In: Optimization in Polymer Processing ISBN # 978-1-61122-818-2
Editor: A.Gaspar-Cunha, J. A.Covas, pp.1-33©2011 Nova Science Publishers, Inc.

Chapter 4

EXTENDING OPTIMIZATION ALGORITHMS TO COMPLEX ENGINEERING PROBLEMS

António Gaspar-Cunha*, José Ferreira*, José António Covas*, Carlos Fonseca**

* Institute for Polymers and Composites/I3N, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal

** Department of Informatics Engineering, University of Coimbra, Pólo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal and CEG-IST, Instituto Superior Técnico, Technical University of Lisbon, 1049-101 Lisboa, Portugal

Key words: Multidisciplinary design optimization, Engineering design, Multi-Objective Optimization, Evolutionary Algorithms

1 Introduction

Real engineering design and optimization problems are complex, multidisciplinary and difficult to manage within reasonable timings; in some cases, they can, at least to some extent, be mathematically described by sophisticated computational tools, which generally require significant resources. The momentous advances in some scientific and technological subjects (e.g., computational fluid dynamics, structural mechanics), coupled to the development of highly performing computing techniques (e.g., parallel and/or grid computing) and computer facilities, make it possible to progressively tackle more features of complicated problems [1]. Multidisciplinary Design Optimization, MDO, can be described as a technology, an environment, or a methodology for the design of complex integrated engineering structures, which combines different disciplines and takes into account in a synergistic manner the interaction between the various subsystems [2-4]. Examples of its practical application include the design and optimization of aircrafts, cars, building structures and manufacturing systems [1, 5-9].

In principle, a strategy like MDO could also be utilized for the optimization of polymer processing, as this encompasses several disciplines, such as fluid mechanics, heat transfer, polymer science, rheology, numerical methods, mechanical engineering and optimization techniques.

Several MDO strategies have been proposed in the literature [1-9]. A common characteristic of earlier methods is the utilization of approximation and decomposition techniques, in order to divide the problem into smaller, more amenable, blocks [1-3], that are subsequently solved separately, using simpler models. However, the unified view is lost and, most likely, the total cost of the design will be actually higher than that of solving the whole problem at once. More recently, integrated MDO approaches have been proposed [4-9], but a few seem to overlook the fact that real world problems contain multiple, often, conflicting objectives, that must be tackled simultaneously [10].

As seen in previous chapters, Evolutionary Algorithms, EAs, are particularly adequate to deal with the multi-objective nature of real problems, as they work with a population (of vectors, or solutions) rather than with a single solution point. This feature enables the generation of Pareto frontiers representing the trade-off between the objectives, while simultaneously providing a link to the decision variables [10-13]. Thus, the result of a Multi-Objective Evolutionary Algorithm, MOEA, is a set of solutions as near as possible to the Pareto optimal front [10]. Other multi-objective algorithms may be preferable for specific situations, such as Ant Colony Optimization, ACO [13, 14], Stochastic Local Search, SLS [15], among others.

In all cases, it will be necessary to provide information regarding the relative importance of every problem objective. This is usually accomplished by introducing in the optimization system the preferences of a Decision Maker, DM [16]. Depending on the decision making strategy adopted, such information can be introduced before, during, or after the optimization (see chapter 3 for more details) [11-13, 16]. Additionally, since in real applications small variations of the design variables, or of environmental parameters, may frequently occur, the performance of the prospective optimal solution(s) should be only slightly affected by them, i.e., the solutions must be robust [17-21]. Even though robustness is clearly an important aspect to consider during optimization, it is rarely included in traditional algorithms [19, 21].

One of the major difficulties of applying MOEAs to real engineering problems is the large number of evaluations of objective functions that are necessary to obtain an acceptable solution - typically, of the order of several thousands. Furthermore, these evaluations are often time-consuming, as they use large numerical codes generally based on computationally costly methods, such as finite-differences or finite-elements. Consequently, reducing the number of evaluations necessary to reach an acceptable solution is of major practical importance [22]. However, finding good approximate methods may be more difficult than anticipated, due to the existence of several objectives and to the possible interactions between them, different approaches having been pursued, often involving the hybridization of MOEAs with local search procedures, known as Memetic algorithms [22-27].

Finally, it is vital to define/control the dimension of the multi-objective problem to solve. As the number of individual objectives raises, so does the number of non-dominated solutions. In turn, the problem becomes much more difficult to solve (at least from the EAs' point of view), since a large quantity of the common solutions move from one generation to the next, reducing the selection pressure. Simultaneously, it turns out to be increasingly difficult to visualize the correlations between the different solutions. Whenever possible, one should consider ways of reducing the number of objectives, for example using approaches based on statistical techniques [28, 29].

In conclusion, the application of a multidisciplinary design optimization methodology to solve multi-objective engineering problems should entail optimization together with engineering and design tools. The former must encompass the selection of the relevant algorithm(s), a decision support system, an analysis of the robustness of the solution(s), the reduction of the number of evaluations required by the optimization routine (and thus of the computing times) and the reduction of the number of objectives. The present chapter presents and discusses a possible approach to solve problems of the polymer processing type by employing tools that are able to deal with multiple objectives, decision making and robustness of the solutions, among others.

2 The Methodology

2.1 Methodology Structure

Although the methodology proposed here does not possess the breadth of a complete MDO, it attempts to incorporate the different facets associated to a MOEA (including, as discussed above, decision making, robustness of the solutions and reduction of the number of objectives and computation times), together with the software/calculations required to evaluate specific aspects of the design and optimization problem (such as, for example, flow and heat transfer, aerodynamics, or structural mechanics). Also, even though the concept was developed having mainly in mind polymer processing, it is sufficiently flexible to be applicable to other engineering problems. As a matter of fact, the linkage to diverse engineering disciplines is mainly performed via the tools used to evaluate the solutions (which can vary from modeling software yielding quantitative data to qualitative aesthetical or comfort judgments).

As seen in Figure 1, the methodology involves necessarily analysis, modeling and optimization steps. One should begin with the identification of the problem characteristics, namely what are the major objectives and constraints, what are the main process parameters and whether automated calculation tools are available to provide solutions for subsequent evaluation, or qualitative and empirical knowledge should also apply.

An optimization step using MOEA will follow, with the aim of obtaining a good approximation to the Pareto front. The DM defines the relative importance of the various objectives via weights and, depending on his degree of confidence (which can be defined in the algorithm), the MOEA will be able to move towards Pareto frontiers of different sizes [16]. The robustness analysis may suggest fewer solutions. If the problem has many objectives, reduction of their number will be attempted [29]. Depending on the time spent on the evaluation of each objective function, the hybridization of MOEA with function approximation algorithms can also be adopted [25, 26]. The final result of this stage will consist of Pareto frontiers expliciting the trade-off between the different objectives and the decision variables (i.e., the parameters to be optimized).

The solutions will subsequently be presented to the DM in graphical form. This is a key step when non-quantifiable objectives exist, or when empirical knowledge must be also taken into consideration. The DM indentifies search space regions (on the objectives domain) that may satisfy his requirements in terms of the design (on the decision variables domain) - these are denoted as DM1 and DM2 in Figure 1.

In the following step, the methodology is used to tackle the inverse process, i.e., to determine the set of weights (one per objective) corresponding to the search space regions selected by the DM. This information is incorporated on the MOEA to generate new improved solutions. The process is repeated until the DM is satisfied with the results.

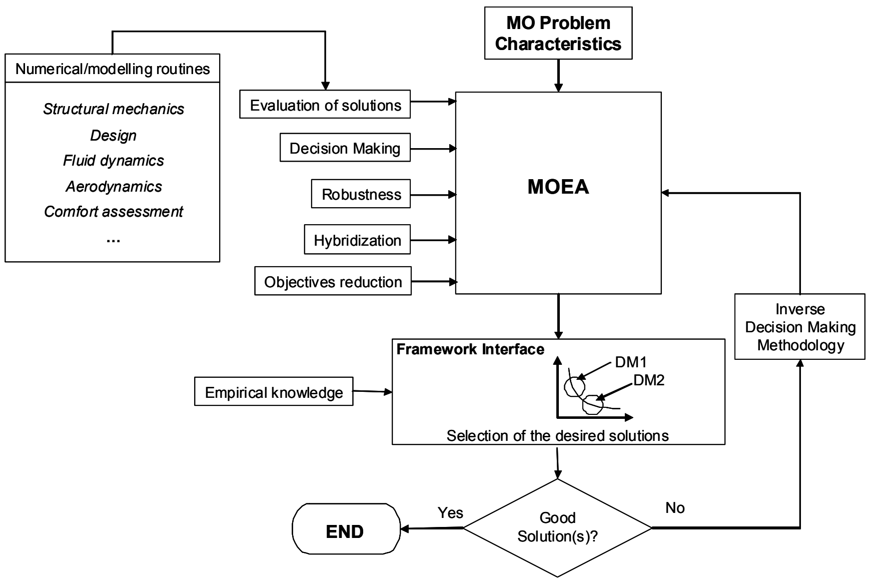

Figure 1. Integrated MO-MDO system.

2.2 Multi-Objective Evolutionary Algorithms

Following the ideas presented in Chapter 3, a MOEA named Reduced Pareto Set Genetic Algorithm, RPSGA, has been developed by the authors [12]. Since it will be used later in this Chapter, it makes sense to present it here in some detail.

In the RPSGA, the homogeneous distribution of the population along the Pareto frontier and the improvement of the solutions along successive generations are performed through the use of a clustering technique, which is applied to reduce the number of solutions on the efficient frontier [12]. The structure of the scheme is illustrated in Algorithm 1 below.

Algorithm 1: RPSGA

```
1    Initialize $p_e$ (external population) and Archive to empty set
2    $p_i$ is a randomly generated, initial population (internal)
3    while (termination condition not satisfied)
4        Evaluate $p_i$
5        Evaluate individuals' fitness considering clustering
6        Copy best individuals to $p_e$
7        if (external population full)
8            $p_e$ ← Clustering $p_e$
9            Copy best individuals of $p_e$ to $p_i$
10       endif
11       Select individuals for reproduction
12       Apply Inver-over operator to selected pairs of individuals
13       Add non-dominated solutions to Archive
14   end while
15   Filter Archive
16   Return Archive
end
```

Initially, an empty external population, p_e, and an empty archive are formed (line 1) and an internal population, p_i, of N candidate solutions is randomly generated (line 2). At each generation, that is, while a termination condition is not satisfied, the following operations are performed:

i) the candidate solutions of p_i are evaluated by the simulation routine (line 4);

ii) a clustering technique (Algorithm 2) is applied to reduce the number of solutions on the efficient frontier and to compute the fitness of each individual of p_i (line 5);

iii) a fixed number of the best individuals are copied to p_e (line 6);

iv) if p_e is full, the clustering technique (Algorithm 2) is applied again to sort the individuals of p_e (line 8) and a pre-defined number of the best individuals from p_e are incorporated into p_i to replace the lowest fitness individuals (line 9);

v) if p_e is not full, individuals of p_i are selected (line 11) for the application of the inver-over operator (line 12);

vi) all non-dominated solutions found during the computations are copied to the archive (line 13);

vii) all non-dominated solutions of the archive are returned after filtering it.

Algorithm 2 starts with the definition of the number of ranks, N_{Ranks}, (line 1) and the rank of each individual *i*, *Rank*[*i*], is set to 0 (line 2). For each rank, *r*, the population is reduced to *NR* individuals (i.e., *NR* is the number of individuals of each rank), using the clustering technique (lines 5 and 6). Then, rank *r* is attributed to these *NR* individuals (line 7) until the number of pre-defined ranks is reached (line 8). Finally, the fitness of individual *i, $F_{i,}$*, is calculated using a linear ranking function (line 9).

Algorithm 2: Clustering

```
1       Definition of $N_{Ranks}$
2       Rank[i] = 0
3       $r = 1$
4       do
5           $NR = r\,(N/NR_{anks})$
6           Reduce the population down to NR individuals
7           $r = r + 1$
8       while ($r < N_{Ranks}$)
9       Calculate fitness
10  end
```

In the computations below, the parameter settings that have been recommended in [12] are used. Hence, p_i and p_e have 100 and 200 individuals, respectively; a roulette wheel strategy is adopted for selection; the probability for applying the inver-over operator is 0.8. For more details and other parameter settings we refer to the original publication [12].

3 Decision Making

3.1 Current Methods

For effective application, the result of a multi-objective optimization process should consist of a single solution, not the set of solutions belonging to the Pareto front. In order to come up with this unique answer, at some point during the optimization the DM must decide what is the relative importance of the various objectives [10]. Consequently, the final outcome of a multi-objective optimization problem results not only from optimization (i.e., from the search process), but also from a decision course of action.

In this respect, existing multi-objective optimization methods [30, 31] are usually classified as no-preference or preference-based, depending on the consideration given to the relative importance of the objectives [32]. In the first case, the problem is solved regardless of that relative importance and the solution(s) obtained is/are made available to the DM, that accepts or rejects it/them. In the second case, the preferences of the DM are introduced in the search procedure and, in principle, the solution that best satisfies his preferences is selected [30].

There are, at least, three opportunities to introduce the DM preferences into the optimization scheme (see also Chapter 3): i) The various individual objectives are aggregated into a single objective, being necessary to decide *a priori* how that combination is realized; ii) Decision making and optimization are intertwined, *i.e.*, after an optimization step, the DM provides preference information on the set of available solutions, so that the optimization algorithm can proceed with the search; iii) The set of objectives is optimized simultaneously to obtain non-dominated vectors, the best solution being selected by integrating the DM preferences [10].

The open literature presents various methods of combining the search process with the DM preferences [10, 30], some being described in more detail in Chapter 3. For example, the decision making strategy proposed by Fonseca and Fleming [33] is based on pre-defined goals and priorities. If in a problem with n objectives, a preference vector g = [($g_{1,1}$, … , $g_{1,n1}$), … , ($g_{p,1}$, … , $g_{p,np}$)], where $\sum_{i=1}^{p} n_i = n$, is defined by the DM, then the sub-vectors g_i of the preference vector associate priorities i and goals g_{i,j_i} to the corresponding objective functions f_{i,j_i}. In this strategy, the comparison operator is structured taking into account, in each priority level, the components of the objective vector that do not meet their goals. Thus, the objective vectors u and v are compared in terms of their components with the highest priority p, disregarding those in which the u_p meets the corresponding goals. When both vectors meet all goals with this priority, or if they infringe at least some of them but exactly in the same way, the next priority level (p-1) is considered. The process continues until priority 1 is reached and satisfied, in which case the result is decided by comparing the priority 1 components of the two vectors. The implementation of this method requires the progressive articulation of preferences with the consequent changes on the environment (i.e., location of the solutions on the objective space) and a permanent interaction with the decision maker. Also, it requires the definition of two sets of parameters, namely the goals and the priorities for each objective, which is not an easy task.

Other decision methods can be mentioned, including weighted metrics [30], marginal rate of substitution [30], pseudo-weight vector [10], utility functions [34], biased sharing [10], guided domination [35], weighted domination [36] and the reference point base EMO [37]. Several difficulties may arise when applying these methods to real problems (again, see also Chapter 3). For example, some of the Pareto optimal solutions proposed by weighted metrics may not exist, depending on the problem's degree of non-convexity [30]. The marginal rate of substitution method requires a high computational effort [10]. The pseudo-weight vector does not perform satisfactorily for non-convex fronts, especially when a high importance is attributed to one of the objectives [10]; it is also very complex, as a weight vector must be calculated for each solution. Generally, in most of these methods the DM needs to define various algorithm parameters, which requires *a priori* a good knowledge of the Pareto front characteristics [30].

For illustrative purposes, the Weighted Sum Method (one of the weighted metrics approaches cited above) is applied here, as it is one of the simpler available schemes, transforming a problem with N objectives into a single objective optimization problem as follows:

$$\begin{aligned} &\text{Maximize} \quad F(x) = \sum_{i=1}^{N} w_i f_i(x) \\ &\text{Subject to} \quad x \in S \end{aligned} \tag{1}$$

where $\vec{w} = (w_1, \ldots, w_N)$ is a weighted vector representing the relative importance of each objective. The method can be used either *a priori* or *a posteriori*, depending on whether the DM expresses his preferences before or after the Pareto set approximation has been generated, respectively. In the first case, the optimization of a single objective is carried out, while in the second case the DM selects the best solution from the Pareto front obtained by a multi-objective optimization method. For a two-objectives to be minimized problem, as shown in Figure 2, the technique consists in shifting vertically upwards a straight line, having a slope given by the ratio between the weights attributed to each objective (w_1 and w_2, respectively), until it becomes tangent to the Pareto front contour. Each different pair w_1,w_2 generates a new solution. It is evident that even if the ratio of w_1 to w_2 changes extensively, the concave part of the Pareto front will never be reached.

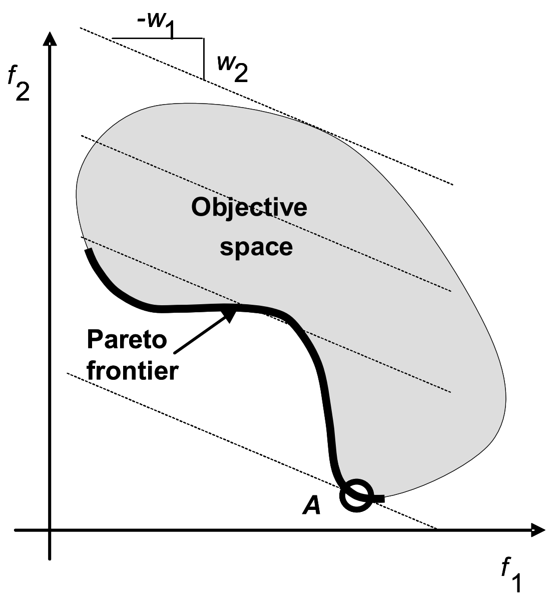

Figure 2. Application of the Weighted Sum Method to a non-convex Pareto-optimal front.

2.2. Weighted Stress Function Method

In an attempt to circumvent the limitations of the methods discussed in the previous Section, a different scheme is presented and analysed here. The Weighted Stress Function Method, WSFM [16], integrates the DM preferences once the search has been concluded (thus, the method is used *a posteriori*), which means that search and decision are sequential. WSFM is based on the assumptions that the best solution that will satisfy the DM preferences must belong to the Pareto Frontier, i.e., to the set of non-dominated solutions, and that the selection must take into consideration an ideal objective vector (denoted as Z^*) that maximizes each of the objective functions. The individual optimization of each objective corresponds to the maximum value of the global objective. The relative importance attributed to each individual objective will induce a "*stress*" to search for solutions that maximize each of the different objectives, the best solution being the one with zero stress.

The concept is based on the typical stress-strain behavior of thermoplastic vulcanizate polymer materials, TPV. A typical structure of a TPV consists of a high volume fraction ($0.40<v_p<0.9$) of cross-linked elastomeric particles suspended in a continuous thermoplastic matrix [38, 39]. Figure 3-A shows typical stress-strain curves for volume fractions of the elastomeric particles (v_p) between 0.0 to 1.0 [38]. Three regions with distinct behavior can be identified: i) at small strains, the stresses increase dramatically; ii) at intermediate strains, the stresses are relatively stable; iii) at high strains, a hardening effect develops. These differences decrease with increasing elastomeric particles volume fraction.

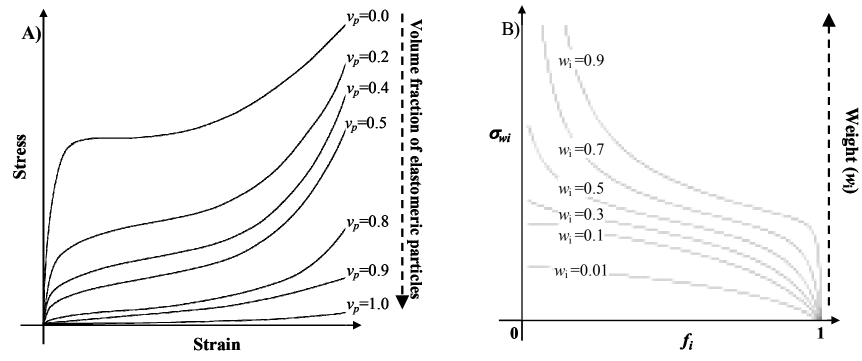

Figure 3. Analogy between TPV elasticity and the weighted stress function method (in the left plot the strain is normalized between 0 and 1).

The method makes an analogy between this stress-strain behavior and a stress function accounting for the weights defined for each objective. Indeed, the stress-strain plots for different volume fractions, v_p (Figure 3-A) and the stress function plots for different weights, w_i attributed to criterion i (Figure 3-B) are mirrored. The v_p and w_i values, ranging in the same interval [0,1], have the same role, i.e., to increase or decrease the stress, although they vary in opposite directions.

Figure 4 presents the Pareto frontier of an optimization problem with two objectives, f_1 and f_2, both to be maximized. For each solution belonging to this frontier, two stresses, σ_{w_1} and σ_{w_2}, are defined, each associated with the corresponding objectives. If w_1 and w_2 are the weights linked to each objective, each stress is proportional to the weighted distance between objective i and the i-th component of the ideal objective vector, z*.

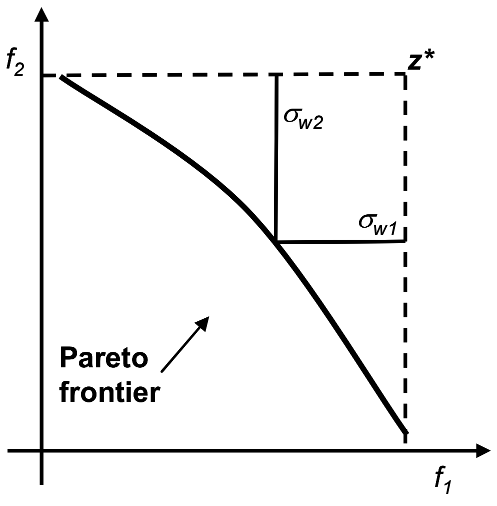

Figure 4. Stresses associated with an optimal solution, for an optimization problem with two objectives to be maximized.

Since the two objectives are conflicting, the ideal objective vector is not the solution to the problem and the stresses associated to the best solution are not nil. The best possible solution is that where the differences between the stresses associated with each objective are minimal. This implies that the value of σ_{w_1} is insufficient to search for the solutions with the best values of f_1, since an increase in the value of f_1 entails a decrease in f_2 and, consequently, an increase in σ_{w_2}. In turn, the increase of σ_{w_2} redirects the search for solutions with the best values for f_2, in detriment of the losses on f_1. Taking these considerations into account, a weighted stress function was mathematically defined in order to describe the shape of the stress versus f_i illustrated in Figure 3 right. This stress function is associated with weight w_i and is defined as [16]:

$$\sigma_{w_i}(f_i) = \begin{cases} \dfrac{w_i}{2}\tan\left(-\dfrac{\pi}{\psi(w_i)}(f_i - w_i)\right) + \xi(w_i), & f_i \le w_i \\[2ex] -\dfrac{\xi(w_i)}{\tan\left(\dfrac{\pi}{\varphi(w_i)}(w_i - 1)\right)}\tan\left(-\dfrac{\pi}{\varphi(w_i)}(f_i - w_i)\right) + \xi(w_i), & f_i > w_i \end{cases} \tag{2}$$

where:

$$\varphi(w_i) = \frac{3}{4}(1 - w_i)^2 + 2(1 - w_i) + \delta_1 \tag{3}$$

$$\psi(w_i) = \varphi(w_i) + 4w_i - 2 \tag{4}$$

$$\xi(w_i) = -\frac{1}{\tan\left(-\dfrac{\pi}{2 + 2\delta_2}\right)}\tan\left(\frac{\pi}{1 + \delta_2}\left(w_i - \frac{1}{2}\right)\right) + 1 \tag{5}$$

$$\delta_1 = 0.002 \quad \delta_2 = 0.008 \tag{6}$$

This description considers that the objectives are normalized, *i.e.*, $f_i \in [0, 1]$ and that the ideal vector is $z^* = (1, \ldots, 1)$. For a set of weights $\vec{w} = (w_1, \ldots, w_N)$, the resolution of a multi-objective optimization problem with N objectives consists in solving the following single objective problem:

$$\begin{aligned} &\text{Minimize} \quad T(X) = \max_i \sigma_{w_i}(f_i(X)) \\ &\text{Subject to} \quad X \in S \end{aligned} \tag{7}$$

In most optimization problems, only small regions of the Pareto frontier have practical interest [33, 40]; they are defined by the incorporation of the

DM's preferences. The definition of an ability function, combining the non-dominance concept with the relative importance of each objective, allows the selection of the solutions satisfying both aspects. Therefore, a different fitness function, *DF*, is defined, to account for the ranking of the solutions, as defined by the concept of non-dominance – Rank (*X*), and for the preferences of the DM, as quantified by equation 7 - *T*(*X*):

$$DF(X) = \text{Rank }(X) + \frac{T(X)}{T(X)+1} \tag{8}$$

This technique can be applied to any multi-objective algorithm. For example, the authors modified their Reduced Pareto Set Genetic Algorithm, RPSGA [13]. Initially, the RPSGA runs without modifications (*i.e.*, without introducing the DM preferences) during N_1 generations (search generations), with the aim of obtaining a first approximation to the Pareto frontier. Then, the modified algorithm will run during N_2 generations (decision generations) applying equation 7, in order to take into account the DM preferences. Simultaneously, a dispersion parameter, ε, ranging in the interval]0,1[, is used to control the extent of the Pareto sub-set; the higher its value, the wider the region. More details can be found elsewhere [16].

3 Robustness

Ideally, the performance of an optimal solution should be insensitive to the unavoidable changes that may occur in the design variables or in the environmental parameters. In other words, the solutions should not only be optimal, but also robust [41, 42]. In practice, different robustness-related types of problems can arise [41-46]: i) those where the performance is affected by the noise originated by sources such as sensor measurements and/or environmental parameters; ii) those where the design variables change after the optimal solution has been found; iii) those where the process performance is estimated by an approximation to the real value; iv) and those where the performance changes with time, which implies that the optimization algorithm must be updated continuously or periodically. Here, we are exclusively concerned with problems of the second category.

Given the above, the optimization algorithm should simultaneously determine the solution (or the set of solutions, in the case of multi-objective optimization) that maximize performance and that guarantee satisfactory robustness. Thus, not only a robustness analysis should be introduced as the search proceeds and not after, but also, as robustness and performance can be conflicting, it is important to know their interdependency for each optimization problem. Robustness analyses for single objective optimization have been applied to various engineering fields and using different optimization methodologies [45-50]. Conversely, only recently these studies have been extended to Multi-Objective Optimization [19, 21, 51-60]. Depending on the type of Pareto frontier, the aim of the analyses is either to locate the optimal Pareto front's most robust section [19, 57] or, in the case of a multimodal problem, to find instead the most robust Pareto frontier [56, 57].

The concept of robustness in a single objective problem can be introduced using Figure 5a, which shows the evolution of the objective function $f(x_1)$ (to be maximized) against the design parameter x_1. The most robust solution is the one for which the objective function f is less sensitive to variations of x_1. Clearly, solution S_2 is less sensitive than S_1 to variations of x_1 as the changes in $f(x_1)$ are less important (Δf_2 and Δf_1 for S_2 and S_1, respectively) and, consequently, it can be considered as the most robust solution. Since S_1 is the most performing and S_2 the most robust, a balance between performance (or fitness) and robustness has to be made [19, 54, 56-58].

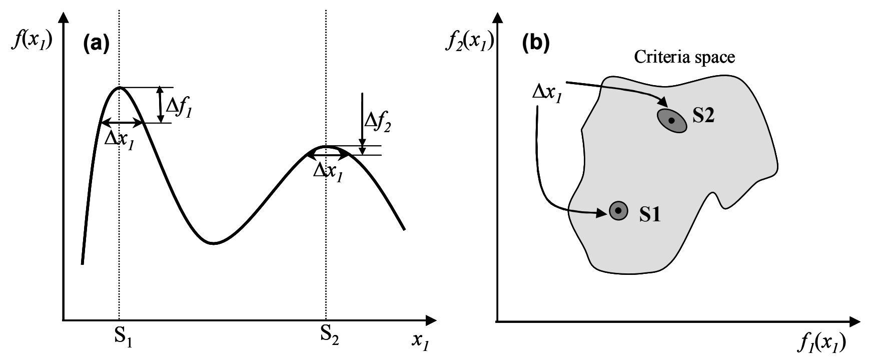

Figure 5. Concept of robustness for: a) single objective optimization; b) multi-objective optimization.

Two kinds of robustness measures have been proposed to deal with robustness [19, 21, 41, 54, 57]:

- *Expectation measure:* where the original objective function is replaced by a measure of both its performance and expectation in the vicinity of the solution considered. Various types of expectation measures have been proposed [19, 21, 43-45, 54, 57]. For example, if $P(\delta)$ is defined as the probability distribution of the disturbances δ in the neighborhood of the design point x_i, the expectation value, $F(x_i)$, of the objective function, $f(x_i)$, can be estimated from [43-45]:

$$F(x_i) = \int_{-\infty}^{+\infty} f(x_i, \delta)\, P(\delta)\, d\delta \tag{9}$$

- *Variance measure*: this consists in adding an additional objective to the objective function, to measure the deviation of the latter around the vicinity of the design point. Variance measures take only into account function deviations, ignoring the associated performance. Thus, in the case of a single objective function, the optimization algorithm must perform a two-objective optimization, one concerning performance and the other robustness [19, 21, 54, 57]. For example, Jin and Sendhoff [21] optimized simultaneously the original fitness function and the following robustness measure:

$$f^R = \frac{1}{N} \sum_{i=1}^{N} \frac{\sigma_f}{\sigma_{x_i}} \tag{10}$$

where σ_f and σ_x are the standard deviation of f and x, respectively. The smaller the f^R, the more robust the solution is.

These measures were classified by Deb and Gupta [57] as type I and II, respectively. Gaspar-Cunha and Covas [19] evaluated the performance of selected expectation and variance measures in terms of their capacity to detect robust peaks, by taking into account the following characteristics: i) easy application to problems where the shape of the objective function is unknown *a priori*, ii) capacity to define robustness regardless of that shape, iii) independence of the algorithm parameters, iv) clear definition of the function maxima in the Fitness *versus* Robustness Pareto representation and v) efficiency. The following variance measure exhibited the best overall performance:

$$f_i^R = \frac{1}{N'} \sum_{j=0}^{N} \left| \frac{\tilde{f}(x_j) - \tilde{f}(x_i)}{x_j - x_i} \right|, \qquad d_{i,j} < d_{\max}. \tag{11}$$

In this equation, the robustness of individual i is defined as the average value of the ratio between the difference of the normalized fitness of individual i, $\tilde{f}(x_i)$, to that of its neighbors (j), and the distance separating them, $\tilde{f}(x_j) = \frac{f(x_i) - f_{\min}}{f_{\max} - f_{\min}}$ for maximization and $\tilde{f}(x_j) = 1 - \frac{f(x_i) - f_{\min}}{f_{\max} - f_{\min}}$ for minimization of the objective function $f(x_i)$, with f_{max} and f_{min} representing the limits of its range of variation. N' is the number of population individuals whose Euclidian distance between points i and j ($d_{i,j}$) is lower than d_{max} (*i.e.*, $d_{i,j} < d_{max}$):

$$d_{i,j} = \sqrt{\sum_{m=1}^{M} \left(x_{m,j} - x_{m,i}\right)^2} \qquad (12)$$

and M is the number of objectives. As before, the smaller the f_i^R, the more robust the solution is.

Similarly to a single objective solution, a multi-objective robust solution must be little sensitive to variations of the design parameters. The concept is illustrated in Figure 5b: the same local perturbation on the parameter space, x_1, causes different consequences to the solutions located in diverse regions of the search space (S_1 and S_2). Those solutions that suffer smaller changes for the same perturbations on the parameters space are the most robust, i.e., S_1 is more robust than S_2 [19, 20]. In general, each of the Pareto optimal solutions must be analyzed in what concerns robustness. Consequently, the combined effect of the changes in all the objectives must be estimated and used as a robustness measure, the aim being to obtain a set of Pareto solutions that are concurrently multi-objectively robust and Pareto optimal. When dealing with multi-objective optimization different situations may arise, as illustrated in Figure 6 [56, 57]: i) every solution on the Pareto-optimal frontier is robust (Figure 6-A); ii) only some of the solutions belonging to the Pareto-optimal frontier are robust (Figure 6-B); iii) the solutions belonging to the Pareto-optimal frontier are not robust, but a robust Pareto frontier exists (Figure 6-C); iv) some of the robust solutions belong to the Pareto-optimal frontier, whereas others do not (Figure 6-D). Although a complete robustness analysis should be taken into consideration in all of the above situations, for simplicity reasons only situations i) and ii) will be discussed in the present chapter.

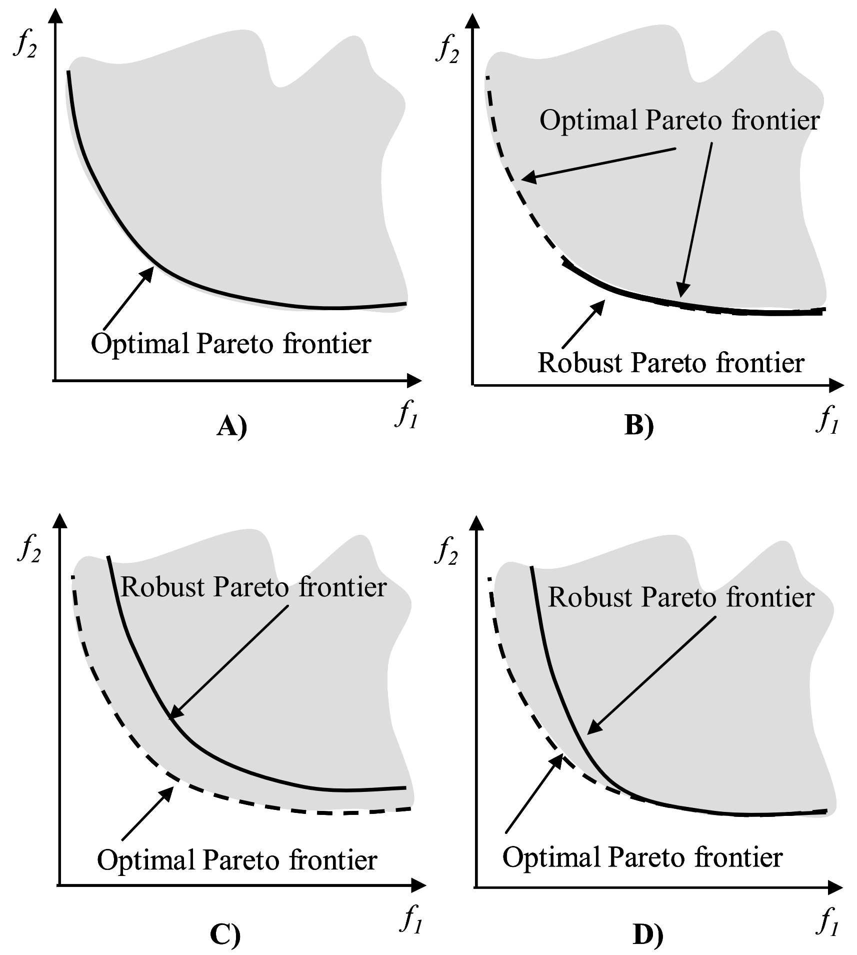

Figure 6. Different possible correlations between the Optimal Pareto frontier and the robust Pareto frontier (both objective functions f_1 and f_2 are to be minimized).

In order to implement a methodology to determine the set of Pareto solutions that are at once robust and optimal, Gaspar-Cunha and Covas [11] added three new steps to the RPSGA:

i) the computation of robustness measures by taking into account a dispersion parameter, ε', that quantifies the extension of the robustness zone. Its value is defined by the DM and ranges between 0 (when a single solution is to be obtained) and 1 (when the entire optimal Pareto frontier is to be obtained);

ii) the calculation of the niche count and the determination of the global fitness. The latter is considered using a sharing function [61]:

$$m(i)=\sum_{j=1}^{N} sh(d_{ij}) \tag{13}$$

where $sh(d_{ij})$ is related to individual i and takes into account its distance to all its neighbors j (d_{ij});

iii) Finally, the global fitness is calculated using the following equation:

$$\tilde{F}(i)=Rank(i)+(1-\varepsilon')\frac{R(i)}{R(i)+1}+\varepsilon'\frac{m(i)}{m(i)+1} \tag{14}$$

The details of this implementation are given elsewhere [19, 20]. Anyway, this methodology can be adapted to be coupled to different MO methods, such as those described in Chapter 3.

4 Memetic Algorithms

4.1 General concepts

One of the major complications in applying MOEAs to real problems is the large number of objective function evaluations that are necessary to obtain an acceptable solution - typically of the order of several thousands. Moreover, each of these evaluations is often time-consuming, as it involves the use of expensive numerical codes. The possibility of reducing the number of evaluations needed to reach an acceptable solution is thus of major practical importance. Finding good approximate methods is particularly hard for multi-objective problems, due to the eventual high number of individual objectives and to the possible interactions between them. Nevertheless, different approaches have been pursued [22-25, 62-66], which can be grouped in terms of the MOEA stage where they are applied:

a) *During evaluation*: some solutions can be evaluated using an approximate function, such as statistical methods, Fitness Inheritance, Artificial Neural Networks, etc, which reduces the required number of exact evaluations;

b) *During local search after recombination*: a number of new individuals is generated by local search algorithms;

c) *During recombination*: a few individuals can be generated using more efficient methods.

The last two approaches generate a faster approximation to the optimal Pareto frontier and, consequently, the number of evaluations is reduced.

An example of the application of the first approach was proposed by Nain and Deb [63], whereby an iterative procedure, where a neural network is trained with a set of exact evaluations of the fitness function, is used. Subsequently, the genetic algorithm uses the neural network to estimate the objective function for a fixed number of generations. The process is repeated until an adequate solution is attained. Recently, this hybrid approach has been extended to MOEAs [62].

Alternatively, EAs could be coupled to local search methods, giving rise to Memetic Algorithms [67]. The main idea consists in obtaining, at each MOEA generation, some (good) solutions through the use of an efficient local search algorithm, or improve the search speed by introducing some selection pressure [25, 66]. At each MOEA generation, new solutions are determined using not only the recombination operators, but also an inverse mapping of the objectives into the decision variables. For this purpose, an Artificial Neural Network, ANN, was coupled to a MOEA. A few modifications to the method have been suggested [68-70]. For example, MOEAs could be coupled to an Inverse Artificial Neural Network, IANN, and the new improved solutions incorporated into the current MOEA's population. This requires an additional RPSGA step, following the selection phase. A few solutions belonging to the non-dominated front are selected, a local search procedure being initiated from each, so that new better solutions are generated and incorporated into the main population.

4.2 Coupling MOEAs to a local search method

Artificial Neural Networks, ANN, implemented by a Multilayer Perceptron, are flexible schemes capable to approximate an arbitrary complex function, if assured that enough training data is offered [71, 72]. Basically, an ANN builds a map between a set of inputs and their related outputs, and is particularly well suited to non-linear regression analyses of noisy signals and of incomplete data. An ANN consists of a set of input nodes connected to a set of output nodes, using one or more intermediate nodes. A weight is attributed to each connection, initially randomly, but later it is adjusted during the training process. For a specific layer, the output in each node is given as a function of the sum of the weights coming from previous layers. The training process consists basically on an iterative scheme where a set of examples are presented to the ANN, which is interrupted when the error produced by the ANN is lower than a pre-defined value, or when the number of iterations reaches a certain limit. The definition of the best ANN to be used on a specific problem depends on the number of input and output nodes, on the number of intermediate layers and nodes and on the training method applied. A more thorough explanation is available [62, 71, 72].

Combining ANN with EAs is a powerful approach to address the exploitation/exploration dilemma. Neural Networks can be trained to build a smooth map of the fitness landscape and, for that reason, are adequate to perform a local search exploiting specific regions for possible candidate solutions. EAs are adequate for a global search, since they are efficient in exploring huge search spaces of multivariate functions with many local minima. The goal is to implement an approximation to the fitness functions of multi-objective optimization, which is independent of the objectives to be optimized and of the EA parameters. The training data is composed of previous exact function evaluations, performed by the evolutionary algorithm.

Considering the preceding discussion, the use of an Inverse ANN, IANN, is suggested [25, 66]. The use of an ANN with a single hidden layer of variable size that is adjustable to the complexity of the problem is advantageous. The idea is to train the neural network in an inverse way, that is, the inputs fed to the network are the objectives, while the outputs are the independent variables. Tentative individuals are generated by the IANN before being presented to the MOEA. The network is used to perform a local search near to the non-dominated solutions from the previous generation, in order to directly discover new tentative solutions with higher fitness in their vicinity. This stratagem guides locally the MOEA, thus avoiding extrapolations to regions not covered by the training data. As explained above, the training process is made using the solutions available from exact function evaluations of previous generations.

All inputs and outputs can be normalized between 0 and 1. An additional step is introduced after the selection phase (i.e., between lines 11 and 12 of Algorithm 1) and all solutions are evaluated using an exact function evaluation. The IANN is only used to obtain new solutions on the decision variables domain. Figure 7 illustrates the local search operator for a problem with two objectives to be minimized. First, the best individuals of the present generation are selected (individuals *1* to *4*, *e1* and *e2* in Figure 7). For the farthest solutions *e1* (minimum found so far for objective 1) and *e2* (minimum found for objective 2), three new individuals are generated. For instance, for *e1* the new coordinates of the solutions *a*, *b* and *c* are generated as follows:

$$\begin{aligned} &\text{solution}\,a: \quad \vec{C}^{new} = (C_1 + \Delta C_1, C_2 + \Delta C_2) \\ &\text{solution}\,b: \quad \vec{C}^{new} = (C_1, \quad C_2 + \Delta C_2) \\ &\text{solution}\,c: \quad \vec{C}^{new} = (C_1 - \Delta C_1, \quad C_2) \end{aligned} \tag{15}$$

where ΔC_j is the displacement applied to each objective (a parameter to be empirically defined). For solutions 1 to 4, new tentative ones are obtained according to [66]:

$$\vec{C}^{new} = (C_1 + \Delta C_1, C_2 + \Delta C_2) \tag{16}$$

For each new point generated, the use of IANN identifies the corresponding individual in the input decision variables space. Only a percentage of these new individuals is later used to form a new population that is fed to the MOEA.

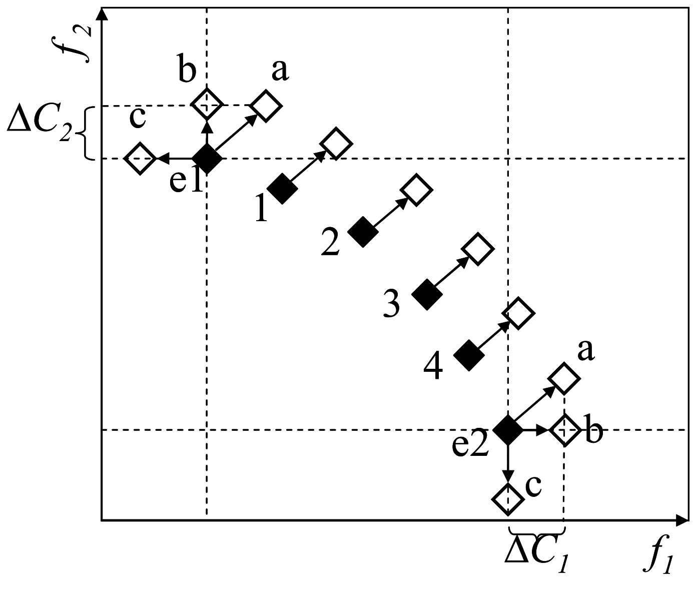

Figure 7. Scheme used for the Inverse ANN (IANN) local approximation.

A different local search procedure, based on the use of a Pattern Search Filter Method and exhibiting promising results, has also been proposed [73].

The Filter Method was introduced by Fletcher and Leyffer [74, 75] as an alternative to merit functions, to guarantee global convergence in different iterative methods for nonlinear programming problems. The procedure consists in performing a local search in the neighborhood of some non-dominated solutions selected from the Pareto frontier. The concept of non-dominance is incorporated to build a filter that is able to accept good trial iterates and enforce global convergence from arbitrary starting points.

5 Application Examples

For illustrative purposes, the methods presented and discussed along this chapter are applied separately here to 3 Test Problems, TP, each of a different type and with distinctive Pareto frontier characteristics. In chapter 5, the Decision Making and the Robustness strategies will be used in problems involving single and twin-screw extrusion.

TP1 is a simple one parameter problem (*L*=1) with two objectives (*M*=2); TP2 has 30 parameters (*L*=30) and two objectives (*M*=2), while TP3 has 2 parameters (*L*=2) and 3 objectives (*M*=3):

- **TP1**: $x \in [-2;6]$; minimize; *L*=1; *M*=2.

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= e^{|x|-5} + (6/5)\cos(2x) - 2{,}7x + 1 \end{aligned} \tag{17}$$

- **TP2** (ZDT1): $x_i \in [0;1]$; minimize; *L*=30; *M*=2 [11].

$$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(x_2,\cdots,x_L) &= g(x) \times \left(1 - \sqrt{f_1(x_1) \big/ g(x)}\right) \\ \text{with}, g(x) &= 1 + 9\frac{\sum_{l=2}^{L} x_l}{L-1} \end{aligned} \tag{18}$$

- **TP3**: $x_1 \in [0;2\pi]$, $x_2 \in [0;5]$; minimize; *L*=2; *M*=3.

$$\begin{aligned} f_1(x) &= \sin(x_1).g(x) \\ f_2(x) &= \cos(x_1).g(x) \\ f_3(x) &= {x_2}^2 \\ g(x) &= e^{x_2(x_2-5)} - (6/5)\sin(2x_2) - 2{,}7x_2 - 1 \end{aligned} \tag{19}$$

The RPSGA parameters are the following:
- $N_{ranks} = 20$;
- $d_{max} = 0.008$;
- indifference limits equal to 0.1 for all objectives;
- SBX real crossover operator with an index of 10;
- real polynomial mutation operator with an index of 20.

Decision Making

Figures 8 and 9 show the results obtained from the application of the DM strategy described in section 2 to TP1 and TP2, and TP3, respectively. A dispersion parameter, $\varepsilon = 0.1$ was used. It is clear that the method is sensitive to changes in the relative importance of the objectives. For example, in the case of the bi-objective problems (Figure 8) when the relative importance of f_1 decreases (moving from (a) to (c)), the algorithm converges to higher values of this objective (remember that the objectives are to be minimized). This performance is also evident for the three objectives problem (Figure 9), with the solutions converging to the best location that corresponds to the weights chosen. Moreover, the topography of the surface generated for each set of weights changes due to the non-uniformity of that of the optimal Pareto front. Finally, the method takes into account the magnitude of the dispersion parameter, as only a small portion of the Pareto curve is obtained. A more complete set of results has been reported elsewhere [16].

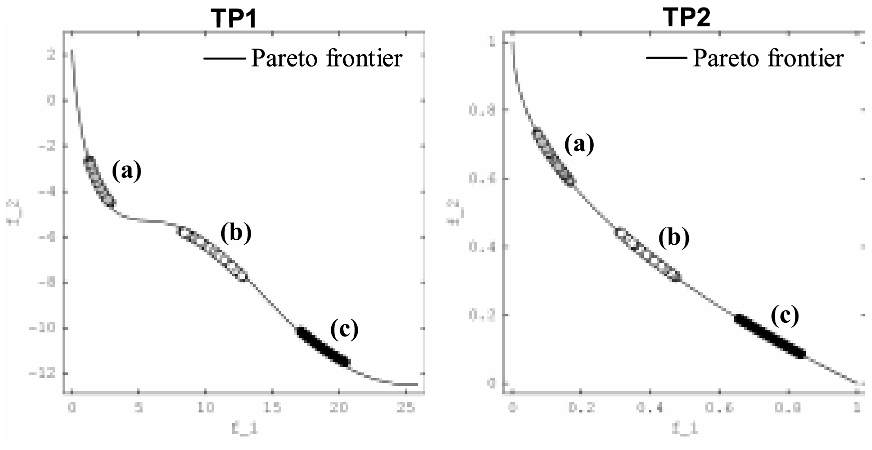

Figure 8. DM results for TP1 and TP2 and different weight vectors: (a) (0.8,0.2), (b) (0.5,0.5) and (c) (0.2,0.8). The continuous line shows the optimal Pareto frontier.

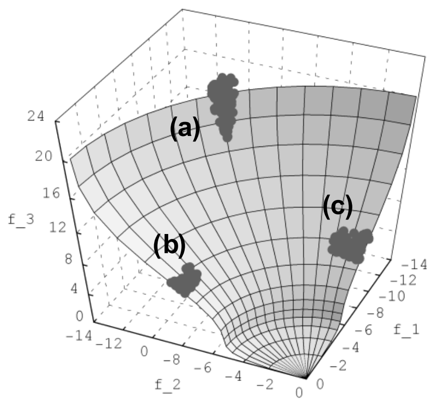

Figure 9. DM results for test problem TP3 with different weight vectors: (a) (0.5,0.5,0.0), (b) (0.0,0.5,0.5) and (c) (0.5,0.0,0.5). The grey surface shows the 3-dimensional optimal Pareto frontier.

Robustness

The influence of the dispersion parameter ε on the robustness testing is shown in Figure 10 for TP1. When $\varepsilon = 0.2$, the algorithm converges to a relatively narrow robust region, where f_1 is approximately 5. As ε increases, so does the size of the region selected. Figure 11 presents the most robust regions for TP2 and TP3, when $\varepsilon =$ to 0.1 (again, more comprehensive data can be analyzed elsewhere [20]).

Memetic Algorithm

In a recent study, both hybrid techniques referred above, i.e., RPSGA coupled with IANN and PSFM, were applied to a few benchmark problems [73]. The first step of this study was to set the best values of the Memetic algorithm parameters. For example, Figure 12 shows the results obtained when the RPSGA algorithm is applied by itself (RPSGA), and coupled to the inverse neural network (IANN) to TP2. Figure 12a shows the influence of the number of solutions selected to which the IANN is applied (N_{sol}), while Figure 12b shows the influence of the displacement applied (the same for the two objectives) as defined in equation 14. In this case, the best values are N_{sol} equal to 30 and ΔC equal to 0.5, which are used in the remaining test problems studied.

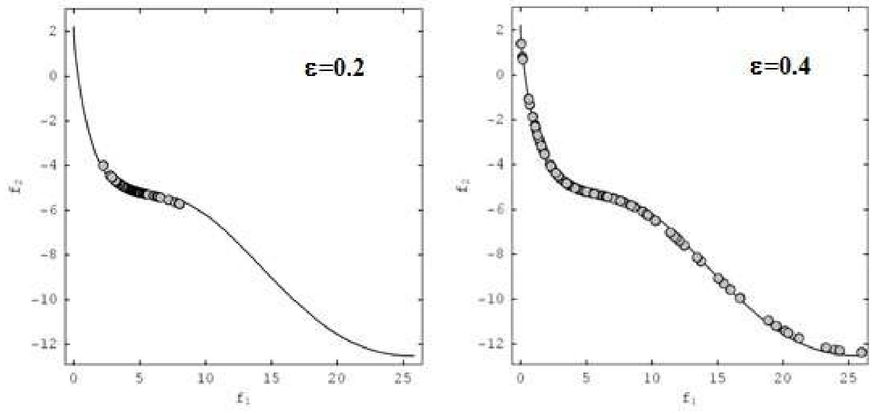

Figure 10. Effect of ε on robustness for TP1.

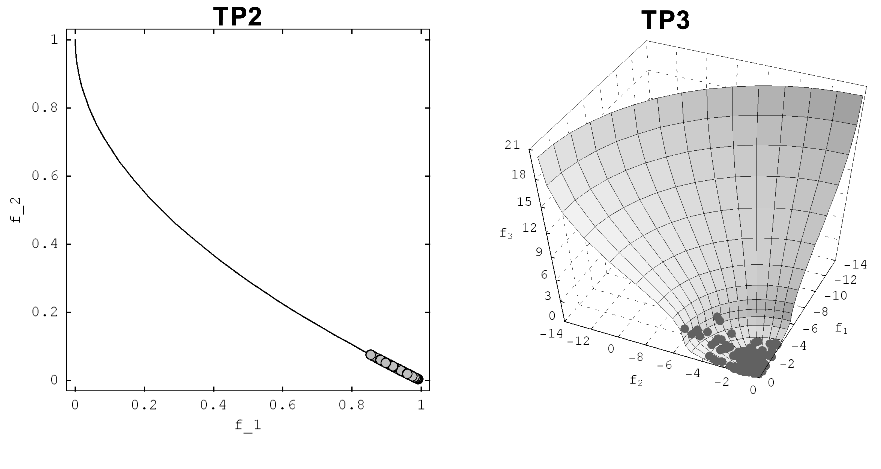

Figure 11. Robustness for TP2 and TP3 for $\varepsilon = 0.1$.

Then, a comparative study with different algorithms was performed. Figure 12 compares the performance of different Memetic algorithms applied to TP2. Included are RPSGA, NSGA-II [77], RPSGA coupled with IANN and PSFM [73]. Since the hypervolume metric can be taken as a measure of the quality of the Pareto frontiers [76], the figure shows its evolution with the number objective functions evaluations required by each algorithm. The RPSGA-IANN performance is much higher, similar results having been obtained with other test problems [66, 73].

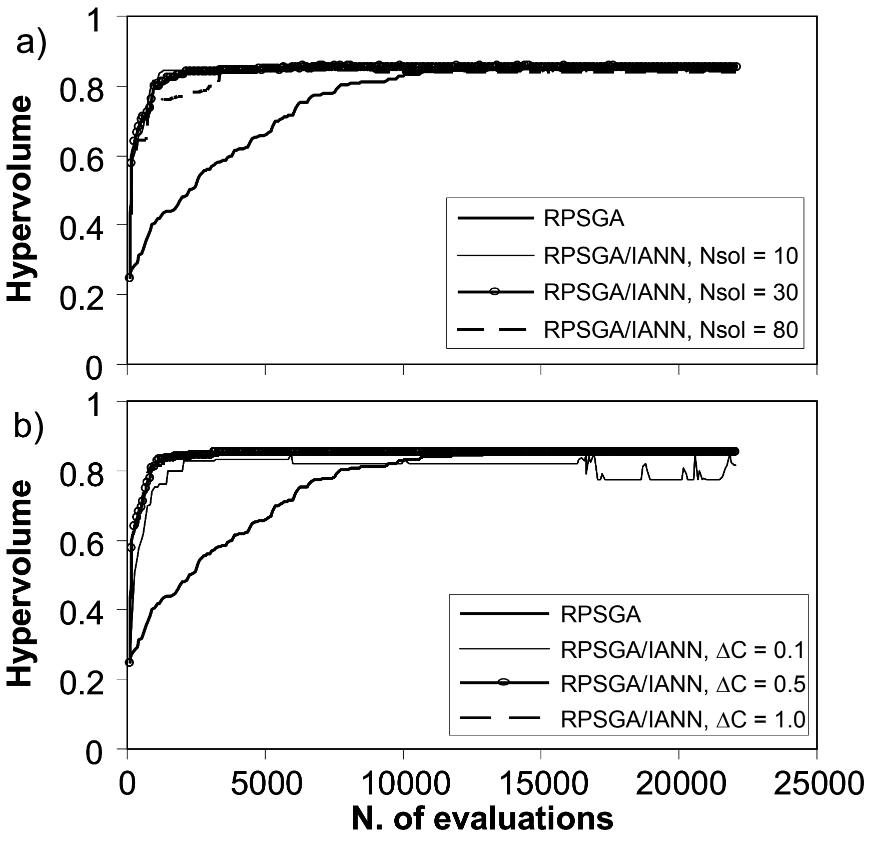

Figure 12. Evolution of the hypervolume metric as a function of the number of evaluations: a) influence of the number of solutions selected for applying the IANN approach (N_{sol}) and b) influence of the displacement applied (the same for the two objectives) as defined in equation (14).

6 Conclusion

A Multi-Objective Multidisciplinary Design Optimization (MO-MDO) approach was proposed to deal with the practical complexities of large multi-objective problems. The methodology links a MOEA to decision making and robustness strategies that are able to assist the decision maker in selecting the best solutions that satisfy his preferences and/or are sufficiently robust against changes of the values of the decision variables, respectively. Also, with the aim of reducing the computation time often required by the evaluation routines (by decreasing the number of required real function evaluations) two different Memetic algorithms were suggested. Application of the methods to a few test problems demonstrated their effectiveness.

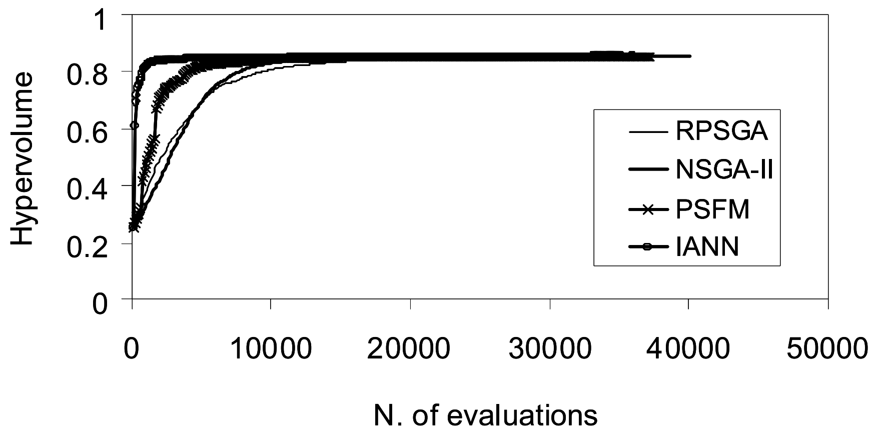

Figure 13. Performance of different Memetic algorithms applied to TP2.

References

[1] Bentley, P.J. *Evolutionary Design by Computers*, Morgan Kaufmann: San Francisco, CA, 1999.

[2] Avriel, M.; Rijckaert, M.J.; Wilde, D.J. (eds.), *Optimization and Design*, Prentice-Hall: Englewood Cliffs, NJ, 1973.

[3] Avriel, M.; Dembo, R.S. (eds.), *Mathematical Programming Studies on Engineering Optimization*, North-Holland: New York, NY, 1979.

[4] Siddall, J.N., *Optimal Engineering Design*, CRC: London, 1982.

[5] Mack, Y.; Goel, T.; Shyy, W.; Haftka, R. In *Evolutionary Computation in Dynamic and Uncertain Environments*; Yang, S.; Ong, Y-S.; Jin, Y.; Ed.; Studies in Computational Intelligence, Springer, 2007; Vol. 51, pp. 323–342.

[6] Oztemela, E.; Tekezb, E.K. *Engineering Applications of Artificial Intelligence*. 2009, 22, 855-864.

[7] Luo, Z.; Chen, L.; Yang, J.; Zhang, Y.; Abdel-Malek, K. *Struct Multidisc Optim*. 2005, 30: 142–154.

[8] Grujicic, M.; Arakere, G.; Sellappan, V.; Ziegert, J.C.; Koçer, F.Y.; Schmueser, D. *Multidiscipline Modeling in Materials and Structures*. 2009, 5, pp. 1-28.

[9] Dulikravich,G.S.; Dennis,B.H.; Martin,T.J.; Egorov,I.N. In *Evolution Methods for Design, Optimization and Control*; Giannakoglou, K.; Tsahalis, D.; Periaux, J.; Papailiou, K.; Fogarty, T.; Ed.; CIMNE, Barcelona, 2001.

[10] Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley: Chichester, UK, 2001.

[11] *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Branke, J.; Deb, K.; Miettinen, K.; Slowinski, R.; Ed.; Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2008.

[12] Gaspar-Cunha, A.; Covas, J.A. In *Metaheuristics for Multiobjective Optimisation*. Gandibleux, X.; Sevaux, M.; Sörensen, K.; T'kindt, V.; Ed.; Springer: Berlin, Germany, 2004; pp.221-249.

[13] García-Martínez, C.; Cordón, O.; Herrera, F. *European J. of Operational Research*, 180,116-148, 2007.

[14] Stützle, T.; Hoos, H. In *Proceedings of Artificial Neural Nets and Genetic Algorithms*. Smith, G.D.; Steele, N.C.; Albrecht,R.F.; Ed.; Springer Verlag Wien, 1998; pp.245-249.

[15] Paquete, L.; Stützle, T. In *Handbook of Approximation Algorithms and Metaheuristics*. Gonzalez, T. F.; Ed.; Computer and Information Science Series, Chapman & Hall, Boca Raton, FL, USA, 2007.

[16] Ferreira, J.C.; Fonseca, C.M.; Gaspar-Cunha, A. In *Evolutionary Methods For Design, Optimization and Control*. Neittaanmäki, P.; Périaux, J.; Tuovinen, T.; Ed.; CIMNE, Barcelona, 2008, pp. 197-202.

[17] Ray, T. In *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, 2002, pp. 419-424.

[18] Jin, Y.; Branke, J. *IEEE Transactions on Evolutionary Computation*, 2005, 9, 303-317.

[19] Gaspar-Cunha, A.; Covas, J. *Computational Optimization and Applications*, 2008, 39, 75-96.

[20] Ferreira, J.; Fonseca, C.; Covas, J.A.; Gaspar-Cunha, A. In *Advances in Evolutionary Algorithms*, Kosińsk, W.; Ed.; I-Tech Education and Publishing, Vienna, Austria, 2008, pp. 261-278.

[21] Jin, Y.; Sendhoff, B. In *Proceedings of the Second Int. Conf. on Evol. Multi-Objective Optimization -EMO'2003*, Faro, Portugal, 2003, pp. 237-251.

[22] Jin, Y.; Olhofer, M.; Sendhof, B. *IEEE Trans. on Evolutionary Computation*. 2002, 6, 481-494.

[23] Poloni, C.; Giurgevich, A.; Onesti, L.; Pedirola, V. *Computer Methods in Applied Mechanics and Engineering*. 2000, 186, 403-420.

[24] Talukder, A.K.A.; Kirley, M.; Buyya, R. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary cconference*. New York, NY, USA: ACM, 2008, pp. 721–728.

[25] Gaspar-Cunha, A.; Vieira, A. In *European Conference on Applications of Genetic Algorithms EUROGEN 2003*, Barcelona, 2003, pp. 157-165.

[26] Gaspar-Cunha, A.; Vieira, A. In *Hybrid Metaheuristics (HM 2004) Workshop at ECAI 2004*. Valencia, Spain, 2004, pp. 25-30.

[27] Norman, M.G.; Moscato, P. In *Technical Report Caltech Concurrent Computation Program*, Report 790, California Institute of Technology, Pasadena, California, USA, 1989.

[28] Deb, K.; Saxena, D. In *IEEE Congress on Evolutionary Computation*. IEEE Computer Society Press, Los Alamitos, 2006, pp. 3353-3360.

[29] Costa, L.; Oliveira, P. *PAMM*, Vol. 7, pp. 2060047 – 2060048, Wiley, 2007.

[30] Miettinen, K. *Nonlinear Multiobjective Optimization*, Kluwer: Boston, 1999.

[31] Kaliszewski, I. *Soft Computing for Complex Multiple Criteria Decision Making*, Springer: Berlin, 2006.

[32] Hwang, C.L.; Masud, A.M. In *Lecture Notes in Economics and Mathematical Systems*, vol.164, 1979.

[33] Fonseca, C.M.; Fleming, P.J. *IEEE Transactions on Systems, Man and Cybernetics*, 28, pp.26-37, 1998.

[34] Keeney, R.L.; Raiffa, H. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley: New York, 1976.

[35] Branke, J.; Deb,K. In *Knowledge Incorporation in Evolutionary Computation*, 2004, pp.461-477.

[36] Parmee, I.C.; Cevtkovic, D.; Watson,A.W.; Bonham, C.R. *Evolutionary Computation*, 2000, 8, 197-222.

[37] Deb, K.; Sundar, J.; Bhaskara, U.; Chaudhuri, S. *ISSN International Journal of Computational Intelligence Research*, 2, 2006, 273-286.

[38] Boyce, M.C.; Kear, K.; Socrate, S.; Shaw, K. *Journal of the Mechanics and Physics of Solids*, 2001, 49, 1073-1098.

[39] Coran, A.Y.; Patel, R. *Rubber Chemistry and Technology*, 1980, 53,141-150.

[40] Fonseca, C.M.; Fleming, P.J. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993, pp.141-153.

[41] Ray, T. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002, pp. 419-424, Honolulu.

[42] Jin, Y.; Branke, J. *IEEE Transactions on Evolutionary Computation*, 9, 2005, 303-317.

[43] Wiesmann, D.; Hammel, U.; Bäck, T. *IEEE Transactions on Evolutionary Computation*, 1998, 2, 162-167.

[44] Das, I. *Nonlinear Multicriteria Optimization and Robust Optimality*, Rice University, PhD Thesis, Houston, 1997.

[45] Tsutsui, S.; Ghosh, A. *IEEE Transactions on Evolutionary Computation*, 1997, 1, 201-208.

[46] Chen, W.; Sahai, A.; Messac, A.; Sundararaj, G. In *Structural Dynamics and Materials Conference*, St. Louis, USA, 1999.

[47] Ribeiro, J.L.; Elsayed, E.A. *Int J Prod Res*, 1995, 33, 3233-3248.

[48] Du, X.; Chen, W. In *Engineering Design, Proceedings of DETC 99*, Las Vegas, USA, 1999.

[49] Arrold, D.V.; Beyer, H.-G. *Computational Optimization and Applications*, 2003, 24, 135-159.

[50] Sörensen, K. *J. of Mathematical Modelling and Algorithms*, 2004, 3, 89-103.

[51] Kouvelis, P.; Sayin, S. Annals of Operational Research, 2006, 147, 71–85.

[52] Bagchi, T.P. Materials and Manufacturing Processes, 2003, 18, 341-354.

[53] Kazancioglu, E.; Wu, G.; Ko, J.; Bohac, S.; Filipi, Z.; Hu, S.; Assanis, D.; Saitou, K. In *Proceedings of DETC'03,* pp. 1-12, Chicago, USA, 2003.

[54] Gaspar-Cunha, A.; Covas, J. In *Proceedings of 10th Online World Conference in Soft Computing in Industrial Applications*, pp. 189-193, 2005, Springer, Berlin.

[55] Olvander, J. *J. of Engineering Design*, 2005, 16, 511-523.

[56] Guanawan, S.; Azarm, S. *Struct. Multidisciplinar Optimization*, 2005, 29, 50-60.

[57] Deb, K.; Gupta, H. *Evolutionary Computation*, 2006, 14,463-494.

[58] Paenk I., Branke, J.; Jin, Y. *IEEE Transations on Evolutionary Computation*, 2006, 10, 405-420.

[59] Barrico, C.; Antunes, C.H. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 6778-6783, Vancouver, Canada, 2006.

[60] Moshaiov, A.; Avigrad, G. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 6785-6791, Vancouver, Canada, 2006.

[61] Goldberg, D.; Richardson, J. In *Proceedings of Second Int. Conf. on Genetic Algorithms*, pp. 41-49, 0-8058-0158-8, Cambridge, 1985.

[62] Gaspar-Cunha, A.; Vieira, A., *International Journal of Computers, Systems, and Signals*, 2005, 6, 18-36.

[63] Nain, P.K.S.; Deb, K. (2002) Kangal Report No. 2002005, http://www.iitk.ac.in/kangal/deb.htm.

[64] Poloni, C., Giurgevich, A., Onesti, L., Pedirola, V. *Computer Methods in Applied Mechanics and Engineering*, 2000, 186, 403-420.

[65] Talukder, A. K. A., Kirley, M., Buyya, R. In *Proceedings of the 10th annual conference on Genetic and evolutionary*, 2008, pp. 721-728.

[66] Gaspar-Cunha, A.; Vieira, A.S.; Fonseca, C.M. In *Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*, November, Nottingham, UK, 2004.

[67] Krasnogor, N., Smith, J.E. *IEEE Transactions on Evolutionary Computation*, 2005 9, 474-488.

[68] Adra, S.F., Griffin, I., Fleming, P.J. Multiobjective Memetic Algorithms, 2009, 183-208.

[69] Adra, S.F., Griffin, I., Fleming, P.J. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2005, pp. 1009-1010.

[70] Soh, H., Ong, Y. S., Salahuddin, M., Hung, T., Lee, B. S. In *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, 2007, pp. 325-332.

[71] Bishop, C.; Ed.; *Neural Networks for Pattern Recognition*, Oxford University Press. 1997.

[72] Bull, L. *Soft Comput. Fusion Found. Methodol. Appl.*, 1999, 3, 76-82.

[73] Gaspar-Cunha, A.; Mendes, F.; Costa, M.F.P. International Transactions in Operational Research, In press, 2010.

[74] Fletcher, R.; Leyffer, S. *Filter-type Algorithms for Solving Systems of Algebraic Equations and Inequalities*. Dundee Numerical Analysis Report NA/204, 2001.

[75] Fletcher, R., Leyffer, S. *Mathematical Programming*, 2002, 91, 239-269.

[76] Zitzler, E.; Deb K.; Thiele, L. *Evolutionary Computation*, 2000, 8, 173-195.

[77] Deb, K.; Pratap, A.; Agrawal, S.; Meyarivan, T. *IEEE Transactions on Evolutionary Computation*, 2002, 6, 182-197.