Herrero Durá, JM.; Reynoso Meza, G.; Martínez Iranzo, MA.; Blasco Ferragud, FX.; Sanchís Saez, J. (04-2). A Smart-Distributed Pareto Front Using ev-MOGA Evolutionary Algorithm. International Journal of Artificial Intelligence Tools. 23(2):1-22. https://doi.org/10.1142/S021821301450002X

International Journal on Artificial Intelligence Tools

A Smart-distributed Pareto Front using the ev-MOGA Evolutionary Algorithm

J.M. Herrero, G. Reynoso-Meza, M. Martínez, X. Blasco and J. Sanchis

Predictive Control and Heuristic Optimization Group
Department of Systems Engineering and Control
Universitat Politècnica de València
Camino de Vera 14, 46022 - Valencia, Spain
juaherdu@isa.upv.es, gilreyme@upv.es, mmiranzo@isa.upv.es, xblasco@isa.upv.es, jsanchis@isa.upv.es

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Obtaining multi-objective optimization solutions with a small number of points smartly distributed along the Pareto front is a challenge. Optimization methods, such as the normalized normal constraint (NNC), propose the use of a filter to achieve a smart Pareto front distribution. The NCC optimization method presents several disadvantages related with the procedure itself, initial condition dependency, and computational burden. In this article, the epsilon-variable multi-objective genetic algorithm (ev-MOGA) is presented. This algorithm characterizes the Pareto front in a smart way and removes the disadvantages of the NNC method. Finally, examples of a three-bar truss design and controller tuning optimizations are presented for comparison purposes.

Keywords: multi-objective optimization; Pareto front; engineering design; evolutionary algorithms; multi-objective evolutionary algorithms.

1. Introduction

Many engineering design problems can be translated into multi-objective optimization (MO) problems. MO techniques offer advantages over single-objective optimization approaches because they enable a set of solutions to be found with different trade-offs among the objectives. Therefore, the decision maker (DM) can analyze the set and select the best solution. These three steps (measure, search, and selection) are fundamental for the successful application of the MO technique.[1]

In some engineering fields, problem design is based on single-objective optimization techniques that weigh different objective functions to obtain the best solution from the design variables. [2] Choosing weighting factors for the cost index is usually a tedious trial-and-error process, and due to the configuration of the index (for example, linear or quadratic), it is often impossible to find good trade-off solutions. [3] This is because most the cost function to be optimized is usually stated from the point of view of the optimizer, despite a possible loss of flexibility when defining the

desired balance among objectives.

The MO methodology enables the designer to carry out a better selection of the final solution, since no part of the searching space is ignored. Solutions provided by MO algorithms should be representative of the whole design variable space. Since computational algorithms perform a discrete search in the space of design variables, the group of solutions found should be evenly distributed to avoid over- or under-explored areas. This group of solutions should not contain non-optimal solutions, since this situation could lead the DM to select a potentially inappropriate value for some design variables.

Solving an MO problem could be associated with the approximation of the Pareto front. Each point of this Pareto front represents a solution to the MO problem in the objective function space, which is a Pareto optimal solution.[4] That is, for any given pair of Pareto optimal solutions, an improvement in one of the components entails a deterioration in the others. Therefore, we will have a set of optimal solutions, with differing trade-offs among the objectives. This is because there is usually no overall optimal solution, which is the best solution for each individual objective.

MO algorithms based on numerical optimization and random search are analyzed in [5] and a new numerical optimization method was proposed: the normalized normal constraint (NNC).[a] This approach offers acceptable properties since it generates well-distributed Pareto front approximations. However, because it uses a search-based Gauss-Newton method, the solution obtained is highly dependent on the objective selected for optimization and on the initial optimization conditions.

To avoid this major problem a modified variant of the NNC (MNNC) can be used which overcomes the above mentioned disadvantages.[6,7] MNNC enables the construction of the Pareto front regardless of the objective selected for optimization. It also presents an alternative to the construction of the Pareto front based on the redistribution of front points, and uses a genetic algorithm (GA) to achieve global optimum solutions - but without dependence on the initial conditions. The principal drawback of this approach is its high computational burden since an independent optimization process (using a GA) is needed to achieve each point of the Pareto front.

However, MNNC (and therefore also NNC) cannot characterize certain areas of the Pareto front (as will be shown in Section 2.1) when:

- The optimal solution of two objectives (or more) is the same (for problems with three or more objectives).
- The optimal solution of one objective (or more) is multimodal.

Both the NNC and MNNC algorithms approximate evenly distributed Pareto fronts but they are not necessarily the most appropriate approximation. In some

[a]An implementation of the NNC algorithm is available in MATLAB Central. http://www.mathworks.com/matlabcentral/fileexchange/38976

cases, it could be more interesting for the less desired parts of the Pareto front to have a lower degree of characterization than the more desired parts.[8] For example, the regions where the slope of the front is lower could be characterized with a lower density of solutions. This kind of distribution is known as a smart distribution.[9] To achieve a smart distribution, both NNC and MNNC first need to accomplish a uniform, dense distribution of solutions on the front; they then use a smart filter to reduce the number of points in the regions with less slope. Solutions with practically insignificant trade-off (PIT) are omitted from the Pareto set approximation. The resulting set will be smaller, alleviating the DM's need to compare solutions with uninteresting trade-offs. This is important, since the selection procedure is usually more time consuming than the optimization process[1]. This also means that some points will be eliminated, despite the computational burden invested in obtaining them.

Another interesting alternative for solving MO problems is based on the use of evolutionary algorithms (EAs), which allow several elements of the Pareto front to be generated simultaneously (in parallel and in a single run) owing to the populational nature of EAs. ([10,11,12,13])

Many different operators or strategies have been developed that convert the original EAs into multiobjective evolutionary algorithms (MOEAs). MOEAs converge towards the Pareto optimal set and their solution is diverse enough to be able to characterize it. The good results obtained with MOEAs and their capacity to handle a wide variety of problems with different degrees of complexity explain why they are being increasingly used; [14] indeed they are currently one of the areas where most progress is being made within the field of EAs. [15,16,17,18,19,20]

In this work, a new MOEA algorithm called the epsilon-variable multi-objective genetic algorithm (ev-MOGA)[b] has been designed to achieve a reduced but well-distributed representation of the Pareto front. Front solutions are smartly distributed without using a filter, so avoiding the need to eliminate solutions *a posteriori*, and ensuring that no computational burdens are wasted. In addition, the algorithm adjusts the limits of the Pareto front dynamically, and prevents solutions belonging to the ends of the front from being lost. This algorithm, as it will be shown, incorporates the PIT criterion. This feature makes it an algorithm closer to the decision making step that is fundamental in the MO technique. This is important, since MOEAs usually only focus on providing a dense set of Pareto optimal solutions - regardless of the subsequent selection process.[21]

To evaluate the performance of the ev-MOGA algorithm we used two optimization problems and compared the results with those obtained using the NNC algorithm with a smart filter. This paper is organized as follows. Section 2 presents the mathematical foundations of the NNC method and the smart filter. Section 3 presents the ev-MOGA algorithm based on the $\epsilon-$dominance concept. Sections 4

[b]ev-MOGA algorithm is now available in MATLAB Central. http://www.mathworks.com/matlabcentral/fileexchange/31080

and 5 compare the performance of the ev-MOGA and NNC algorithms with two optimization problems: a three-bar truss example; and a proportional-integral (PI) controller tuning problem. Finally, some concluding remarks are provided in Section 6.

2. NNC Method with Smart Pareto Filter

The MO problem can be formulated as follows:

$$\min \boldsymbol{J}(\boldsymbol{\theta}) = \min[J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta}), \ldots, J_s(\boldsymbol{\theta})] \tag{1}$$

subject to:

$$\begin{array}{rl} g_q(\boldsymbol{\theta}) \leq 0, & (1 \leq q \leq r) \\ h_k(\boldsymbol{\theta}) = 0, & (1 \leq k \leq n) \\ \theta_{li} \leq \theta_i \leq \theta_{ui}, & (1 \leq i \leq L) \end{array} \tag{2}$$

where $J_i(\boldsymbol{\theta})$, $i \in B := [1 \ldots s]$ are the objectives to be optimized, $\boldsymbol{\theta}$ is a solution inside the L-dimensional solution space D, $g_q(\boldsymbol{\theta})$ and $h_k(\boldsymbol{\theta})$ are each of the r inequality and n equality problem constraints respectively, and θ_{li} and θ_{ui} are the lower and upper constraints that defined the solution space D.

To solve the MO problem the Pareto optimal set $\boldsymbol{\Theta}_P$ (solutions where none dominate any of the others) must be found. Pareto dominance is defined as follows:

A solution $\boldsymbol{\theta}^1$ dominates another solution $\boldsymbol{\theta}^2$, denoted by $\boldsymbol{\theta}^1 \prec \boldsymbol{\theta}^2$, if

$$\forall i \in B, J_i(\boldsymbol{\theta}^1) \leq J_i(\boldsymbol{\theta}^2) \; and \; \exists k \in B : J_k(\boldsymbol{\theta}^1) < J_k(\boldsymbol{\theta}^2) \; .$$

Therefore, the Pareto optimal set $\boldsymbol{\Theta}_P$ is given by

$$\boldsymbol{\Theta}_P = \{\boldsymbol{\theta} \in D \mid \nexists \; \tilde{\boldsymbol{\theta}} \in D \; : \; \tilde{\boldsymbol{\theta}} \prec \boldsymbol{\theta}\} \; . \tag{3}$$

$\boldsymbol{\Theta}_P$ is unique and normally includes infinite solutions. Hence a set $\boldsymbol{\Theta}_P^*$, with a finite number of elements from $\boldsymbol{\Theta}_P$, should be obtained.[c]

Below, an extract of the NNC method to solve an n-objectives optimization problem is presented. A detailed description of the method can be found in [5].

Step 1: Anchor points computation. Firstly, the minimum of each objective function, J_i^* $(i \in B)$, is calculated by solving the following optimization problems:

$$\boldsymbol{\theta}^{i*} = \min \; J_i(\boldsymbol{\theta}) \qquad (i \in B) \tag{4}$$

subject to (2).

[c]Notice that $\boldsymbol{\Theta}_P^*$ is not unique.

The anchor points thus obtained determine the ends of the Pareto front $J_i^* = J(\boldsymbol{\theta}^{i*})$. Additionally, the utopia point,[d] denoted by $\mathbf{J}^u$, comprises the good elements of each anchor point.

$$\mathbf{J}^u = [\, J_1(\boldsymbol{\theta}^{1*})\; J_2(\boldsymbol{\theta}^{2*}) \cdots J_s(\boldsymbol{\theta}^{s*})\,]^T \tag{5}$$

Step 2: Objective space normalization. By defining the matrix L_s as the maximum distances in each component of the anchor points relative to the utopia plane, a normalization of the searching space can be performed.

$$L_s = \left[\, l_1\; l_2 \cdots l_s \,\right]^T = \mathbf{J}^S - \mathbf{J}^u \tag{6}$$

where $\mathbf{J}^S$ is the nadir point

$$\mathbf{J}^S = \left[\, {J_1}^S\; {J_2}^S \cdots {J_s}^S \,\right] \tag{7}$$

$${J_i}^S = max\left[\, J_i(\boldsymbol{\theta}^{1*})\; J_i(\boldsymbol{\theta}^{2*}) \cdots J_i(\boldsymbol{\theta}^{s*}) \,\right] \tag{8}$$

which leads to the normalized design metric as

$$\overline{J}_i = \frac{J_i - J_i(\boldsymbol{\theta}^{i*})}{l_i}, i \in B \tag{9}$$

Step 3: Utopia line vector generation. Let vectors $\overline{N}_k$ be defined as the difference between the normalized anchor vectors (Figure 1), from $\overline{J}^{k*}$ to $\overline{J}^{s*}$ for $k \in 1, 2, ..., s-1$.

$$\overline{N}_k = \overline{J}^{s*} - \overline{J}^{k*} \tag{10}$$

Step 4: Normalized increment definition. The normalized increment δ_k is defined, in direction $\overline{N}_k$ for a prescribed number of solutions, as m_k.

$$\delta_k = \frac{1}{m_k - 1}, (1 \le k \le s-1) \tag{11}$$

where the resulting segment size can be expressed as

$$\Delta_k = \delta_k |\overline{N}_k| \tag{12}$$

Step 5: Generate utopia line points. The points distributed over the utopia hyperplane are described as

$$\overline{X}_{pj} = \sum_{k=1}^{s} \alpha_{kj} \overline{J}^{k*} \tag{13}$$

where

$$\textstyle\sum_{k=1}^{s} \alpha_{kj} = 1 \;\; 0 \le \alpha_{kj} \le 1 \tag{14}$$

[d]Since it is the best point, but cannot be achieved.

Step 6: Pareto front approximation. The **NNC** method states that the solution to the MO problem (1) can be transformed into the minimization of X_{pj} single-objective problems, but in the normalized domain. The optimization problem can be formulated as:

$$\min \overline{J}_s(\theta) \tag{15}$$

subject to:

$$\begin{array}{rl} g_q(\boldsymbol{\theta}) \leq 0, & (1 \leq q \leq r) \\ h_k(\boldsymbol{\theta}) = 0, & (1 \leq k \leq n) \\ \theta_{li} \leq \theta_i \leq \theta_{ui}, & (1 \leq i \leq L) \end{array} \tag{16}$$

$$\begin{array}{c} \overline{N}_k^T(\overline{J} - \overline{X}_{pj}) \leq 0 \quad k = 1 \ldots s-1 \\ \overline{\mathbf{J}} = [\overline{J}_1(\boldsymbol{\theta}) \quad \cdots \quad \overline{J}_s(\boldsymbol{\theta})]^T \end{array} \tag{17}$$

Note that for each problem j, $s-1$ additional constraints (17) are added. Each constraint represents the scalar product of vector $\overline{N}_k$ and the vector formed by the difference between the points of the feasible area $\overline{\mathbf{J}}$ and point $\overline{X}_{pj}$. By making this scalar product smaller than zero, the optimization is forced to search for the minimum value when the hyperplanes are in opposition.[e] This ensures that this minimum ($\overline{\boldsymbol{\theta}}^{j*}$) in the Pareto front will be found for each point $\overline{X}_{pj}$ (see Figure 1).

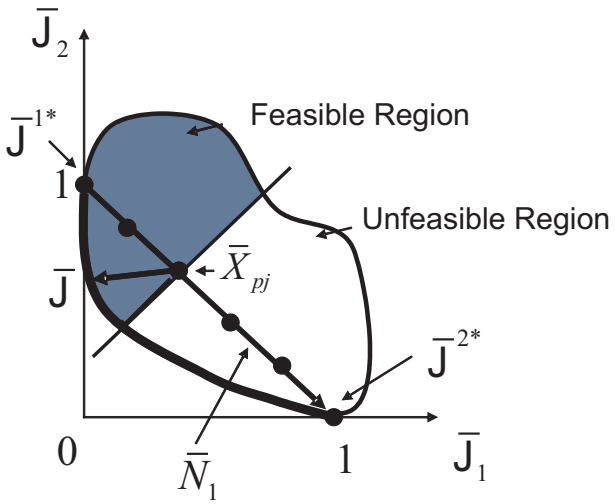

Figure 1. NNC in the bi-objective case and $m_1 = 6$. For the sake of simplicity, only the bi-objective case is presented graphically.

[e]In the bi-objective case, hyperplanes are singled-out vectors (Figure 1).

The construction of the Pareto front from equation 15 can include non-Pareto or local Pareto points in its solution.[f] $\boldsymbol{\Theta}^*$ is defined as the discrete set of all the solutions found in the optimization problems (15) and (4) that are carried out. A filter is used to eliminate the solutions that do not belong to the Pareto front and to obtain the final $\boldsymbol{\Theta}_P^*$.

$$\boldsymbol{\Theta}_P^* = \{\boldsymbol{\theta}^* \in \boldsymbol{\Theta}^* | \nexists \ \tilde{\boldsymbol{\theta}} \in \boldsymbol{\Theta}^* \ : \ \tilde{\boldsymbol{\theta}} \prec \boldsymbol{\theta}^*\}. \tag{18}$$

Notice that $\boldsymbol{\Theta}_P^*$ will contain Pareto points that are evenly distributed across the Pareto front.

The density of the points should be high enough to allow characterization of this front. To obtain a reduced sample of points of $\boldsymbol{\Theta}_P^*$ in [9] the application of a smart filter to $\boldsymbol{\Theta}_P^*$ to obtain $\boldsymbol{\Theta}_{SP}^*$ is proposed. Therefore, from $\boldsymbol{\Theta}_P^*$ several points are eliminated using the PIT criterion to obtain $\boldsymbol{\Theta}_{SP}^*$ (see [9] for more details about PIT and the smart filter).

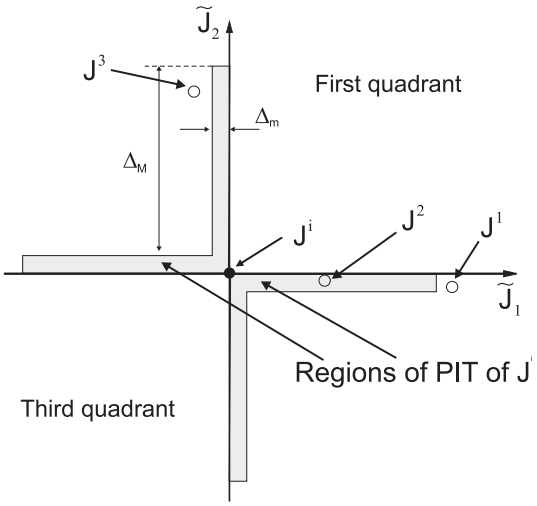

Figure 2. Regions of the PIT for the bi-objective case. The gray area is the $\mathbf{J}^i$ PIT. $\tilde{\mathbf{J}}^2$ and $\tilde{\mathbf{J}}^1$ represent deviations of the respective objectives from the point $\mathbf{J}^i$. $\mathbf{J}^2$ belongs to the $\mathbf{J}^i$ PIT, and therefore it will be removed when the smart filter is applied, whereas $\mathbf{J}^1$ and $\mathbf{J}^3$ do not belong to $\mathbf{J}^i$ PIT and so they will continue in $\boldsymbol{\Theta}^*$.

Given a Pareto front point $\mathbf{J}^i$, its PIT is defined by means of Δ_m and Δ_M designer parameters as shown in Figure 2[g]. For two Pareto points in $\boldsymbol{\Theta}_P^*$ whose difference between their objectives values is less than Δ_m, the PIT criterion prevents

[f]Local Pareto points are those that are not locally dominated by any other point. Non-Pareto points are locally dominated.

[g]Since the smart filter is applied to $\boldsymbol{\Theta}_P^*$ Pareto points, the first and third quadrants are not populated and consequently are not considered.

them being in $\mathbf{\Theta}^*_{SP}$ - unless the difference between any other objectives is greater than Δ_M.

Let ν be the absolute vector between the two points being compared:

$$\nu = abs(\mathbf{J}^i - \mathbf{J}^k).$$

Therefore $\mathbf{J}^k$ is removed when it is compared to $\mathbf{J}^i$, if

$$\nu_m < \Delta_m \; and \; \nu_M < \Delta_M$$

where ν_m and ν_M are minimum and maximum vector components of ν.

In Figure 2, for instance, as $\mathbf{J}^2$ belongs to the $\mathbf{J}^i$ PIT it will be removed when the smart filter is applied to $\mathbf{\Theta}^*$, whereas $\mathbf{J}^1$ and $\mathbf{J}^3$ do not belong to $\mathbf{J}^i$ PIT and so they will continue in $\mathbf{\Theta}^*$ for the moment.

A smart filter starts with a $\mathbf{J}^i$ in $\mathbf{\Theta}^*_P$ which is declared smart and compared with successive points $\mathbf{J}^k$ in $\mathbf{\Theta}^*_P$. Points in $\mathbf{J}^i$ PIT are eliminated from $\mathbf{\Theta}^*_P$. This procedure is repeated with other $\mathbf{J}^i$ points in $\mathbf{\Theta}^*_P$ until every point in $\mathbf{\Theta}^*_P$ has been declared a smart point. The points finally remaining in $\mathbf{\Theta}^*_P$ constitute $\mathbf{\Theta}^*_{SP}$.

Figure 3 shows the process of smart filtering. ∘ points are eliminated because they belong to PIT regions of • points which are declared smart. Given a $\mathbf{\Theta}^*_P$ set, the resulting $\mathbf{\Theta}^*_{SP}$ is not unique since it depends on the analysis order followed, as shown in cases (a) and (b) in Figure 3.

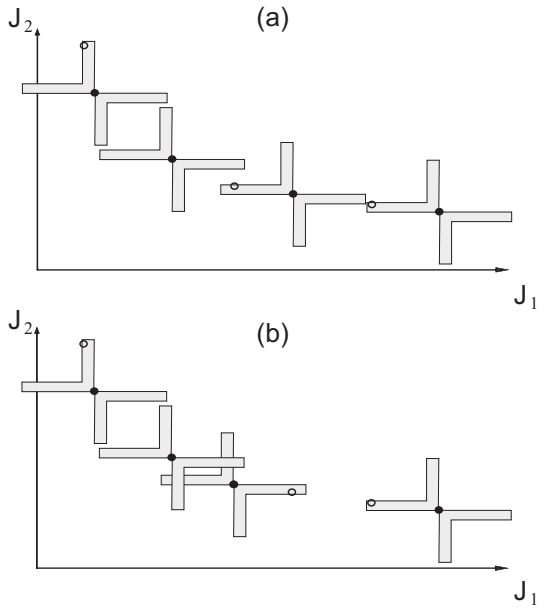

Figure 3. • and ∘ constitute $\mathbf{\Theta}^*_P$. • represents the $\mathbf{\Theta}^*_{SP}$ obtained after eliminating points in PIT regions. (a) Analysis order 1. (b) Analysis order 2.

2.1. *NNC algorithm drawbacks*

The procedure used by the NNC algorithm is extremely dependent on the anchor point calculation. Therefore, if these are not correct the algorithm may not adequately characterize parts of the Pareto front. Two examples of this situation are shown in this section. An example is shown in Figure 4 when objective functions are multimodal and therefore the anchor points obtained could not correspond with the real ends of the Pareto front. This is because the anchors are static and are not updated in the NNC algorithm once calculated.

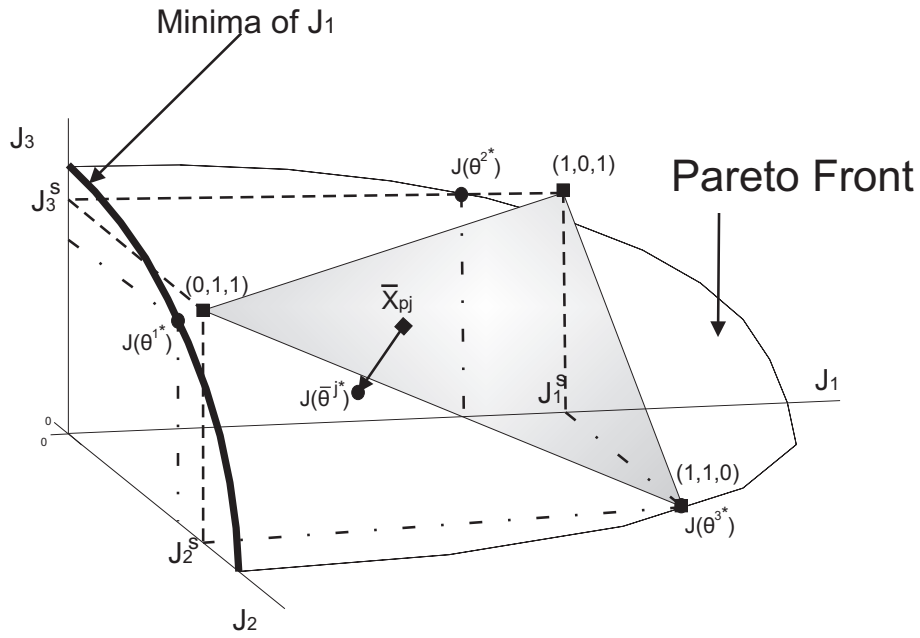

Figure 4. NNC in a three objective case when the objective functions are multimodal.

For instance, if J_1 is minimized any solution in the highlighted line (minimum of J_1) could be obtained. Assume that the $\boldsymbol{J}(\boldsymbol{\theta}^{1*})$ solution is obtained and that $\boldsymbol{J}(\boldsymbol{\theta}^{2*})$ and $\boldsymbol{J}(\boldsymbol{\theta}^{3*})$ are also obtained when J_2 and J_3 are minimized, respectively. These solutions define the utopia point $(0,0,0)$ and the nadir point (J_1^S, J_2^S, J_3^S) used to normalize the objective space and to define the utopia plane (limited by the points represented by squares in the figure). For each point $\overline{X}_{pj}$ in the utopia plane, a single-objective optimization is made and a Pareto front point $\boldsymbol{J}(\boldsymbol{\theta}^{j*})$ will be obtained that characterises the Pareto front under the utopia plane. However, as the utopia plane does not completely cover the Pareto front, it would not be totally characterized and parts of it would not be obtained.

Another example is shown in Figure 5. In this three objective case, the minimum solution for objectives J_1 and J_2 is the same, and the utopia plane is reduced to a

line. This fact, means that only Pareto points under this line would be obtained, and therefore, the Pareto front would be incompletely characterized.

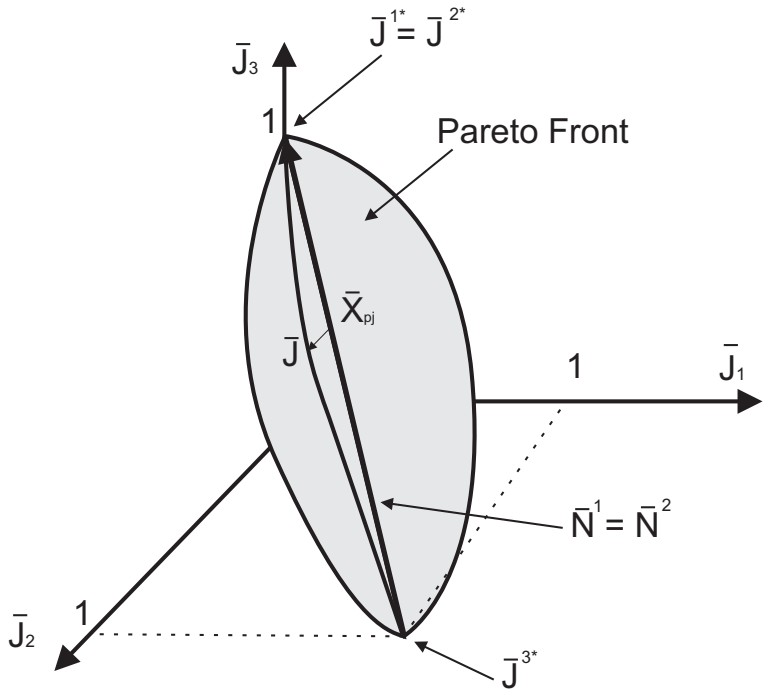

Figure 5. NNC in a three objective case when two anchor points are the same $\overline{J}^{1*} = \overline{J}^{2*}$.

3. ev-MOGA

The ev-MOGA [22] is an elitist multi-objective evolutionary algorithm based on the concept of ϵ-dominance, [23] which is used to control the content of the archive $A(t)$ where the result of the optimization problem is stored. ev-MOGA tries to ensure that $A(t)$ converges toward an ϵ-Pareto set, $\mathbf{\Theta}^*_{P_\epsilon}$ in a smart distributed manner along the Pareto front $\mathbf{J}(\mathbf{\Theta}_P)$ with limited memory resources. This is due to the ϵ-dominance concept which helps maintain solutions with significant trade-off and the dynamic adjustment of the limits of the Pareto front by preserving its extremes (anchors). This creates the possibility of overcoming the aforementioned problems in the NNC algorithm.

For this reason, the objective space is split into a fixed number of boxes. For each dimension $i \in B$, n_box_i cells of ϵ_i width are created where

$$\epsilon_i = (J_i^{max} - J_i^{min})/n\_box_i, \tag{19}$$

$$J_i^{max} = \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}^*_{P_\epsilon}} J_i(\boldsymbol{\theta}),\ J_i^{min} = \min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}^*_{P_\epsilon}} J_i(\boldsymbol{\theta}). \tag{20}$$

This grid preserves the diversity of $\mathbf{J}(\boldsymbol{\Theta}^*_{P_\epsilon})$ since each box can be occupied by only one solution in $A(t)$, and at the same time produces a smart distribution as will be shown later.[h]

The concept of ϵ-dominance is defined as follows. For a solution $\boldsymbol{\theta} \in D$, $box_i(\boldsymbol{\theta})$ is defined by

$$box_i(\boldsymbol{\theta}) = \left\lceil \frac{J_i(\boldsymbol{\theta}) - J_i^{min}}{J_i^{max} - J_i^{min}} \cdot n\_box_i \right\rceil \ \forall i \in B. \tag{21}$$

Let $\mathbf{box}(\boldsymbol{\theta}) = \{box_1(\boldsymbol{\theta}), \ldots, box_s(\boldsymbol{\theta})\}$. A solution $\boldsymbol{\theta}^1$ with value $\mathbf{J}(\boldsymbol{\theta}^1)$ ϵ-dominates the solution $\boldsymbol{\theta}^2$ with value $J(\boldsymbol{\theta}^2)$, denoted by $\boldsymbol{\theta}^1 \prec_\epsilon \boldsymbol{\theta}^2$, if and only if

$$\mathbf{box}(\boldsymbol{\theta}^1) \prec \mathbf{box}(\boldsymbol{\theta}^2) \vee \left(\mathbf{box}(\boldsymbol{\theta}^1) = \mathbf{box}(\boldsymbol{\theta}^2) and \boldsymbol{\theta}^1 \prec \boldsymbol{\theta}^2\right)\ . \tag{22}$$

Hence, a set $\boldsymbol{\Theta}^*_{P_\epsilon} \subseteq \boldsymbol{\Theta}_P$ is ϵ-Pareto if and only if

$$\forall \boldsymbol{\theta}^1, \boldsymbol{\theta}^2 \in \boldsymbol{\Theta}^*_{P_\epsilon}, \boldsymbol{\theta}^1 \neq \boldsymbol{\theta}^2,\ \mathbf{box}(\boldsymbol{\theta}^1) \neq \mathbf{box}(\boldsymbol{\theta}^2) and \mathbf{box}(\boldsymbol{\theta}^1) \not\prec_\epsilon \mathbf{box}(\boldsymbol{\theta}^2) \tag{23}$$

Therefore, ev-MOGA is responsible for updating the content of $A(t)$ by saving only ϵ-dominant solutions that do not share the same box. When two mutually ϵ-dominant solutions compete, the solution that prevails in $A(t)$ will be the one that is closest to the center of the box. It is thereby possible to prevent solutions belonging to adjacent boxes (neither of them dominating the other) from being too close to each other, thus encouraging a smart distribution.

The aim of ev-MOGA is to achieve a $\boldsymbol{\Theta}^*_{P_\epsilon}$ with the greatest possible number of solutions in order to characterize the Pareto front adequately. Although the number of possible solutions will depend on the shape of the front and for n_box_i, it will not exceed the following level

$$|\boldsymbol{\Theta}^*_{P_\epsilon}| \leq \frac{\prod_{i=1}^{n} n\_box_i + 1}{n\_box_{max} + 1},\ \ n\_box_{max} = \max_i n\_box_i \tag{24}$$

which is advantageous, as it is possible to control the maximum number of solutions that will characterize the Pareto front.

Furthermore, thanks to the definition of the box, the anchor points $J_i(\boldsymbol{\theta}^{i*})$ are assigned a value of $box_i(\boldsymbol{\theta}^{i*}) = 0$, whereby $J_i(\boldsymbol{\theta}^{i*}) = J_i^{min}$. Therefore, no solution θ can ϵ-dominate because, by applying the definition of box, their $box_i(\boldsymbol{\theta}) \geq 1$.

Figure 6 shows what $\boldsymbol{\Theta}^*_{P_\epsilon}$ would be obtained by applying concepts of ϵ-dominance for a bi-objective example, when $n\_box_1 = n\_box_2 = 10$ is used. The values ϵ_1 and ϵ_2 depend on the limits of the front J_1^{min}, J_2^{min}, J_1^{max} and J_2^{max}, which adjust dynamically in accordance with the utopia solution calculated in each

[h]The algorithm only checks occupied boxes (not all boxes). This content management of $A(t)$ avoids the need to use other clustering techniques to obtain adequate distributions, and so considerably reduces computational cost (see reference [23]).

generation. It can be seen that the distribution of solutions comprised by $J(\mathbf{\Theta}^*_{P\epsilon})$ along the front depends on objective exchange. The greatest number of points are accumulating in the central area (indicated by a dotted line) where the trade-off among objectives changes quickly. This property is equivalent to the smart filter used in the PIT criterion and therefore, helpful in the decision making process. The ϵ-dominance concept is helpful for avoiding a high density of solutions in the approximated Pareto front and brings useful solutions for the DM. Approaches using crowding measures (for example) seek to avoid high density areas, with no regard for PIT criterion.

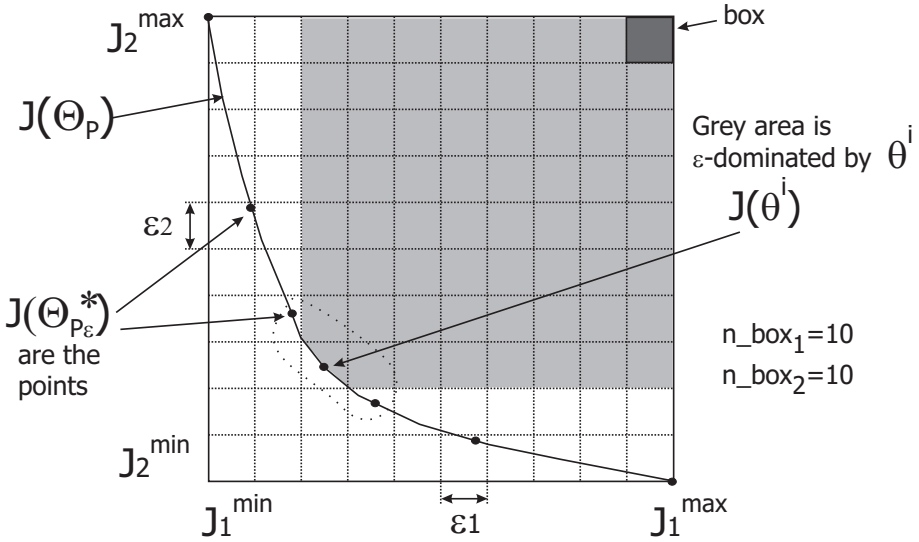

Figure 6. The concept of ϵ-dominance. ϵ-Pareto front $J(\mathbf{\Theta}^*_{P\epsilon})$ in a bi-objective problem. J_1^{min}, J_2^{min}, J_1^{max}, J_2^{max}, Pareto front limits; ϵ_1, ϵ_2 box widths; and n_box_1, n_box_2, number of boxes for each dimension.

A description of the ev-MOGA algorithm for obtaining an ϵ-Pareto front $\mathbf{J}(\mathbf{\Theta}^*_{P\epsilon})$, is presented below. The algorithm, which adjusts the width ϵ_i dynamically, is composed of three populations:

(1) Main population $P(t)$ explores the searching space D during the algorithm iterations (t). Population size is $Nind_P$.
(2) Archive $A(t)$ stores the solution $\mathbf{\Theta}^*_{P\epsilon}$. Its size $Nind_A$ is variable but bounded (see equation (24)).
(3) Auxiliary population $G(t)$. Its size is $Nind_G$, which must be an even number.

The pseudocode of the ev-MOGA algorithm is given by

```
1. t:=0
2. A(t):=∅
```

```
3. P(t):=ini_random(D)
4. eval(P(t))
5. A(t):=store_ini(P(t),A(t))
6. while t<t_max do
7.      G(t):=create(P(t),A(t))
8.      eval(G(t))
9.      A(t+1):=store(G(t),A(t))
10.     P(t+1):=update(G(t),P(t))
11.     t:=t+1
12. end while
```

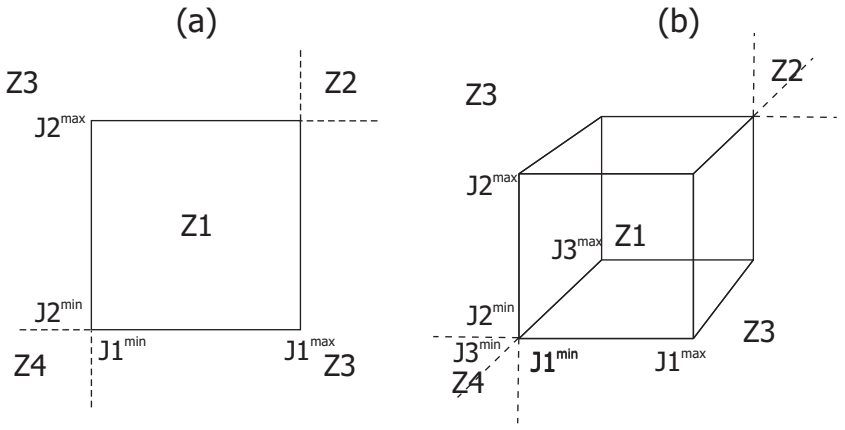

Figure 7. Function space areas (Z) and limits (J). (a) two-dimensional case; (b) tri-dimensional case.

Each line of the pseudocode is detailed as follows:

Line 1. Initialize termination condition (generation counter).

Line 2. Initialize archive $A(t)$

Line 3. $P(0)$ is initialized with $Nind_P$ individuals (solutions) that have been randomly selected from the searching space D.

Line 4. Function **eval** calculates the function value (Equation (1)) for each individual in $P(t)$.

Line 5. Function $\mathbf{store}_{ini}$ checks individuals in $P(t)$ that might be included in the archive $A(t)$ as follows:

(1) Non-dominated $P(t)$ individuals are detected, $\mathbf{\Theta}_{ND}$.
(2) Pareto front limits J_i^{max} and J_i^{min} are calculated from $\mathbf{J}(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \mathbf{\Theta}_{ND}$.
(3) Individuals in $\mathbf{\Theta}_{ND}$ are analyzed, one by one, and those that are not ϵ-dominated by individuals in $A(t)$, will be included in $A(t)$.

Line 6. The algorithm will execute while t<t_max.

Line 7. With each iteration, the function **create** creates $G(t)$ as follows:

(1) Two individuals are randomly selected, $\boldsymbol{\theta}^P$ from $P(t)$ and $\boldsymbol{\theta}^A$ from $A(t)$.
(2) A random number $u \in [0 \dots 1]$ is generated.
(3) If $u > P_{c/m}$ (probability of crossing/mutation), $\boldsymbol{\theta}^P$ and $\boldsymbol{\theta}^A$ are crossed over by means of the extended linear recombination technique. [24]
(4) If $u \leq P_{c/m}$, $\boldsymbol{\theta}^P$ and $\boldsymbol{\theta}^A$ are mutated using random mutation with Gaussian distribution [24] and then included in $G(t)$.

This procedure is repeated $Nind_G/2$ times until $G(t)$ is filled.

Line 8. Function **eval** calculates the function value (Equation (1)) for each individual in $G(t)$.

Line 9. Function **store** checks, one by one, which individuals in $G(t)$ must be included in $A(t)$ on the basis of their location in the objective space (see Figure 7). Thus $\forall\, \boldsymbol{\theta}^G \in G(t)$

(1) If $J(\boldsymbol{\theta}^G)$ belongs to the area $Z1$ and is not ϵ-dominated by any individual from $A(t)$, it will be included in $A(t)$ (if its box is occupied by an individual that is also not ϵ-dominated, then the individual lying furthest away from the center box will be eliminated). Individuals from $A(t)$ which are ϵ-dominated by $\boldsymbol{\theta}^G$ will be eliminated.
(2) If $J(\boldsymbol{\theta}^G)$ belongs to the area $Z2$ then it is not included in the archive, since it is dominated by all individuals in $A(t)$.
(3) If $J(\boldsymbol{\theta}^G)$ belongs to the area $Z3$, the same procedure is applied as was used with the function $\mathbf{store}_{ini}$ but now applied over a population $P'(t) = A(t) \bigcup \boldsymbol{\theta}^G$, that is, $\mathbf{store}_{ini}(P'(t), \emptyset)$. In this procedure, new Pareto front limits and ϵ_i widths could be recalculated.
(4) If $J(\boldsymbol{\theta}^G)$ belongs to the area $Z4$, all individuals from $A(t)$ are deleted since they are all ϵ-dominated by $\boldsymbol{\theta}^G$. $\boldsymbol{\theta}^G$ is included and the objective space limits are $\mathbf{J}(\boldsymbol{\theta}^G)$.

Line 10. Function **update** updates $P(t)$ with individuals from $G(t)$. Every individual $\boldsymbol{\theta}^G$ from $G(t)$ is compared with an individual $\boldsymbol{\theta}^P$ that is randomly selected from the individuals in $P(t)$ which are dominated by $\boldsymbol{\theta}^G$. $\boldsymbol{\theta}^G$ will not be included in $P(t)$ if there is no individual in $P(t)$ dominated by $\boldsymbol{\theta}^G$.

Line 11. Iteration counter t is incremented by one.

Line 12. Algorithm terminates. Individuals from $A(t)$ comprise $\boldsymbol{\Theta}^*_{P_\epsilon}$, the smart characterization of the Pareto front.

4. Three-bar truss example

The first optimization problem is related to the three-bar truss described in Figure 8. This truss is broadly used as a benchmark to define the best solutions based on certain specifications. The truss is statically indeterminate; thus the solution of the balance of forces has to be supplemented with the deformation equations. For this case, the parameters $L = 1m$, $\beta = 45^o$, $\alpha = 30^o$ and $F = 20kN$ proposed in [5,6]

were selected.

The design variables correspond to the sections of the bars $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$. The objectives correspond to the total volume of the truss ($J_2(\boldsymbol{\theta})$) and to a linear combination of the displacement of node P ($J_1(\boldsymbol{\theta})$).

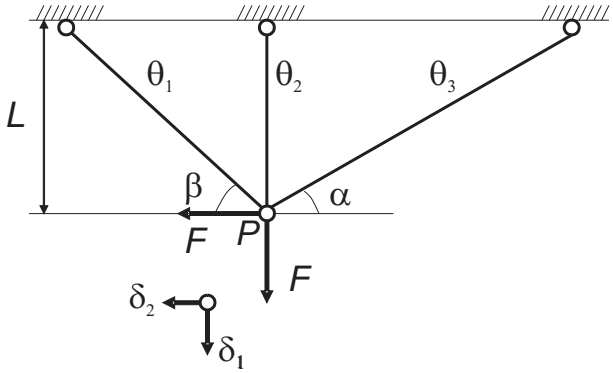

Figure 8. Three-bar truss problem with $\beta = 45^o$ and $\alpha = 30^o$.

The problem can be formulated as follows:

$$\min \mathbf{J}(\boldsymbol{\theta}) = [J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta})] \tag{25}$$

subject to

$$0.1 \cdot 10^{-4} m^2 \leq \theta_i \leq 2 \cdot 10^{-4} m^2, \; i = 1 \ldots 3,$$

where:

$$J_1(\boldsymbol{\theta}) = 0.25\delta_1 + 0.75\delta_2, \tag{26}$$

$$J_2(\boldsymbol{\theta}) = L\left(\frac{\theta_1}{\sin\beta} + \theta_2 + \frac{\theta_3}{\sin\alpha}\right). \tag{27}$$

Deformations δ_1 and δ_2 are calculated as [25]:

$$\begin{bmatrix}\delta_1 \\ \delta_2\end{bmatrix} = \frac{L}{E}\begin{bmatrix}\gamma_1 & \gamma_2 \\ \gamma_2 & \gamma_3\end{bmatrix}^{-1}\begin{bmatrix}F \\ F\end{bmatrix}, \tag{28}$$

where $E = 200GPa.$ is the *Young*'s modulus and

$$\gamma_1 = \theta_2 + \theta_1 \sin^3\beta + \theta_3 \sin^3\alpha,$$
$$\gamma_2 = -\theta_1 \sin^2\beta\cos\beta + \theta_3 \sin^2\alpha\cos\alpha,$$
$$\gamma_3 = \theta_1 \sin\beta\cos^2\beta + \theta_3 \sin\alpha\cos^2\alpha.$$

Moreover, the problem is subject to three constraints related to the reaction forces in each bar N_i:

$$\frac{|N_i|}{\theta_i} \leq \sigma, \; i = 1 \ldots 3, \tag{29}$$

with $\sigma = 200MPa$. These reaction forces are calculated according to the following expressions: [25]

$$N_1 = \frac{\theta_1 E}{L}(\delta_1 \sin\beta - \delta_2 \cos\beta)\sin\beta, \tag{30}$$

$$N_2 = \frac{\theta_2 E}{L}\delta_1, \tag{31}$$

$$N_3 = \frac{\theta_3 E}{L}(\delta_1 \sin\alpha + \delta_2 \cos\alpha)\sin\alpha. \tag{32}$$

The constraints (29) will be taken into account through static penalty functions. [26,27] [i] Therefore, the objective functions (26) and (27) result in:

$$J_1(\boldsymbol{\theta}) = 0.25\delta_1 + 0.75\delta_2 + C(\boldsymbol{\theta}), \tag{33}$$

$$J_2(\boldsymbol{\theta}) = L\left(\frac{\theta_1}{\sin\beta} + \theta_2 + \frac{\theta_3}{\sin\alpha}\right) + C(\boldsymbol{\theta}). \tag{34}$$

where:

$$C(\boldsymbol{\theta}) = \sum_{i=1}^{3} max\left[0,\ \frac{|N_i|}{\theta_i} - \sigma\right]. \tag{35}$$

To solve this optimization problem, the NNC with a smart filter and ev-MOGA algorithms are used and their results are compared to check their strengths and weaknesses.

The parameters of the ev-MOGA algorithm were set to:

- $Nind_G = 4$ and $Nind_P = 100$.
- $t_{max} = 4975$, resulting in 20000 evaluations of $J_1(\boldsymbol{\theta})$ and $J_2(\boldsymbol{\theta})$.
- $P_{c/m} = 0.1$.
- $n\_box_1 = n\_box_2 = 50$ so the maximum number of points in the Pareto front will be fewer than 52.

The parameters of the NNC algorithm and the smart filter were set to:

- $m_1 = 200$ in order to obtain a good density of points in the Pareto front.[j]
- $\Delta_m = 0.02$ and $\Delta_M = \infty$.

Δ_m was set to 0.02 with the intention of comparing the smart Pareto front and ϵ-Pareto front since $1/0.02 = 50$, which is the number of boxes the objective space is split into with ev-MOGA.

Figure 9 shows the results of the multi-objective optimization problem. Notice that the Pareto front is concave and disjointed. Both algorithms have captured the

[i] With this technique, the greater the non-fulfillment by a solution, the greater is the value of $C(\boldsymbol{\theta})$, and it will therefore be considered a worse solution; while if a solution fulfills all the constraints, then $C(\boldsymbol{\theta}) = 0$ and the equations (26) and (27) correspond to (33) and (34) respectively.

[j] For the method based on the NNC algorithm to give good results, the front must be characterized with a large number of uniformly distributed points.

anchor points perfectly and they have characterized the Pareto front with the same number of points (20 points) with a Smart distribution which is more or less the same. This proves that ev-MOGA and NNC Pareto front characterizations can be equivalent if Δ_m, Δ_M and n_box_i are set in an appropriate manner. The box limits are included in the figure to check the ϵ-dominance concept.

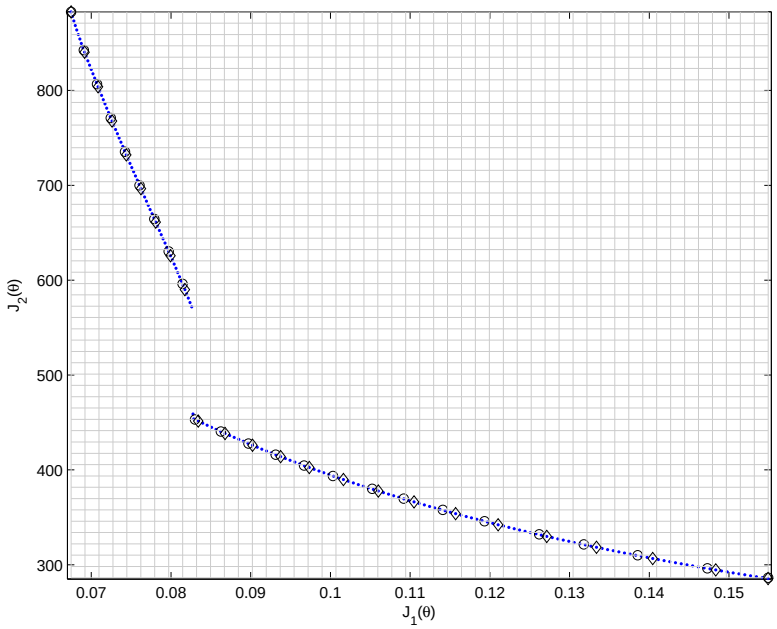

Figure 9. Three-bar truss example. '·' is the Pareto front obtained with NNC (it is composed of 200 evenly distributed points). 'o' is the ϵ-Pareto front obtained with ev-MOGA using $n\_box_1 = n\_box_2 = 50$. The horizontal and vertical lines represent the limit boxes. '◇' represents the smart Pareto front obtained with NNC results and $\Delta_m = 0.02$ and $\Delta_M = \infty$ are smart filter parameters.

The main advantage of the NNC algorithm over the ev-MOGA algorithm is its low computational burden, since it only needs about 5000 evaluations of the $J_1(\boldsymbol{\theta})$ and $J_2(\boldsymbol{\theta})$ functions to obtain the 200 points in the Pareto front; versus the 20000 required by the ev-MOGA algorithm. Conversely, determining the initial conditions of the optimizations addressed by the NNC is not so straightforward. For this particular example, each of the 200 optimizations was solved by strategically choosing its initial conditions so as to avoid local minimums.

5. Proportional-integral controller tuning example

This example is related to the proportional-integral (PI) controller tuning problem described in [28] by means of multi-objective optimization design. [29,30] The PI transfer

function used is:

$$G_c(s) = k_c \left(1 + \frac{1}{T_i s}\right) E(s) \tag{36}$$

where k_c (the proportional gain) and T_i (the integral time) are the design variables, $\boldsymbol{\theta} = [k_c, T_i]$. PI controllers are a reliable and practical control solution for industrial environments. They are widely used and any efforts to develop new tuning techniques are worthwhile. [31,32] This optimization procedure focuses on achieving a trade-off between load disturbance rejection, robustness, and setpoint response. It defines as a parameter for design a given value of the maximum sensitivity function

$$M_s = \max \left| \frac{1}{1 + G_c(\jmath\omega) G_p(\jmath\omega)} \right| \in [1.2, 2.0] \tag{37}$$

and the maximum complementary sensitivity function

$$M_p = \max \left| \frac{G_c(\jmath\omega)}{1 + G_c(\jmath\omega) G_p(\jmath\omega)} \right| \in [1.0, 1.5], \tag{38}$$

where $G_c(\jmath\omega), G_p(\jmath\omega)$ represents the controller and process transfer functions in the frequency domain, respectively. A numerical non-convex optimization is employed, by increasing as much as possible the integral gain $k_i = k_c/T_i$ subject to the pre-defined M_s and M_p values.

Therefore, a multiobjective optimization problem can be stated, where a trade-off between performance (integral gain, $J_1(\boldsymbol{\theta}) = -k_c/T_i$) and robustness ($J_2(\boldsymbol{\theta}) = M_s$, $J_3(\boldsymbol{\theta}) = M_p$) is formulated as:

$$\min \mathbf{J}(\boldsymbol{\theta}) = [J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta}), J_3(\boldsymbol{\theta})] \tag{39}$$

subject to

$$k_c + k_c/T_i \leq K_u, \tag{40}$$

$$1.2 \leq M_s \leq 2.0, \tag{41}$$

$$1.0 \leq M_p \leq 1.5. \tag{42}$$

$$0.0 \leq k_c \leq K_u. \tag{43}$$

$$0.01 \leq T_i \leq 20.0. \tag{44}$$

Constraint (40) is used to bound the maximum allowed control action effort to the ultimate gain K_u. Constraints (42) and (43) are used to obtain a Pareto front $\mathbf{J}^*_{\mathbf{P}}$ that is useful from the control point of view, while (43) and (44) determine the searching space.

The process transfer function to be used is:

$$G_p(s) = \frac{1}{(s+1)^3} \tag{45}$$

with $K_u \approx 7.8$.

The constraints (40, 41, 42) will be taken into account by using penalty functions again. [26] In this case, the problem is reformulated as follows:

$$\min_{\boldsymbol{\theta}\in\Re^2} \mathfrak{J}(\boldsymbol{\theta}) = \begin{cases} \mathbf{J}(\boldsymbol{\theta}) \quad \textbf{\textit{if}} \sum_{k=1}^{5} C_k(\boldsymbol{\theta}) = 0 \\ \textbf{\textit{offset}} + \left(\sum_{k=1}^{5} C_k(\boldsymbol{\theta}) \right) \cdot [1,1,1] \quad \textbf{\textit{otherwise}} \end{cases} \tag{46}$$

$$\textbf{\textit{offset}} = [0, 2.0, 1.5] \tag{47}$$

$$C_1(\boldsymbol{\theta}) = \max\{0, k_c + k_c/T_i - K_u\} \tag{48}$$
$$C_2(\boldsymbol{\theta}) = \max\{0, 1.2 - M_s\} \tag{49}$$
$$C_3(\boldsymbol{\theta}) = \max\{0, 1.0 - M_p\} \tag{50}$$
$$C_4(\boldsymbol{\theta}) = \max\{0, M_s - 2.0\} \tag{51}$$
$$C_5(\boldsymbol{\theta}) = \max\{0, M_p - 1.5\} \tag{52}$$

The parameters of the ev-MOGA algorithm were set to:

- $Nind_G = 16$ and $Nind_P = 160$.
- $t_{max} = 500$, resulting in 8160 evaluations of $J_1(\boldsymbol{\theta})$, $J_2(\boldsymbol{\theta})$ and $J_3(\boldsymbol{\theta})$.
- $P_{c/m} = 0.1$.
- $n\_box_1 = n\_box_2 = n\_box_3 = 50$ so the maximum number of points in the Pareto front will be fewer than 2602.

The parameters of the NNC algorithm and the smart filter were set to:

- $m_1 = 200$ in order to obtain a good density of points in the Pareto front.
- $\Delta_m = 1/50 = 0.02$ and $\Delta_M = \infty$.

Figure 10 shows the results of the multi-objective optimization problem obtained with NNC and ev-MOGA algorithms.

In this example, the solution that minimizes the objective $J_3(\boldsymbol{\theta})$ and $J_2(\boldsymbol{\theta})$ is the same. Therefore, there are only two anchor points and the utopia hyperplane is reduced to a line - which in the case of NNC leads to fewer solutions in the central area of the Pareto front than with ev-MOGA. This prevents the NNC from characterizing the surface of the Pareto front in the central area.

When $J_3(\boldsymbol{\theta})$ is minimized in order to obtain the anchor points, so that there are several solutions such as $J_3(\boldsymbol{\theta}) = M_p = 1.0$ (J_3 is multimodal). There is no guarantee that the NNC algorithm will obtain the most useful J_3 anchor.

To evaluate the performance of each MOEA, the hypervolume (or Lebesgue measure) computed by means of a Monte-Carlo approximation method has been

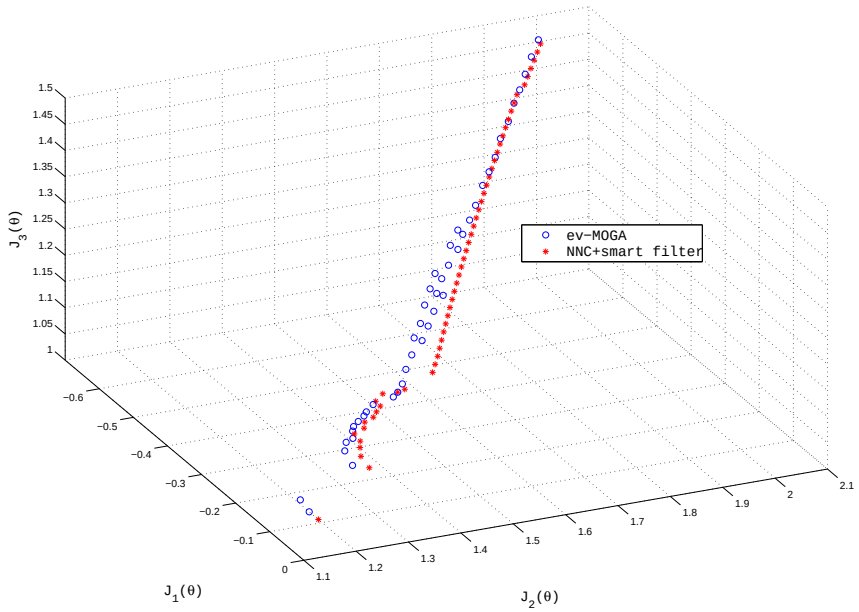

Figure 10. PI design example. '$\circ$' is the Pareto front obtained with NNC. '$*$' is the ϵ-Pareto front obtained with ev-MOGA.

obtained and the results are:[k]

- NNC=0.1563
- ev-MOGA=0.1676

That is, ev-MOGA improves the hypervolume indicator by 7.2% in comparison with NNC with a smart filter.

Both hypervolume and qualitative inspection of the Pareto front show that ev-MOGA algorithms can characterize the Pareto front better than NNC, mainly due to the geometry and shape of this Pareto front and the problems previously described for NNC.

6. Conclusions

A multi-objective evolutionary algorithm, ev-MOGA, based on the concept of ϵ-dominance has been presented to characterize the Pareto front in a smart way and compare it with the NNC with the smart filter method. To evaluate the performance

[k]Hypervolume was computed taking $[0, 2, 1.5]$ as a reference point and 100,000 as the number of samples used for the Monte-Carlo approximation. The Matlab function used is available at www.mathworks.com/matlabcentral/fileexchange/19651.

of these algorithms, two optimization problems were utilized.

Some of main conclusions are:

- The NNC method generates evenly distributed Pareto fronts but:
  (1) The solution is dependent on the initial optimization conditions since it uses a search-based Gauss-Newton method which can cause some local Pareto points to be obtained.
  (2) May have difficulties properly characterizing the Pareto front when two or more anchor points are the same (in three or more objective problems).
  (3) Something similar could happen when an objective function is multimodal since the anchor points cannot correspond to the end of the Pareto front (in three or more objective problems).
  (4) With an *a priori* knowledge of the Pareto front geometry it is possible to improve the NNC algorithm to overcome the latter difficulties. Nevertheless, such information is not always available.
  (5) The computational burden grows exponentially with respect to the dimension of the objective function space since the transformed optimization problem to be executed also grows exponentially if the same density of Pareto points is required.
- The MNNC eliminates the first NNC disadvantage, but the second disadvantage is increased considerably.
- A smart filter based on PIT is a very effective and flexible procedure to obtain smart Pareto fronts, but the result depends on the order in which the analysis of the Pareto points is carried out. To reduce this problem, it is very important that the NNC method characterizes the Pareto front with many points, which again increases the computational burden.
- ev-MOGA algorithm eliminates the first NNC disadvantage. Its computational burden is also more competitive than that of MNNC, thanks to the fact that the Pareto points are generated in parallel and in a single run. Other features of ev-MOGA are:
  (1) It dynamically adjusts the precision of the Pareto front without increasing the archive size, so that the memory requirements are always bounded (n_box_i parameters).
  (2) It adapts the extremes of the Pareto front, regardless of the parameters n_box_i and ensures that anchor points are not eliminated from the archive. At the same time this eliminates the second NNC disadvantage.
  (3) It automatically characterizes all kinds of Pareto fronts (i.e. non-convex and disjoined ones) in a smart way in a similar manner to NNC with smart filter methods if $\Delta_M = \infty$.
  (4) It is an algorithm useful for the designer, since it approximates the Pareto front (search process) with significant solutions for the DM (selection step).

Acknowledgments

This work was partially supported by the FPI-2010/19 grant and the PAID-06-11 project from the Universitat Politècnica de València, projects TIN2011-28082 and ENE2011-25900 (Spanish Ministry of Economy and Competitiveness) and the GV/2012/073 (Generalitat Valenciana).

Bibliography

1. Coello, C. A. C., Veldhuizen, D. V., Lamont, G., 2002. Evolutionary algorithms for solving multi-objective problems. Kluwer Academic press.
2. Marler, R. and Arora, Jasbir, 2010. The weighted sum method for multi-objective optimization: new insights,Structural and Multidisciplinary Optimization, 41 (6), 853 – 862
3. Messac A, Mattson CA. Generating well-distributed sets of pareto points for engineering design using physical programming. Optimization and Engineering 2002; 3:431-450.
4. Miettinen, K. M., 1998. Nonlinear multiobjective optimization. Kluwer Academic Publishers.
5. Messac A, Ismail-Yahaya A, Mattson CA. The normalized normal constraint method for generating the Pareto frontier. Structural and Multidisciplinary Optimization 2003; 25:86-98.
6. Martínez M, Blasco X, Sanchis J. Global and well-distributed Pareto frontier by modified normalized constraint methods. Structural Multidisciplinary Optimization 2007; 34:197-207.
7. Martínez M, García-Nieto S, Sanchis J, Blasco X. Genetic algorithms optimization for normalized normal constraint method under Pareto construction. Advances in Engineering Software 2009, 40(4): 260-267.
8. Mattson, C. A., Messac, A., 2005. Pareto frontier based concept selection under uncertainty, with visualization. Optimization and Engineering 6, 85–115, 10.1023/B:OPTE.0000048538.35456.45.
9. Mattson CA, Muller AA, Messac A. Smart Pareto Filter: Obtaining a minimal representation of multi-objective design space. Engineering Optimization 2004; 36(6):721-740.
10. Coello Coello, C., feb. 2006. Evolutionary multi-objective optimization: A historical view of the field. Computational Intelligence Magazine, IEEE 1 (1), 28 – 36.
11. Coello, C., 2011. An introduction to multi-objective particle swarm optimizers. In: Gaspar-Cunha, A., Takahashi, R., Schaefer, G., Costa, L. (Eds.), Soft Computing in Industrial Applications. Vol. 96 of Advances in Intelligent and Soft Computing. Springer Berlin / Heidelberg, pp. 3–12, $10.1007/978-3-642-20505-7_1$.
12. Mezura-Montes, E., Reyes-Sierra, M., Coello, C., 2008. Multi-objective optimization using differential evolution: A survey of the state-of-the-art. Advances in Differential Evolution (SCI 143), 173 – 196.
13. Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., Zhang, Q., 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation 1 (1), 32 – 49.
14. Coello, C. A. C., Lamont, G. B., 2004. Applications of Multi-Objective evolutionary algorithms, advances in natural computation vol. 1 Edition. World scientific publishing.
15. Zitzler E. Evolutionary algorithms for multi-objective optimization: Methods and ap-

plications, Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 1999.
16. Coello C, Veldhuizen D, Lamont G. Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, 2002.
17. Coello C, Toscano G, Mezura E. Current and future research trends in evolutionary multi-objective optimization. In: Manuel Graña, Richard Duro, Alicia dÁnjou, and Paul P. Wang, editors. Information Processing with Evolutionary Algorithms: From Industrial Applications to Academic Speculations. Springer-Verlag 2005; 213-231.
18. Alander J. An indexed bibliography of Genetic Algorithms & Pareto and constrained optimization. Technical report, Department of Information Technology, University of Vaasa, 2002.
19. Garrett D., Dasgupta D., Vannucci J. and Simien J. Applying Hybrid Multiobjective Evolutionary Algorithms to the Sailor Assignment Problem. book chapter in Jain L. C., Palade V., Srinivasan D. (Eds.): Advances in Evolutionary Computing for System Design, Springer 2007.
20. Wang Z. and Palade V. Multi-Objective Evolutionary Algorithms based Interpretable Fuzzy Models for Microarray Gene Expression Data Analysis. Proc. of the 2010 IEEE International Conference on Bioinformatics and Biomedicine - BIBM 2010, Hong Kong - China, Dec 2010, pages 308-313.
21. Bonissone, P., Subbu, R., Lizzi, J., aug. 2009. Multicriteria decision making (mcdm): a framework for research and applications. Computational Intelligence Magazine, IEEE 4 (3), 48 –61.
22. Herrero JM. Non-linear Robust identification using evolutionary algorithms, Ph.D. Thesis, Polytechnic University of Valencia, 2006.
23. Laumanns M, Thiele L, Deb K, Zitzler E. Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary computation 2002; 10(3):263-282.
24. Herrero JM, Blasco X, Martínez M, Sanchis J: Robust identification of a biomedical process by evolutionary algorithms (in Spanish). Revista Iberoamericana de Automática e Informática Industria 2006; 3(4):75-86
25. Batill SM. Course: ME/AE 446. Finite Element Methods in Structural Analysis, Planar truss applications 1995. www.nd.edu.
26. Coello C: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 2002; 191:1245-1287.
27. Mezura-Montes, E., Coello, C. A. C., December 2011. Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm and Evolutionary Computation 1 (4), 173 – 194.
28. Åström, K.J.: Design of PI Controllers based on Non-Convex Optimization. Automatica 1998; 34(5):585-601.
29. Reynoso-Meza, G., Sanchis, J., Blasco, X., Herrero, J., sept. 2011. Handling control engineer preferences: Getting the most of PI controllers. In: Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on. pp. 1 –8.
30. Reynoso-Meza, G., Sanchis, J., Blasco, X., Juan M., H., 2012. Multiobjective evolutionary algortihms for multivariable PI controller tuning. Expert Systems with Applications Volume 39, Issue 9, July 2012, Pages 7895–7907.
31. Vilanova, R., Alfaro, V. M., 2011. Robust PID control: an overview (in spanish). Revista Iberoamericana de Automática e Informática Industrial 8 (3), 141-158.
32. Reynoso-Meza, G., Blasco, X., Sanchis, J., Martínez, M., 2013. Evolutionary algorithms for PID controller tuning: Current trends and perspectives (in spanish). Revista Iberoamericana de Automática e Informática Industrial 10 (3), 251-268.