation. In *Artificial Evolution'01* (2002), Springer, pp. 103–115.

[145] Schoenauer, M., and Michalewicz, Z. Evolutionary Computation at the Edge of Feasibility. In *Proceedings of the Fourth Conference*

on Parallel Problem Solving from Nature (PPSN IV) (Heidelberg, Germany, September 1996), H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., Berlin, Germany, Springer-Verlag, pp. 245–254.

[146] Schott, J. R. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[147] Smith, A. E., and Coit, D. W. Constraint Handling Techniques—Penalty Functions. In *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford University Press and Institute of Physics Publishing, 1997, ch. C 5.2.

[148] Smith, R. E., Dike, B. A., and Stegmann, S. A. Fitness Inheritance in Genetic Algorithms. In *SAC '95: Proceedings of the 1995 ACM Symposium on Applied Computing* (Nashville, Tennessee, USA, 1995), ACM Press, pp. 345–350.

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

[150] Steuer, R. E. *Multiple Criteria Optimization: Theory, Computation, and Applications.* John Wiley, New York, 1986.

[151] Storn, R., and Price, K. Differential evolution - a simple and efficient adaptative scheme for global optimization over continuous spaces. Technical Report TR-95- 12, International Computer Science, Berkeley, California, March 1995.

[152] Storn, R., and Price, K. Differential evolution - a fast and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11* (1997), 341–359.

[153] Syswerda, G. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms* (San Francisco, CA, USA, 1989), Morgan Kaufmann Publishers Inc., pp. 2–9.

[154] Tan, K. C., Khor, E. F., Lee, T. H., and Sathikannan, R. An Evolutionary Algorithm with Advanced Goal and Priority Specification for Multi-objective Optimization. *Journal of Artificial Intelligence Research 18* (2003), 183–215.

[155] Tanaka, M., Watanabe, H., Furukawa, Y., and Tanino, T. GA-Based Decision Support System for Multicriteria Optimization. In *Proceedings of the International Conference on Systems, Man, and Cybernetics* (Piscataway, NJ, 1995), vol. 2, IEEE, pp. 1556–1561.

[156] Toscano Pulido, G., and Coello Coello, C. A. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I* (Seattle, Washington, USA, June 2004), K. D. et al., Ed., Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, pp. 225–237.

[157] Tripathi, P. K., Bandyopadhyay, S., and Pal, S. K. Adaptive multi-objective particle swarm optimization algorithm. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)* (Singapore, September 2007), IEEE Press, pp. 2281–2288.

[158] Ulmer, H., Streicher, F., and Zell, A. Model-assisted steady-state evolution strategies. In *Proceedings of Genetic and Evolutionary Computation Conference* (2003), LNCS 2723, pp. 610–621.

[159] Ulmer, H., Streichert, F., and Zell, A. Evolution startegies assisted by gaussian processes with improved pre-selection criterion. In *Proceedings of IEEE Congress on Evolutionary Computation* (2003), pp. 692–699.

[160] Vapnik, V. *Statistical Learning Theory.* Wiley, 1998.

[161] Vapnik, V. N. *The Nature of Statistical Learning.* Springer, 1995.

[162] Veldhuizen, D. A. V. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[163] VELDHUIZEN, D. A. V., AND LAMONT, G. B. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Tech. Rep. TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[164] VELDHUIZEN, D. A. V., AND LAMONT, G. B. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation* (Piscataway, New Jersey, July 2000), vol. 1, IEEE Service Center, pp. 204–211.

[165] WETZEL, A. *Evaluation of Effectiveness of Genetic Algorithms in Combinatorial*. PhD thesis, University of Pittsburgh, 1983.

[166] WILLIAMS, C. K. I., AND RASMUSSEN, C. E. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo., Eds. MIT Press, 1996.

[167] WILSON, B., CAPPELLERI, D. J., SIMPSON, T. W., AND FRECKER, M. I. Efficient pareto frontier exploration using surrogate approximations. In *Symposium on Multidisciplinary Analysis and Optimization* (Long Beach, CA,, September 2000).

[168] WOLPERT, D. H., AND MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1* (1997), 67–82.

[169] WON, K., AND RAY, T. Performance of kriging and cokriging based surrogate models within the unified framework for surrogate assisted optimization. In *Congress on Evolutionary Computation* (2004), IEEE, pp. 1577–1585.

[170] XUE, F., SANDERSON, A. C., AND GRAVES, R. J. Pareto-based Multi-Objective Differential Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)* (Canberra, Australia, December 2003), vol. 2, IEEE Press, pp. 862–869.

[171] ZADEH, L. Fuzzy sets. *Information and Control 8*, 1 (Fall 1965), 338–353.

[172] Zadeh, L. A. Optimality and nonscalar-valued performance criteria. *IEEE Transactions on Automatic Control AC-8*, 1 (1963), 59–60.

[173] Zitzler, E., Deb, K., and Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation 8*, 2 (Summer 2000), 173–195.

[174] Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems* (Athens, Greece, 2002), K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., pp. 95–100.

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

[176] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation 7*, 2 (Summer 2003), 117–132.