Hindawi
Complexity
Volume 2018, Article ID 1792420, 22 pages
https://doi.org/10.1155/2018/1792420

Research Article

A Multiobjective Genetic Algorithm for the Localization of Optimal and Nearly Optimal Solutions Which Are Potentially Useful: nevMOGA

Alberto Pajares,[1] **Xavier Blasco**,[1] **Juan M. Herrero**,[1] **and Gilberto Reynoso-Meza**[2]

[1]*Instituto Universitario de Automática e Informática Industrial, Universitat Politecnica de Valencia, Valencia, Spain*
[2]*Industrial and Systems Engineering Graduate Program-PPGEPS, Polytechnic School, Pontifical Catholic University of Paraná (PUCPR), Curitiba, PR, Brazil*

Correspondence should be addressed to Alberto Pajares; alpafer1@upv.es

Received 26 February 2018; Accepted 2 August 2018; Published 2 October 2018

Academic Editor: Guido Caldarelli

Traditionally, in a multiobjective optimization problem, the aim is to find the set of optimal solutions, the Pareto front, which provides the decision-maker with a better understanding of the problem. This results in a more knowledgeable decision. However, multimodal solutions and nearly optimal solutions are ignored, although their consideration may be useful for the decision-maker. In particular, there are some of these solutions which we consider specially interesting, namely, the ones that have distinct characteristics from those which dominate them (i.e., the solutions that are not dominated in their neighborhood). We call these solutions *potentially useful* solutions. In this work, a new genetic algorithm called nevMOGA is presented, which provides not only the optimal solutions but also the multimodal and nearly optimal solutions nondominated in their neighborhood. This means that nevMOGA is able to supply additional and potentially useful solutions for the decision-making stage. This is its main advantage. In order to assess its performance, nevMOGA is tested on two benchmarks and compared with two other optimization algorithms (random and exhaustive searches). Finally, as an example of application, nevMOGA is used in an engineering problem to optimally adjust the parameters of two PI controllers that operate a plant.

1. Introduction

In industry, there are many situations where a design problem turns into an optimization problem. Moreover, it is usual that these problems have conflicting objectives, which generates a multiobjective optimization problem (MOP) ([1–3]). A MOP basically consists of three stages [4]: its definition, the optimization problem (search), and the multicriteria decision-making (MCDM). In an a priori multiobjective approach [5], the stages of optimization and decision-making are carried out at the same time, which results in a single solution, since the desired preferences are defined beforehand. On the contrary, in an a posteriori approach [6], the search does not supply a single solution but a set of optimal solutions (Pareto optimal solutions). This procedure is more time-consuming, but it gives the designer information about different solutions and provides them with a better understanding of the problem, which will allow them to make a well-informed decision in the decision-making stage. On the other hand, an optimization problem may have multimodal solutions ([7–9]) or nearly optimal solutions ([10, 11]) that are useful for the designer. These alternative solutions, which have a similar or even the same performance as the optimal solutions, are ignored in a traditional multiobjective approach, although they offer interesting and useful information to the designer. In order not to lose this valuable information, there must be taken into account not only the optimal solutions but also the nearly optimal ones ([12]).

Including the multimodal and nearly optimal solutions in the decision-making scenario increases the number of alternatives available to the designer. However, this poses two issues. On the one hand, the number of solutions will increase considerably and this may slow the algorithm to the point of becoming inoperative. On the other hand, having

a huge amount of alternatives to choose makes the decision-making process more complicated. For this reason, it is important in most cases to reduce, in some way, the number of solutions. The way in which this reduction is performed will depend on the criterion or approach that the designer decides to adopt, which, in turn, depends on the application or the context. The existing algorithms usually perform this reduction by means of a discretization of the objective space. But this discretization is carried out without taking into account any consideration of the location of the solutions in the parameter space. The result of proceeding in this way is that nearly optimal solutions which have distinct characteristics could be neglected. Despite this drawback, this method is suitable for some applications.

One possible approach, however, and the one that we assume here, is the following: (1) finding a manageable number of solutions (so that the two issues mentioned above can be avoided) but, at the same time, (2) without neglecting the existing diversity in the characteristics of the nearly optimal solutions. We think that this approach is common in many applications and the existing algorithms do not serve in this case, precisely because they do not pay attention to the characteristics of the solutions. This is why, in this work, we have developed an algorithm to provide the designer with those solutions, among all the nearly optimal solutions, that fulfill that particular criterion, namely, the nondominated in their neighborhood. These solutions are the information that our designer finds more relevant. For example, a designer adopting this approach will not want to be given two nearly optimal solutions which have similar characteristics (neighboring solutions), if one of them is dominated by the other (i.e., worse for at least one of the objectives and not better for the rest), as he or she will logically choose the nondominated one. However, if these two solutions have significantly different characteristics (i.e., they are nonneighboring solutions), then both of them will be interesting for the designer. If the designer has these solutions available, they can analyze them a posteriori (for example, by including new indicators or by considering the physical sense of the solutions), in order to decide which one is the most suitable. From now on, we will call these solutions *potentially useful* solutions. This does not mean that the rest of nearly optimal solutions are useless, but simply that they are dispensable for a designer who has adopted the approach specified before. These potentially useful solutions are the ones that according to his or her criterion gives them the most relevant information, the information that they want to get.

An example where the specified approach could be the chosen one and, therefore, it is desirable to find the nearly optimal solutions nondominated in their neighborhood, is when one of the objectives cannot be included in the optimization process due to its high computational cost. In this case, it is possible to exclude that objective from the optimization process and to obtain the set of optimal solutions for the rest of objectives. When the optimization process has finished, the excluded objective can be evaluated (only at the optimal solutions previously obtained) and incorporated into the decision-making process. The problem here (if a classical MOP approach is followed) is that, since the excluded objective is ignored in the search, there will be interesting solutions (those with a good performance for the excluded objective and with similar performance for the rest of objectives) which will be missed. The new approach that we are proposing in this paper finds them. Another situation where it may also be interesting to examine nearly optimal solutions nondominated in their neighborhood happens when several objectives are aggregated, which is a way of defining a priori preferences. When a MOP has a lot of objectives, the decision-making stage becomes too complicated. That is why sometimes several objectives are aggregated in groups (for example, performance, cost, and robustness), in order to simplify the problem. However, this approach could miss valuable solutions for the designer. This loss of information can largely be avoided by finding the nearly optimal solutions nondominated in their neighborhood.

So, finding the nearly optimal solutions nondominated in their neighborhood solves the lack of information that typically appears in a classical multiobjective approach. Moreover, taking into account these solutions has two additional advantages: (1) it enables the designer to reassess the design objectives—if there are nearly optimal solutions which display certain characteristics that are absent in the optimal ones, this could mean a poor formulation of the objectives—and (2) it detects the existence of overparameterization (i.e., when the number of parameters is higher than needed, which could result in a man-made multimodality).

In this work, a new algorithm is presented, nevMOGA, which is aimed at finding the sets of optimal solutions and nearly optimal solutions nondominated in their neighborhood. There exist several algorithms in the literature that consider nearly optimal solutions as well ([13, 14]). However, they use these solutions simply as a means to compute an approximation of the Pareto front, whereas nevMOGA discriminates them and returns some of them (the nondominated in their neighborhood) as its outcome. nevMOGA is based on an algorithm called evMOGA, which is described in [15]. This new algorithm includes additional parameters to define when a solution is considered to be a nearly optimal solution and the size of a neighborhood.

2. Background

In this section, the concepts of multiobjective optimization problem and Pareto set are formally introduced, and the problem which we are dealing with is graphically explained (i.e., the problem of finding optimal and nearly optimal solutions nondominated in their neighborhood).

A multiobjective optimization problem (A maximization problem can be converted into a minimization one. For each of the objectives that have to be maximized, the transformation: $\max f_i(\boldsymbol{x}) = -\min(-f_i(\boldsymbol{x}))$ can be applied.) can be stated as follows:

$$\begin{aligned} &\min_{\mathbf{x}\in Q} && f(x) \\ &\text{subject to} && \underline{x_i} \le x_i \le \overline{x_i}, \quad i = [1, \ldots, k], \end{aligned} \tag{1}$$

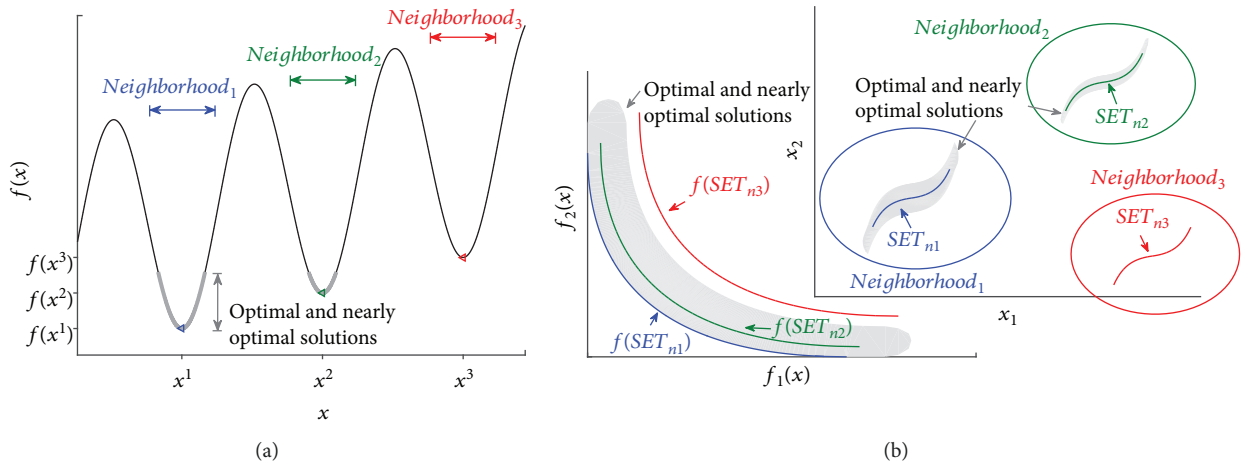


FIGURE 1: A monoobjective example (a) and a multiobjective example (b). In the monoobjective problem, the optimal solution is x^1 (blue triangle) and the nearly optimal solution nondominated in its neighborhood is x^2 (green triangle). In the multiobjective problem, the optimal solutions are the ones in set SET_{n1} (blue front) and the nearly optimal solutions nondominated in their neighborhood are the ones in SET_{n2} (green front).

where $x = [x_1, \dots, x_k]$ is defined as a decision vector in the domain $Q \subset \Re^k$ and f: $Q \to R^m$ is defined as the vector of objective functions $f(x) = [f_1(x), \dots, f_m(x)]$. $\underline{x_i}$ and $\overline{x_i}$ are the lower and upper bounds of each component of x.

Definition 1 (dominance [16]). A decision vector x^1 is dominated by any other decision vector x^2 if $f_i(x^2) \leq f_i(x^1)$ for all $i \in [1, \dots, m]$ and $f_j(x^2) < f_j(x^1)$ for at least one j, $j \in [1, \dots, m]$. This is denoted as $x^2 \preceq x^1$.

Definition 2 (Pareto set). The Pareto set (denoted by P_Q) is the set of solutions in Q which are not dominated by another solution in Q:

$$P_Q := \left\{ x \in Q \mid \nexists x' \in Q : x' \preceq x \right\}. \tag{2}$$

Definition 3 (Pareto front). Given a set of Pareto optimal solutions P_Q, the Pareto front $f(P_Q)$ is defined as

$$f := \left\{ f(x) \mid x \in P_Q \right\}. \tag{3}$$

Figure 1 shows an example to clarify what we mean by optimal solutions, nearly optimal solutions, and nearly optimal solutions nondominated in their neighborhood in a monoobjective (Figure 1(a)) and a multiobjective (Figure 1(b)) problem. Let us consider the monoobjective case. In theory, all the nearly optimal solutions (grey lines) could be interesting for the decision-maker. However, considering all these solutions makes the decision-making stage too difficult. Moreover, many of these solutions (neighborhood_1) will have very similar characteristics to the optimal one (x^1), since they are close to it in the parameter space. This, in addition to the fact that they have a worse performance than the optimal one, leads us to consider them, under the criterion assumed, less relevant and in some cases dispensable. The rest of them (neighborhood_2) will have similar characteristics to x^2, which is the best in nei ghborhood$_2$ and, therefore (always under the specified criterion), they could be dispensable. Consequently, the solutions which provide more relevant information (potentially useful solutions) are x^1 and x^2, i.e., the optimal solution and the nearly optimal solution nondominated in its neighborhood. Likewise, in the multiobjective case, all the nearly optimal solutions (grey area) could be considered but, again, taking into account all of them would significantly complicate the decision-making stage while most of these solutions do not add much relevant information to the process. So, the sets SET_{n1} (optimal solutions) and SET_{n2} (nearly optimal solutions nondominated in their neighborhood) provide the most valuable information to the designer without unnecessarily complicating the decision-making process.

An example of the usefulness of finding the nearly optimal solutions nondominated in their neighborhood is when one of the objectives *cannot* be included in the optimization process due to its high computational cost. The usual way to cope with this difficulty consists of solving the MOP for the rest of objectives and then computing the excluded one only for the optimal solutions. The problem with that (if a classical MOP approach is followed) is that there will be nearly optimal solutions with an outstanding performance with respect to the excluded objective (and therefore worth considering) that will be ignored. On the other hand, if one is not willing to miss all those interesting solutions and decides to take into account all the nearly optimal solutions, then the time-consuming objective will be evaluated at a vast number of solutions, many of which are not even worth considering, as they will have similar characteristics to the optimal solutions (because they are neighboring solutions). In summary, approaching this kind of problems with a classical strategy leads to either neglecting potentially useful solutions or having to evaluate a time-expensive objective at many dispensable solutions, whose inclusion in the decision-making stage will, furthermore, unnecessarily complicate the

Table 1: Analysis of a MOP example with three objectives ($f_1(x)$ to $f_3(x)$) and five possible solutions (x^1 to x^5).

Solutions	$f_1(x)$	$f_2(x)$	$f_3(x)$
x^1 =[0 1]	0.2	0.2	0.2
x^2 =[0.5 0.5]	0.2	0.2	0.2
x^3 =[0.75 0.75]	0.201	0.201	0.19
x^4 =[0 0.99]	0.201	0.2	0.199
x^5 =[0.25 0.75]	1	1	0.15

Table 2: Analysis of a MOP example with four objectives ($f_{11}(x)$, $f_{12}(x)$, $f_{21}(x)$, and $f_{22}(x)$) and four possible solutions (x^1 to x^4) where the MOP is reduced by aggregating f_{11} and f_{12} ($f_1 = f_{11} + f_{12}$) and f_{21} and f_{22} ($f_2 = f_{21} + f_{22}$).

Solutions	$f_{11}(x)$	$f_{12}(x)$	$f_1(x)$	$f_{21}(x)$	$f_{22}(x)$	$f_2(x)$
x^1 =[0 0]	0.2	0.2	0.4	0.2	0.2	0.4
x^2 =[0 1]	0.18	0.22	0.4	0.24	0.16	0.4
x^3 =[1 0]	0.179	0.221	0.4	0.241	0.161	0.402
x^4 =[1 1]	0.9	0.199	1.119	0.9	0.9	1.8

decision. With our new approach, these problems can be overcome. Let us look at it with a concrete example.

Let us suppose a MOP with three objectives $f(x) = [f_1(x)\ f_2(x)\ f_3(x)]$ to minimize. Table 1 shows five possible solutions to the problem (x^1 to x^5). In a classical MOP, all these solutions belong to the Pareto set (except either x^1 or x^2, as they are multimodal solutions and, therefore, one of them would be discarded). Now, assume that the objective f_3 is computationally impracticable, i.e., its incorporation into the optimization process is impossible. For this reason, the search stage has to be solved for the first two objectives ($f(x) = [f_1(x)\ f_2(x)]$) and then f_3 will be evaluated only at the solutions previously obtained. First, note that in a classical MOP, only the Pareto front is obtained, so either x^1 or x^2 (multimodal solutions) and x^3 would be discarded, although x^1, x^2, and x^3 seem to be good solutions, since they are nearly optimal solutions in different neighborhoods and thus worth considering.

With our new approach, x^1, x^2, and x^3 would also be found, because they are nondominated in their neighborhood; in other words, they have significantly different parameter values. x^4 would be discarded since it is dominated by x^1 and they are neighboring solutions, i.e., they are expected to have similar characteristic. x^4 has a better performance in f_3 than x^1; however, this improvement is insignificant (if the neighborhoods are well defined) as they are very similar solutions (neighboring solutions). Through this new procedure, from all the neighboring solutions, only those with the best performance over the objectives considered in the optimization process (in this example, f_1 and f_2) would be selected. The solution x^3 displays a better value for f_3 than x^1 and x^2 (which are in different neighborhoods) but worse values for f_1 and f_2 (this is why it is a nearly optimal solution) and, therefore, in a classical MOP, it would be discarded, although its analysis may be interesting for the decision-maker. Logically, x^5 would be ruled out since it is much worse for f_1 and f_2, although it has a better value for f_3. With the proposed approach, f_3 is only evaluated at x^1, x^2, and x^3 but not at x^4 (x^1 is preferred instead) and x^5 (which is discarded, since it is not a nearly optimal solution) and, consequently, the computational burden is alleviated. Therefore, thanks to this new approach; it is possible to reduce the computational cost while obtaining the potentially useful solutions for the designer.

Another example that shows the usefulness of obtaining also the nearly optimal solutions nondominated in their neighborhood can be found when several objectives are aggregated, which is a common procedure when there are a great number of them. Let us suppose that a MOP originally has four objectives $f(x) = [f_{11}(x)\ f_{12}(x)\ f_{21}(x)\ f_{22}(x)]$ to minimize. Let be x^1 to x^4 (Table 2) the four solutions to this MOP. In a classical MOP, all these solutions would be selected to form the Pareto set.

If the designer wanted to reduce the number of objectives, a possibility would be to add f_{11} and f_{12} to obtain f_1 ($f_1 = f_{11} + f_{12}$) and f_{21} and f_{22} to obtain f_2 ($f_2 = f_{21} + f_{22}$). But now, only x^1 or x^2 (multimodal solutions) would be the optimal solution (in a classical MOP). Let us suppose that the algorithm chooses x^1 and discards x^2. In this case, x^2 and x^3 are discarded, although they are interesting solutions for the designer. However, if the algorithm keeps also the nearly optimal solutions nondominated in their neighborhood, x^2 and x^3 would not be excluded with the aggregated objectives. Finally, x^4 presents a poor performance for both f_1 and f_2 (clearly dominated) and, therefore, it does not seem to be a good solution.

In conclusion, when the objectives are aggregated, a classical MOP finds only one solution from the three that are interesting and the other two are missed. Therefore, finding the nearly optimal solutions nondominated in their neighborhood allows the designer to aggregate objectives (and, in this way, simplifying the decision-making stage) without losing potentially useful solution.

In summary, as seen in the examples, the multimodal solutions and the nearly optimal solutions which are significantly different from those which dominate them are solutions that can be potentially useful for the designer during the decision-making stage. This additional information is not provided to the decision-maker when these kinds of MOPs are approached in a classical manner.

3. Materials and Methods

In this section, we present in detail a novel algorithm, nevMOGA, which is the main contribution of this work, as well as the metric and benchmarks that have been used to demonstrate its correct functioning. Let us begin with the new set of interest and its discretization, which is performed by nevMOGA.

3.1. New Set of Interest. The nevMOGA algorithm is aimed at finding not only the optimal solutions but also the nearly optimal solutions nondominated in their neighborhood. In the following, both sets of solutions are defined.

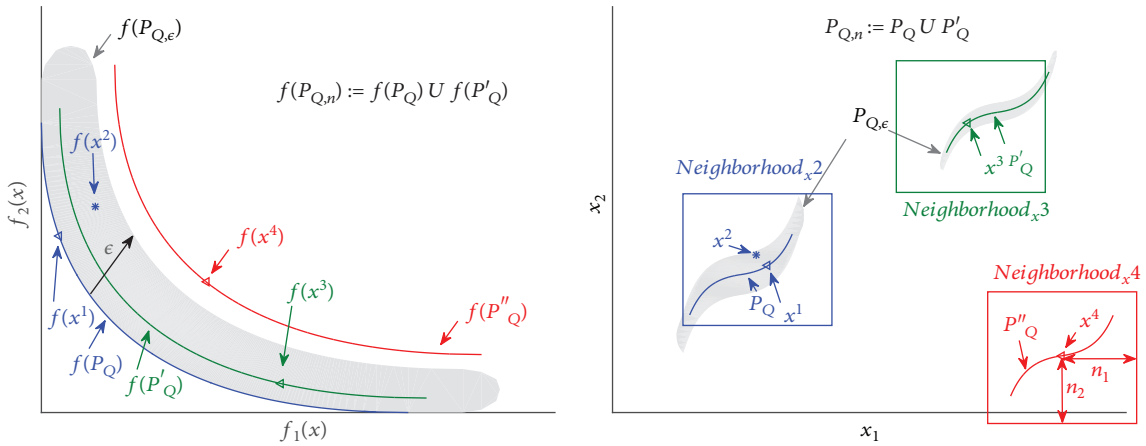


FIGURE 2: Graphical example. The set of interest is $P_{Q,n}$. The set of optimal solutions is P_Q, and the set of nearly optimal solutions nondominated in their neighborhood is P'_Q. x^2 is discarded since it is n dominated (i.e., dominated by a neighboring solution) by x^1 (neighborhood$_{x^2}$). The solutions in P''_Q are discarded since their performances fall out of the area of acceptable performance (grey area).

Definition 4 ($-\epsilon$-dominance [17]). Define $\epsilon = [\epsilon_1, \ldots, \epsilon_m]$ as the maximum acceptable performance degradation. A decision vector x^1 is $-\epsilon$-dominated by another decision vector x^2 if $f_i(x^2) + \epsilon_i \leq f_i(x^1)$ for all $i \in [1, \ldots, m]$ and $f_j(x^2) + \epsilon_i < f_j(x^1)$ for at least one j, $j \in [1, \ldots, m]$. This is denoted by $x^2 \preceq_{-\epsilon} x^1$.

Definition 5 ($-\epsilon$-Pareto set [18]). The $-\epsilon$-Pareto set (denoted by $P_{Q,\epsilon}$) is the set of solutions in Q which are not $-\epsilon$-dominated by another solution in Q:

$$P_{Q,\epsilon} := \left\{ x \in Q \mid \nexists x' \in Q : x' \preceq_{-\epsilon} x \right\}. \tag{4}$$

Definition 6 (neighborhood). Define $n = [n_1, \ldots, n_k]$ as the maximum distance between neighboring solutions. Two decision vectors x^1 and x^2 are neighboring solutions ($x^1 =_n x^2$) if $|x_i^1 - x_i^2| < n_i$ for all $i \in [1, \ldots, k]$.

Definition 7 (n-dominance). A decision vector x^1 is n-dominated by another decision vector x^2 if they are neighboring solutions (Definition 6) and $x^2 \preceq x^1$. This is denoted by $x^2 \preceq_n x^1$.

Definition 8 (n-Pareto set). The n-Pareto set (denoted by $P_{Q,n}$) is the set of solutions of $P_{Q,\epsilon}$ which are not n-dominated by another solution in $P_{Q,\epsilon}$:

$$P_{Q,n} := \left\{ x \in P_{Q,\epsilon} \mid \nexists x' \in P_{Q,\epsilon} : x' \preceq_n x \right\}. \tag{5}$$

The set $P_{Q,n}$ (optimal solutions and nearly optimal solutions nondominated in their neighborhood) is the new set of interest, i.e., what nevMOGA has to find. In order to clarify its nature, we will use a graphical example (see Figure 2). In this example, the set of interest is formed by the union of P_Q and P'_Q. P'_Q is a set of solutions nondominated in their neighborhood which belong to the set $P_{Q,\epsilon}$. P''_Q is a set of solutions nondominated in their neighborhood which do not belong to the set $P_{Q,\epsilon}$. x^4 is not a nearly optimal solution, and, therefore, it will be discarded since it does not have the desirable performance ($P_{Q,\epsilon}$). x^2 is n dominated (i.e., dominated by a neighboring solution) by x^1 and, as a result, this alternative provides less relevant information and, for this reason, is discarded. x^3 is dominated by nonneighboring solutions but not by neighboring solutions, so it belongs to $P_{Q,n}$, and the same can be said of any solution in P'_Q. x^1 belongs to the optimal set P_Q. In summary, nevMOGA searches for the set $P_{Q,n}$, in our example, $P_Q \cup P'_Q$.

The set $P_{Q,n}$ clearly depends on the values given to the new parameters ϵ (Definition 4) and n (Definition 6). Given that in most problems, the objectives have a physical sense, it is possible to decide a priori how much performance we are willing to lose and to define ϵ consequently. Similarly, when the decision variables x have a physical sense, the parameter n can also be set intuitively. By setting n, the designer establishes to what extent the two solutions are considered "similar"; in other words, setting n defines the neighborhood.

If the decision variables do not have a physical sense and it is not evident how to choose n, it is still possible to follow a simple procedure (see Figure 3) which will help us to do it if ϵ has been previously set. First, a reference solution x^R is chosen. Second, in the objective space, a rectangle is defined with the center at $f(x^R)$ and twice ϵ in width and in height. Then, starting from x^R (in the search space), the value of each variable decision is increased and decreased independently, until the solution leaves the rectangle which was previously defined. Finally, each element of the vector neighborhood n is set to the minimum distance between ϵ and the first solutions (one for each direction) which left the mentioned

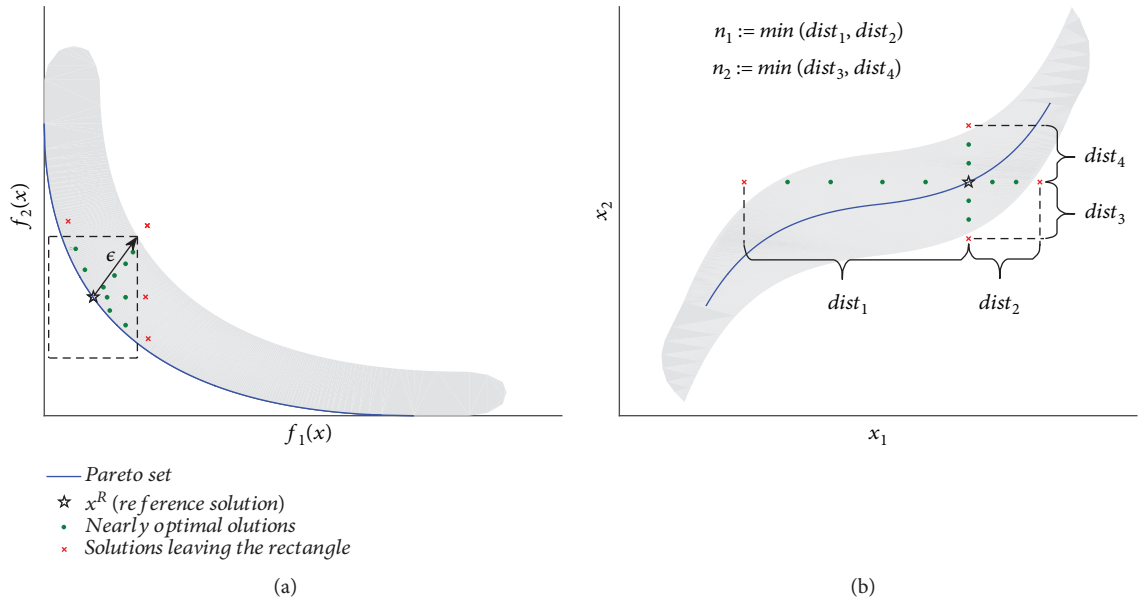


FIGURE 3: Determination of the vector neighborhood n when no other criterion is available. Each element of the vector n (n_1 and n_2) is set to the minimum distance between x^R and the first solutions (one for each direction) which leave the rectangle.

rectangle (n_1 and n_2 in Figure 3). Through this procedure, we manage to quantify what excursion is needed in the search space in order to get significant changes in the objective space and, in this way, we are able to define n when no other criterion is available.

3.2. Discretization of the New Set of Interest. Another issue is that $P_{Q,n}$ may contain infinite solutions, so it is necessary to discretize it and to obtain a finite set of solutions ($P^*_{Q,n}$) which are adequately distributed. In order to carry out this task, the concepts of box, n_box, and their associated dominances have to be defined beforehand.

Definition 9 (box [19]). Let $\delta_i = (f_i^{\max} - f_i^{\min})/n\_\text{box}_i$, with $n\_\text{box} = [n\_\text{box}_i, \ldots, n\_\text{box}_m]$ for all $i \in [1, \ldots, m]$. Given a decision vector x, $\text{box}(x)$ is defined as the vector $\text{box}(x) = [\text{box}_1(x), \ldots, \text{box}_m(x)]$, where

$$\text{box}_i = \quad \forall i \in [1, \ldots, m], \tag{6}$$

for $\delta_i > 0$, and where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of $f(x)$.

Definition 10 (box dominance [19]). Given two decision vectors x^1 and x^2 whose boxes are $\text{box}(x^1)$ and $\text{box}(x^2)$, respectively, x^1 is said to box dominate x^2, $\text{box}(x^1) \preceq \text{box}(x^2)$, if $\text{box}_i(x^1) \leq \text{box}_i(x^2)$ for all $i \in [1, \ldots, m]$ and $\text{box}_j(x^1) < \text{box}_j(x^2)$ for at least one j, $j \in [1, \ldots, m]$.

Definition 11 (n-box dominance). Given the neighboring decision vectors x^1 and x^2 whose boxes are $\text{box}(x^1)$ and $\text{box}(x^2)$, respectively, x^1 is said to n-box dominate x^2, box $(x^1) \preceq_n \text{box}(x^2)$, if $\text{box}(x^1) \preceq \text{box}(x^2)$.

The algorithm only keeps non-n-box-dominated solutions. A box cannot contain two neighboring solutions. When a neighboring solution is found in the same box, the nearest solution to the ideal corner (the lower left one) will be chosen (see Figure 4, d_1 and d_2) and the rest will be discarded. Figure 4 shows how the algorithm performs the discretization. x^1 is n-box dominated (i.e., box dominated by a neighboring solution) by x^2 and, therefore, it is discarded. x^3 is box dominated but not n-box dominated (since x^2 and x^3 are nonneighboring solutions), so it is an alternative, as it is a nearly optimal solution. x^4, x^5, and x^6 are in the same box. Two neighboring solutions cannot be in the same box, so either x^4 or x^5 has to be eliminated. In this case, x^4 is eliminated because it has a greater distance to the ideal corner than x^5 ($d_2 < d_1$). x^6 is a good alternative because it is a nearly optimal solution, it is not n-box dominated and it does not belong to either the neighborhood of x^4 or x^5.

The algorithm searches a discretization of the set of interest. This new discrete set is $P^*_{Q,n}$. The maximum number of solutions (in the worst case) of $P^*_{Q,n}$ is $|P^*_{Q,n}|$ (see 7). Similarly, P^*_Q and $P^*_{Q,\epsilon}$ are the discrete sets of P_Q and $P_{Q,\epsilon}$, respectively.

$$\left|P^*_{Q,n}\right| \leq \frac{\prod_{i=1}^{m}\left(\lceil \epsilon_i/\delta_i \rceil + n_{\text{box}_i}\right)}{\max_{i=1}^{m}\left(\lceil \epsilon_i/\delta_i \rceil + n_{\text{box}_i}\right)} \prod_{j=1}^{k} \frac{\bar{x}_j - \underline{x}_j}{n_i}. \tag{7}$$

3.3. Description of nevMOGA. The algorithm nevMOGA is based on the algorithm evMOGA [15]. The main difference between them is that nevMOGA computes an additional population, namely, the set of nearly optimal solutions

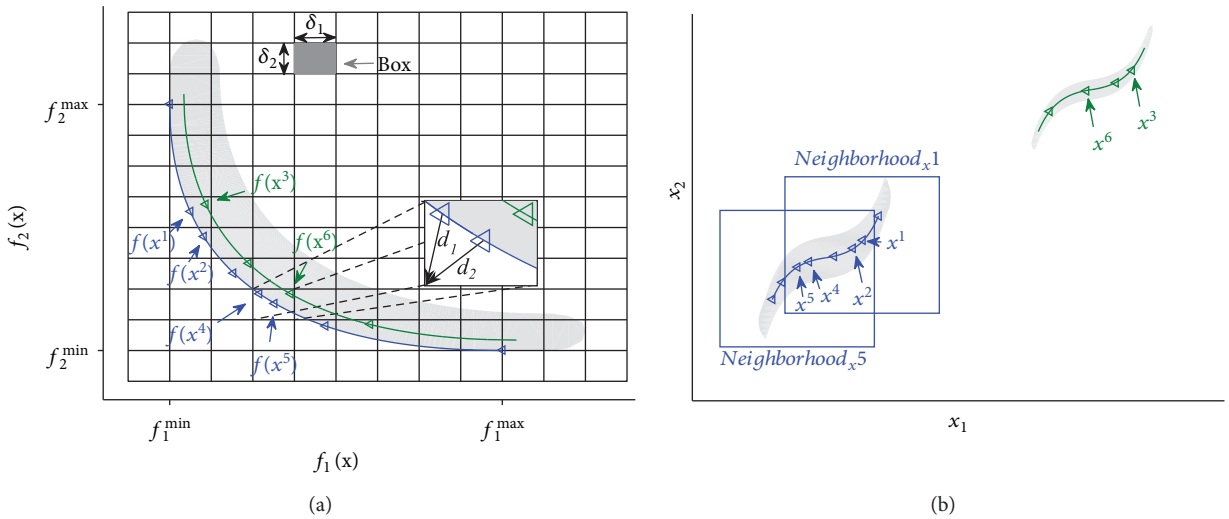


FIGURE 4: An example of discretization of a MOP with nevMOGA. x^1 is discarded because it is n-box dominated (i.e., box dominated by neighboring solutions) by x^2. x^4 is eliminated since there is a neighboring solution (neighborhood$_{x^5}$) in the same box (x^5) with less distance to the ideal corner ($d_2 < d_1$).

nondominated in their neighborhood (subfront(t)). Its main characteristics are the following:

(i) It is an elitist algorithm with two archives where the individuals of P_Q^* and $P_{Q,n}^*/P_Q^*$ are stored. It has also an additional, auxiliary population formed by the new individuals which are created during the process

(ii) It is a real-coded algorithm and uses crossover (extended intermediate crossover) and mutation (random Gaussian distribution) operators

The algorithm manages four populations (see Figure 5):

(1) $P(t)$: by means of the population $P(t)$, the search space is explored, with the aims of obtaining the solutions of $P_{Q,n}$ and having diversity in the set of solutions. The number of individuals in this population is constant and equal to Nind$_P$

(2) Front(t) is the archive where P_Q^* is stored, i.e., a discrete approximation of the Pareto front. The size of this population varies but is always less than or equal to a given maximum size which depends on the number of boxes previously defined by the user

(3) Subfront(t) is the archive where $P_{Q,n}^*/P_Q^*$ is stored, i.e., the nearly optimal solutions nondominated in their neighborhood. Its size is variable but bounded, depending on the number of boxes

(4) $G(t)$ is an auxiliary population where the new individuals generated by the algorithm in each iteration are stored. The number of individuals of this population is Nind$_G$, which must be multiple of 4

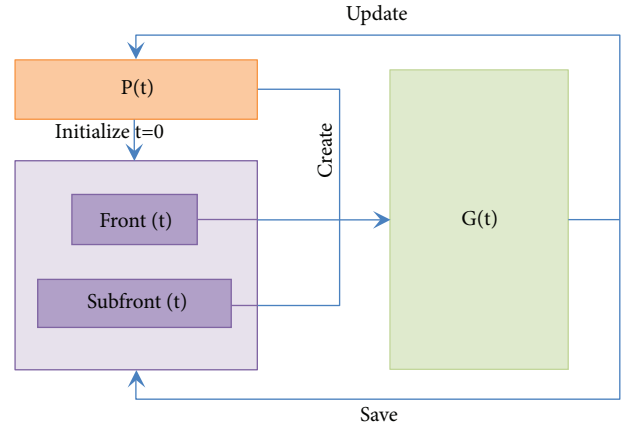


FIGURE 5: Structure of nevMOGA, formed by four populations.

Now that the archives front(t) and subfront(t) have been presented, it is possible, with the help of the prior definitions, to establish the conditions that a solution must fulfill in order to enter them. This is the aim of the following two additional definitions:

Definition 12 (inclusion of x in front(t)). Given a solution x and the archive front(t), x will be included in front(t) if and only if

$$\nexists x^* \in \text{front}(t)\ (\text{box}(x^*) \preceq \text{box}(x)) \vee (\text{box}(x^*) = \text{box}(x) \wedge \text{dist}(x^*) \leq \text{dist}(x)), \quad (8)$$

where dist(z) is the distance from z to the ideal corner (lower left) of the box (see Figure 3, d_1 and d_2). Additionally, if x is included in front(t), then all the solutions x^* that fulfill the following condition will be eliminated from front(t):

```
1:  t:=0;
2:  Front (t):= ∅;
3:  Sub-Front (t):= ∅;
4:  Create initial population P (t) at random
5:  Calculate f (x) ∀ x ∈ P (t)
6:  Order population P (t) according to the niche count                ▷ Definition 14
7:  Inclusion of the individuals of P (t) in Front (t)                 ▷ using Algorithm 2
8:  Inclusion of the individuals of P (t) ∉ Front (t) in Sub-Front (t) ▷ using Algorithm 3
9:  for t:= 1:Number of iterations
10:     Create population G (t)                                        ▷ using Algorithm 4
11:     Calculate f (x) ∀ x ∈ G (t)
12:     Inclusion of the individuals of G (t) in Front (t)             ▷ using Algorithm 2
13:     Inclusion of the individuals of G (t) ∉ Front (t) in Sub-Front (t) ▷ using Algorithm 3
14:     Update P (t) with the individuals of G (t)                     ▷ using Algorithm 5
15:     Order population P (t)
16: end for
```

Algorithm 1: Main pseudocode

$$(\mathrm{box}(x) \preceq \mathrm{box}(x^*)) \vee (\mathrm{box}(x) = \mathrm{box}(x^*) \wedge \mathrm{dist}(x) \leq \mathrm{dist}(x^*)). \tag{9}$$

Then, it will be determined whether any of these solutions x^* which have been eliminated from $\mathrm{front}(t)$ is included in $\mathrm{subfront}(t)$ or not, by applying Definition 13. Furthermore, all the solutions x' that fulfill the following condition will be eliminated from $\mathrm{subfront}(t)$:

$$x \preceq_{-\epsilon} x'. \tag{10}$$

Definition 13 (inclusion of x in $\mathrm{subfront}(t)$). Given a solution x such that $x \notin \mathrm{front}(t)$, x will be included in $\mathrm{subfront}(t)$ if and only if

$$\begin{aligned} &\nexists x^* \in \mathrm{front}(t) \ : x^* \preceq_{-\epsilon} x \vee \nexists x^* \in \mathrm{front}(t) \\ &\quad \bigcup \mathrm{subfront}(t) \ : (\mathrm{box}(x^*) \preceq_n \mathrm{box}(x)) \\ &\quad \vee (\mathrm{box}(x^*) =_n \mathrm{box}(x) \wedge \mathrm{dist}(x^*) \leq \mathrm{dist}(x)). \end{aligned} \tag{11}$$

Additionally, if x is included in $\mathrm{subfront}(t)$, then all the solutions x^* that fulfill the following condition will be eliminated from $\mathrm{subfront}(t)$:

$$(\mathrm{box}(x) \preceq_n \mathrm{box}(x^*)) \vee (\mathrm{box}(x) =_n \mathrm{box}(x^*) \wedge \mathrm{dist}(x) \leq \mathrm{dist}(x^*)). \tag{12}$$

In order to maintain the diversity of solutions in $P^*_{Q,n}$, it is necessary that the population $P(t)$ be diverse too. For this, $P(t)$ is permanently ordered by using the niche count, which is an indicator of how densely populated a solution is in $P(t)$ (Definition 14). In this way, during the processes of selection and substitution, the solutions in the less populated areas have a higher probability of being chosen to generate new individuals, whereas those in the more populated areas have a higher probability of being substituted.

Definition 14 (niche count [20]).

$$\mathrm{niche}_i = \sum_{i=1}^{\mathrm{Nind}_p} \mathrm{sh}(d_{ij}), \tag{13}$$

where $\mathrm{sh}(d_{ij})$ is the sharing function, which gives a measure of how similar two elements in a population are and is defined by

$$\mathrm{sh}(d_{ij}) = \begin{cases} \sum_{j=1}^{k} \left(1 - \dfrac{\left|x_j^1 - x_j^2\right|}{n_j}\right), & \text{if } x^1 =_n x^2, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The algorithm pseudocode is the following:

Lines 3, 6, 8, 10, 13, 14, and 15 of the previous pseudocode constitute the changes which have been added to evMOGA, the algorithm on which ours is based. Next, the most important steps in nevMOGA are explained:

(i) Lines 7 and 12 (Algorithm 2). Here, it is determined whether a new individual is included in $\mathrm{front}(t)$ or not. For this, the individual has to fulfill Definition 12. If the new individual is finally included in $\mathrm{front}(t)$, then it is analyzed whether its inclusion eliminates other individuals from $\mathrm{front}(t)$ (in which case it will be established whether they must be included in $\mathrm{subfront}(t)$) and from $\mathrm{subfront}(t)$

(ii) Lines 8 and 13 (Algorithm 3). Here, it is determined whether a new individual which was not included in $\mathrm{front}(t)$ has to be included in $\mathrm{subfront}(t)$ or not. For this, the individual has to fulfill Definition 14. If the new individual is included in $\mathrm{subfront}(t)$, then it is analyzed whether its inclusion eliminates other individuals from $\mathrm{subfront}(t)$

if $\exists \boldsymbol{x}^* \in Front(t) : (\mathbf{Box}(\boldsymbol{x}^*) \preceq \mathbf{Box}(\mathbf{x}))$ **then**
  return
else if $\exists \boldsymbol{x}^* \in Front(t) : (\mathbf{Box}(\boldsymbol{x}^*) = \mathbf{Box}(\boldsymbol{x}) \,\&\, dist(\boldsymbol{x}^*) \leq dist(\boldsymbol{x}))$ **then**
  return
end if
Include $\boldsymbol{x}$ in Front (t)
if $\exists \boldsymbol{x}^* \in Front(t) : (\mathbf{Box}(\boldsymbol{x}) \preceq \mathbf{Box}(\boldsymbol{x}^*))$ **then** ▷ Remove non-optimal solutions
  Remove $\mathbf{x}^*$ from Front (t)
else if $\exists \boldsymbol{x}^* \in Front(t) : (\mathbf{Box}(\boldsymbol{x}) = \mathbf{Box}(\boldsymbol{x}^*) \,\&\, dist(\boldsymbol{x}) \leq dist(\boldsymbol{x}^*))$ **then**
  Remove $\boldsymbol{x}^*$ from Front (t)
end if
if $\exists \boldsymbol{x}^* \in Sub - Front(t) : \boldsymbol{x} \preceq_{-\epsilon} \boldsymbol{x}^*$ **then** ▷ Remove non-nearly-optimal solutions
  Remove $\boldsymbol{x}^*$ from Sub-Front (t)
end if

Algorithm 2: Inclusion of x in front(t). ▷ Definition 12

if $\exists \boldsymbol{x}^* \in Front(t) : \boldsymbol{x}^* \preceq_{-\epsilon} \boldsymbol{x}$ **then**
  return
else if $\exists \boldsymbol{x}^* \in Front(t) \bigcup Sub - Front(t) : (\mathbf{Box}(\boldsymbol{x}^*) \preceq_n \mathbf{Box}(\boldsymbol{x}))$ **then**
  return
else if $\exists \boldsymbol{x}^* \in Front(t) : (\mathbf{Box}(\boldsymbol{x}^*) =_n \mathbf{Box}(\boldsymbol{x}) \,\&\, dist(\boldsymbol{x}^*) \leq dist(\boldsymbol{x}))$ **then**
  return
end if
Include $\boldsymbol{x}$ in Sub-Front (t)
if $\exists \boldsymbol{x}^* \in Sub - Front(t) : (\mathbf{Box}(\boldsymbol{x}) \preceq_n \mathbf{Box}(\boldsymbol{x}^*))$ **then** ▷ Remove non-nearly-optimal solutions
  Remove $\boldsymbol{x}^*$ from Sub-Front (t)
else if $\exists \boldsymbol{x}^* \in Sub - Front(t) : (\mathbf{Box}(\boldsymbol{x}) =_n \mathbf{Box}(\boldsymbol{x}^*) \,\&\, dist(\boldsymbol{x}) \leq dist(\boldsymbol{x}^*))$ **then**
  Remove $\boldsymbol{x}^*$ from Sub-Front (t) then
end if

Algorithm 3: Inclusion of x in subfront(t). ▷ Definition 13

Choose $\boldsymbol{x}^{\mathbf{F}} \in$ Front (t) ▷ Random selection
Choose $\boldsymbol{x}^{\mathbf{P1}} \in$ P (t) ▷ According to an exponential distribution ($\mu = Nind_p/10$)
u≔ random(1)
if u> P_c/m **then** ▷ P_c/m is the probability of crossover and mutation
  $\boldsymbol{x}^{\mathbf{G1}}$ and $\boldsymbol{x}^{\mathbf{G2}}$ are obtained by crossing over $\boldsymbol{x}^{\mathbf{P1}}$ and $\boldsymbol{x}^{\mathbf{F}}$
else
  $\boldsymbol{x}^{\mathbf{G1}}$ and $\boldsymbol{x}^{\mathbf{G2}}$ are obtained by mutating $\boldsymbol{x}^{\mathbf{P1}}$ and $\boldsymbol{x}^{\mathbf{F}}$
end if
Choose $\boldsymbol{x}^{\mathbf{SF}} \in$ Sub-Front (t) ▷ Random selection
Choose $\boldsymbol{x}^{\mathbf{P2}} \in$ P (t) ▷ According to an exponential distribution ($\mu = Nind_p/10$)
u≔ random(1)
if u>P_c/m then
  $\boldsymbol{x}^{\mathbf{G3}}$ and $\boldsymbol{x}^{\mathbf{G4}}$ are obtained by crossing over $\boldsymbol{x}^{\mathbf{P2}}$ and $\boldsymbol{x}^{\mathbf{SF}}$
else
  $\mathbf{x}^{\mathbf{G3}}$ and $\mathbf{x}^{\mathbf{G4}}$ are obtained by mutating $\mathbf{x}^{\mathbf{P2}}$ and $\mathbf{x}^{\mathbf{SF}}$
end if

Algorithm 4: Create population $G(t)$

(iii) Line 10 (Algorithm 4). The population $G(t)$ is created by using the following procedure:

(1) Two individuals are chosen at random, one from front(t) (x^F) (where any individual has the same

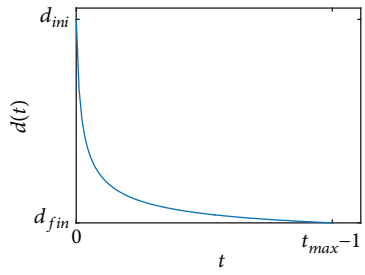


Figure 6: Graph of $d(t)$ from $t = 0$ to $t = t_{\max} - 1$. The parameters d_{ini} and d_{fin} are the initial and final values of $d(t)$.

probability of being chosen) and another from $P(t)$ (x^P) (where each one has a different probability of being chosen). In effect, the solutions of $P(t)$ are ordered according to their niche count (from least to greatest) and its individual is chosen according to an exponential distribution, which makes it more likely for a solution with a lower niche count to be chosen. This procedure favors the uniform distribution of the solutions

(2) A random number $u \in [0\ 1]$ is generated. If $u > P_c/m$ (probability of crossover/mutation), then a crossover will be performed (step 3); otherwise, a mutation will be performed (step 4)

(3) x^P and x^F are crossed by means of an extended intermediate crossover operator, which produces two new individuals x^{h1} and x^{h2}:

$$\begin{aligned} x_i^{h1} &= \alpha_i(t)x_i^P + (1 - \alpha_i(t))x_i^F, \\ x_i^{h2} &= (1 - \alpha_i(t))x_i^P + (\alpha_i(t))x_i^F. \end{aligned} \tag{15}$$

The parameter $\alpha_i(t)$ is a random value uniformly distributed which belongs to interval $[-d(t)\ 1 + d(t)]$, and $d(t)$ is a parameter which is adjusted by using an exponential decreasing function, as in *simulated annealing* [19]:

$$d(t) = \frac{d_{\text{ini}}}{\sqrt{1 + \left((d_{\text{ini}}/d_{\text{fin}})^2 - 1\right)(t/t_{\max-1})}}, \quad d_{\text{ini}} < d_{\text{fin}} \tag{16}$$

Figure 6 shows $d(t)$ from $t = 0$ to $t = t_{\max} - 1$, where $d(0) = d_{\text{ini}}$ and $d(t_{\max} - 1) = d_{\text{fin}}$.

(4) x^P and x^F are mutated by using a random mutation with Gaussian distribution:

$$\begin{aligned} x_i^{h1} &= x_i^P + N(0, \beta 1_i(t)), \\ x_i^{h2} &= x_i^F + N(0, \beta 2_i(t)), \end{aligned} \tag{17}$$

where the variances $\beta 1_i(t)$ and $\beta 2_i(t)$ are expressed as a percentage of $x_{i\ \max} - x_{i\ \min}$. These variances are calculated by a function which is similar to the one previously used for the parameter $d(t)$:

$$\beta(t) = \frac{\beta_{\text{ini}}}{\sqrt{1 + \left((\beta_{\text{ini}}/\beta_{\text{fin}})^2 - 1\right)(t/t_{\max-1})}} \tag{18}$$

(5) x^{h1} and x^{h2} are included in $G(t)$

These five steps are repeated again from step one but now with a solution from subfront(t) instead of one from front(t). The whole process is executed again and again until $G(t)$ is full. In order to set the parameters which appear in the equations of the steps 3 and 4, the default values suggested by [19] for the original algorithm (evMOGA) are taken.

(iv) Line 14 (Algorithm 5). In the updating of $P(t)$, it is determined whether a new individual from $G(t)$ (x^G) substitutes any other existing individual in $P(t)$ or not. The search for the individual to be substituted (x^P) starts from an individual chosen by an exponential distribution which is the inverse of the one used for the choice of x^P in the creation of $G(t)$ (line 10, Algorithm 4) and, therefore, now the more populated solutions will be more likely to be chosen. This is aimed at achieving a more uniform distribution of the solutions. During this updating, there may occur three distinct cases:

(1) The new individual x^G is ϵ dominated by some member of front(t). In this case, an individual x^P dominated by x^G will be searched to be substituted

(2) The new individual x^G is not ϵ dominated by any member of front(t), and there exists an individual x^P dominated by x^G which is ϵ dominated by some member of front(t). In this case, it will be substituted

(3) The new individual x^G is not ϵ dominated by any member of front(t), and there does not exist an individual x^P dominated by x^G which is not ϵ dominated by any member of front(t). In this case, the initial solution x^P is chosen. If x^P and x^G are neighboring solutions, a solution n dominated by x^G will be searched to be substituted. If they are not, a solution n dominated by x^P will be searched to be substituted. In this last case, if that solution is not found, x^G will be randomly substituted by a solution from the neighborhood of x^P

3.4. Performance Metrics. Next, we present one metric which will serve to assess the performance of the different algorithms to be compared. This metric measures both the convergence and the diversity between two sets (the outcome

Choose $\boldsymbol{x}^{\mathbf{P}_{\text{Initial}}} \in$ P (t) ▷ According to an exponential distribution ($\mu = Nind_p/10$)
if $\nexists \boldsymbol{x}' \in$ Front (t) : $\boldsymbol{x}' \preceq_{-\epsilon} \boldsymbol{x}^{\mathbf{G}}$ **then**
  if $\exists \boldsymbol{x}^* \in$ P (t): $\exists \boldsymbol{x}'' \in$ Front (t) : $\boldsymbol{x}'' \preceq_{-\epsilon} \boldsymbol{x}^* \,\&\, x^G \preceq x^*$ **then** ▷The search for $\boldsymbol{x}^*$ starts from $\mathbf{x}^{\mathbf{P}_{\text{Initial}}}$
    $\boldsymbol{x}^* \coloneqq \boldsymbol{x}^{\mathbf{G}}$
    return
  else
    $X_N \coloneqq \forall \boldsymbol{x}^* \in P(t)$: $\mathbf{x}^{\mathbf{P}_{\text{Initial}}} =_n \boldsymbol{x}^*$
    if $\boldsymbol{x}^{\mathbf{G}} =_n \boldsymbol{x}^{\mathbf{P}_{\text{Initial}}}$ **then**
      if $\exists \boldsymbol{x}^* \in X_N : \boldsymbol{x}^{\mathbf{G}} \preceq \boldsymbol{x}^*$ **then** ▷ The search for $\boldsymbol{x}^*$ starts from $\boldsymbol{x}^{\mathbf{P}_{\text{Initial}}}$
        $\boldsymbol{x}^* \coloneqq \boldsymbol{x}^{\mathbf{G}}$
        return
      end if
    else
      if $\exists \boldsymbol{x}_1^* \,\&\, \mathbf{x}_2^* \in X_N : \boldsymbol{x}_1^* \preceq \boldsymbol{x}_2^*$ **then**
        $\boldsymbol{x}_2^* \coloneqq \boldsymbol{x}^{\mathbf{G}}$
        return
      end if
    end if
  end if
  $\boldsymbol{x}^{\mathbf{P}_{\text{Random}}}$ =Random(X_N) ▷ Random selection of an individual from X_N
  $\boldsymbol{x}^{\mathbf{P}_{\text{Random}}} \coloneqq \boldsymbol{x}^{\mathbf{G}}$
else
  if $\exists \boldsymbol{x}^* \in$ P (t): $\boldsymbol{x}^{\mathbf{G}} \preceq \boldsymbol{x}^*$ **then** ▷ The search for $\boldsymbol{x}^*$ starts from $\boldsymbol{x}^{\mathbf{P}_{\text{Initial}}}$
    $\boldsymbol{x}^* \coloneqq \boldsymbol{x}^{\mathbf{G}}$
  end if
end if

Algorithm 5: Update of $P(t)$ with the individuals of $G(t)$

set and the target set). The outcome set of each algorithm is the set of solutions that it returns, whereas the target set results from discretizing the search space with a fine grain. We use this metric to measure the performance of the algorithms in the objective space (convergence toward the front) and in the parameter space (diversity of solutions).

3.4.1. Averaged Distance Hausdorff. A single indicator is used to measure the convergence and diversity between two sets (averaged distance Hausdorff Δ_p, [21]). The sets Y and X have n_y and n_x solutions, respectively. The Δ_p is calculated by

$$\Delta_p(X, Y) \coloneqq \max\left(\left(\frac{1}{n_x}\sum_{i=1}^{n_x} \operatorname{dist}(x_i, Y)^p\right)^{1/p}, \left(\frac{1}{n_y}\sum_{i=1}^{n_y} \operatorname{dist}(X, y_i)^p\right)^{1/p}\right), \tag{19}$$

where

$$\operatorname{dist}(u, A) \coloneqq \inf_{v \in A} \|u - v\|. \tag{20}$$

This indicator is the average of the Hausdorff distance, and we use it here in order to measure the performance between an outcome set H and a target set $P^*_{Q,n}$. This performance is measured both in the objective space and in the parameter space. In this way, it is possible to measure convergence toward the front and diversity of solutions. We use $p = 2$. It is desirable that this indicator has the smallest possible value. Ideally, when $H = P^*_{Q,n}$, then $\Delta_p = 0$.

3.5. Benchmarks. Our algorithm nevMOGA has been tested on two distinct benchmarks. In this section, we describe them.

3.5.1. Benchmark 1. The first benchmark considered in this work corresponds to an academic example [18] and is stated as follows:

$$\begin{aligned}\min_x \quad & f(x) = [f_1(x) f_2(x)], \quad f_1(x) \\ & = (x_1 - t_1(c + 2a) + a)^2 + (x_2 - t_2 b)^2 + \delta_t, f_2(x) \\ & = (x_1 - t_1(c + 2a) - a)^2 + (x_2 - t_2 b)^2 + \delta_t,\end{aligned} \tag{21}$$

where

$$\begin{aligned} t_1 &= \operatorname{sgn}(x_1) \min\left(\left\lceil \frac{||x_1|x_1| - a - c/2}{2a + c} \right\rceil, 1\right), \\ t_2 &= \operatorname{sgn}(x_2) \min\left(\left\lceil \frac{|x_2| - b/2}{b} \right\rceil, 1\right), \\ \delta_t &= \begin{cases} 0, & \text{for } t_1 = 0,\ t_2 = 0, \\ 0.1, & \text{else}, \end{cases} \end{aligned} \tag{22}$$

subject to

$$\begin{aligned} \underline{x} &= [-8\ -8], \\ \bar{x} &= [8\ 8]. \end{aligned} \tag{23}$$

Setting $a = 0.5$, $b = 5$, and $c = 5$, this MOP has one global Pareto set:

$$P_{0,0} = [-0.5,0.5]x\{0\} = P_Q, \tag{24}$$

and eight local Pareto sets:

$$\begin{aligned} P_{-1,-1} &= [-6.5, -5.5]x\{-5\}, \\ P_{0,-1} &= [-0.5,0.5]x\{-5\}, \\ P_{1,-1} &= [5.5,6.5]x\{-5\}, \\ P_{-1,0} &= [-6.5, -5.5]x\{0\}, \\ P_{1,0} &= [5.5,6.5]x\{0\}, \\ P_{-1,1} &= [-6.5, -5.5]x\{5\}, \\ P_{0,1} &= [-0.5,0.5]x\{5\}, \\ P_{1,1} &= [5.5,6.5]x\{5\}. \end{aligned} \tag{25}$$

3.5.2. Benchmark 2. The second benchmark is an adaptation of the modified Rastrigin [22]. This benchmark has been modified in order to turn it into a MOP with one optimal set and nearly optimal solutions lying in different neighborhoods. This is its formulation:

$$\begin{aligned} \min_x \quad & f(x) = [f_1(x)\, f_2(x)], \quad f_1(x) \\ & = -\left(\sum_{i=1}^{2} 10 - 9\cos\left(2\pi \cdot k_i \cdot x_i\right)\right) \\ & \quad \cdot \left(1 - \sqrt{(x_1 - 0.65)^2 + (x_2 - 0.5)^2}\right), f_2(x) \\ & = x_1 + x_2 - 1.4, \end{aligned} \tag{26}$$

where $k_1 = 2$ and $k_2 = 3$ and subject to

$$\begin{aligned} \underline{x} &= [0, 0], \\ \bar{x} &= [2, 2]. \end{aligned} \tag{27}$$

4. Results and Discussion

In this section, we present the results of nevMOGA on the two benchmarks introduced and on a control design problem. These results will be analyzed on the basis of the metric which was introduced in Section 3.4. In addition, we compare nevMOGA with two other algorithms, namely, random and exhaustive searches. In order to make this comparison possible, all three algorithms evaluate the objective functions the same number of times. Although there do exist algorithms which search the set of nearly optimal solutions ([17, 18, 23]), none of them has the same set of interest as nevMOGA ($P^*_{Q,n}$). For this reason, nevMOGA cannot *directly* be compared with any of them. In order to obtain statistical results, each algorithm is tested 50 times on each benchmark. For the exhaustive search case, the evaluated points are uniformly distributed over the search space, forming a hypergrid. In order for it to obtain different results, the hypergrid is slightly lifted each time.

4.1. Benchmark 1. In this benchmark, since the global Pareto set and the local Pareto sets are known (Section 3.5.1), we can set the parameters ϵ and n more easily. Here, we set $\epsilon = [0.15\, 0.15]$ so that the eight sets described in (25) are regarded as nearly optimal solutions. We get n by applying the procedure described in Section 3.1 with a reference solution $x^R = [0\ 0]$ and the parameter ϵ previously set, resulting in $n = [0.13\, 0.38]$. Once these parameters have been determined, the set to be found (target set $P^*_{Q,n}$, see Figure 7) is

$$\begin{aligned} P_{Q,n} \coloneqq P_{0,0} \bigcup P_{-1,-1} \bigcup P_{0,-1} \bigcup P_{1,-1} \bigcup P_{-1,0} \\ \bigcup P_{1,0} \bigcup P_{-1,1} \bigcup P_{0,1} \bigcup P_{1,1}. \end{aligned} \tag{28}$$

Figure 8 and Table 3 show the statistical results of the three algorithms (each one is tested 50 times on this benchmark) that have been compared (nevMOGA, random search, and exhaustive search) using the metric Δ_p. Each algorithm evaluates the objective function 5000 times in each test run. The outcome sets of the three algorithms are compared with the target set $P^*_{Q,n}$. In Figure 8(a), there are the results against $\Delta_p(f(H), f(P^*_{Q,n}))$, whereas in Figure 8(b) against $\Delta_p(H, P^*_{Q,n})$, using boxplots and density estimation. The values yielded by nevMOGA for the metric in the objective space and in the parameter space are good (near 0) and better than those provided by random and exhaustive searches. For the random search, the approximation quality to the front varies enormously. For the exhaustive search, the results display less variability, although it performs worse than nevMOGA in both spaces. Moreover, a multiple comparison test ([24, 25]) leads to the conclusion that the results of the three algorithms are significantly different with respect to the metric in both spaces. For these reasons, it can be concluded that nevMOGA significantly outperforms random and exhaustive searches on benchmark 1.

In order to graphically compare the three algorithms, a representative solution (from the 50 test runs) of each one has been plotted in Figure 9. This representative solution has been chosen as the one whose values of $\Delta_p(H, P^*_{Q,n})$ and $\Delta_p(f(H), f(P^*_{Q,n}))$ are closest to the average values of that metric in both spaces. The outcome set obtained by nevMOGA has 70 solutions, whereas those obtained by random and exhaustive searches have 78 and 48 solutions, respectively. The great number of solutions found by nevMOGA in addition to the results shown in Figure 9 proofs that nevMOGA is capable of characterizing the target set $P^*_{Q,n}$ better than its competitors (random and exhaustive searches) on benchmark 1. This result, plus the fact that nevMOGA achieves lower values

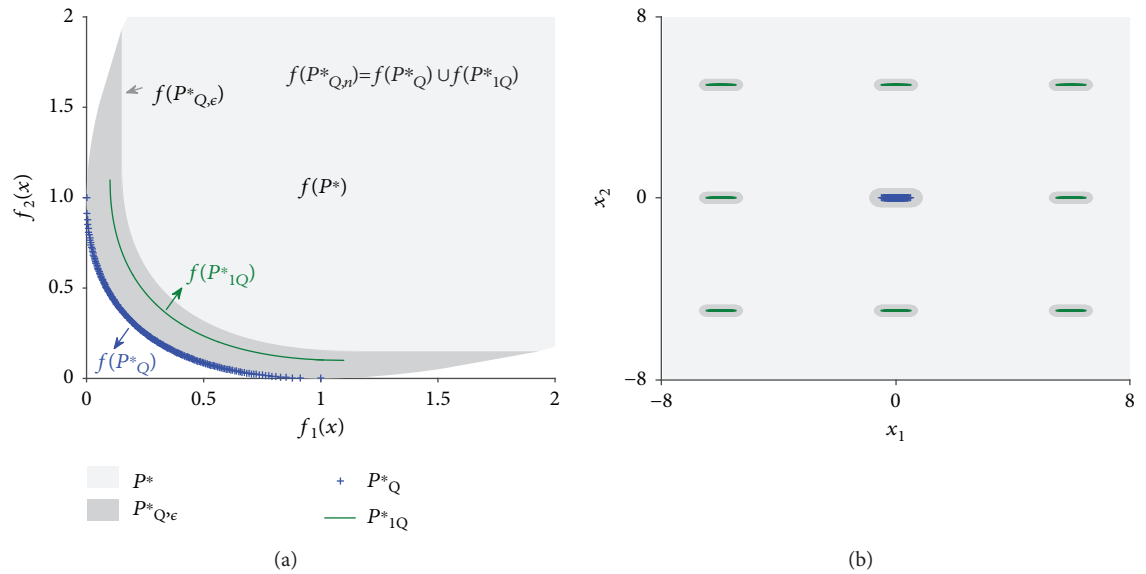


FIGURE 7: Target set $P^*_{Q,n}$, formed by $P^*_Q \cup P^*_{1Q}$. It consists of nine subsets, one of them contains the optimal solutions, and the rest the nearly optimal ones.

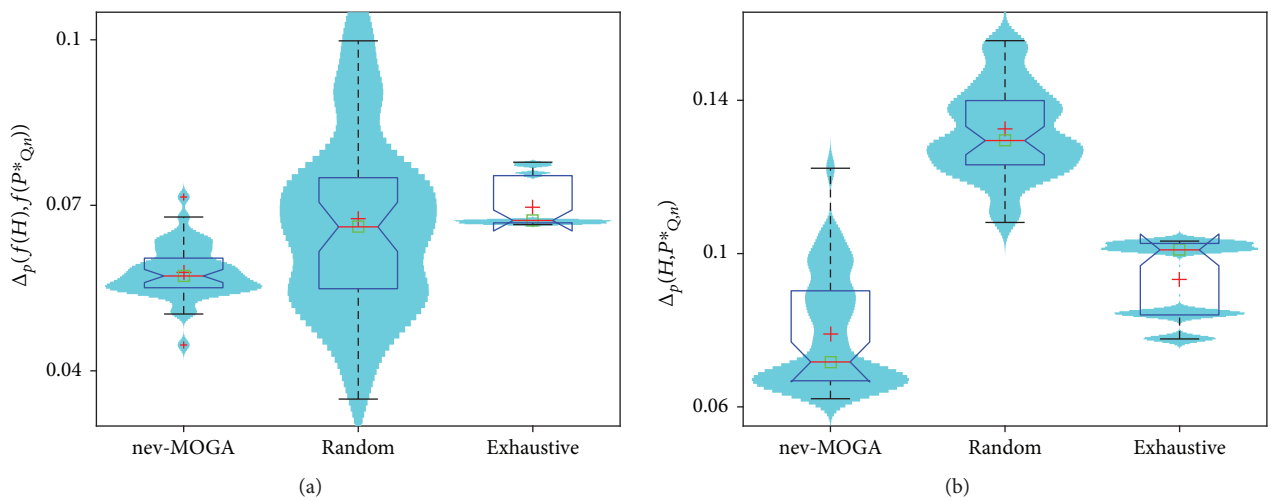
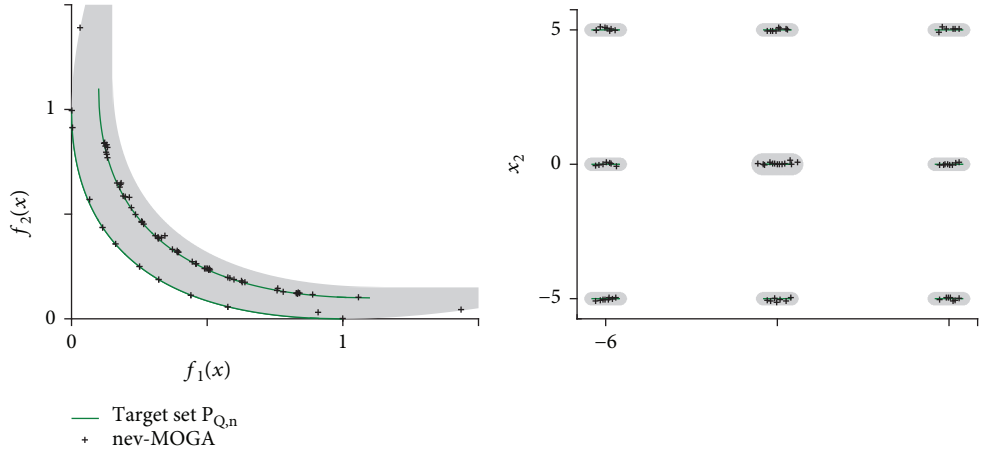


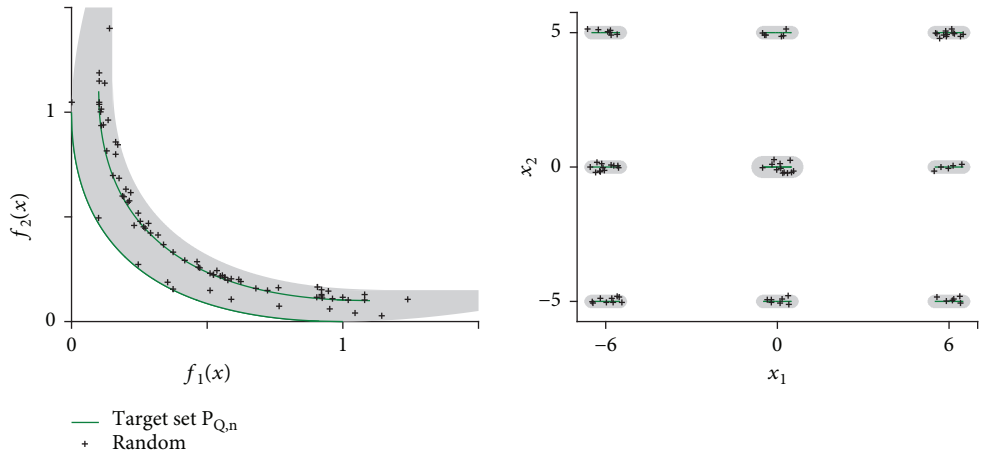
FIGURE 8: Boxplots for benchmark 1, comparing nevMOGA, random search, and exhaustive search. (a) $\Delta_p(f(H), f(P^*_{Q,n}))$ measures the approximation of the outcome set toward the Pareto front. (b) $\Delta_p(H, P^*_{Q,n})$ measures the approximation of the outcome set toward the Pareto set. The most significant values are shown in Table 3.

TABLE 3: Numerical values of the statistical results presented in Figure 8.

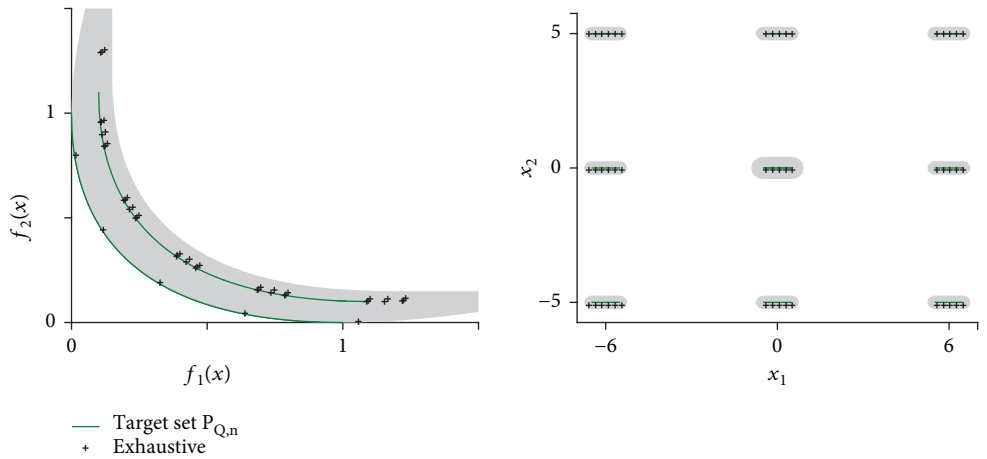
	$\Delta_p(f(H), f(P^*_{Q,n}))$			$\Delta_p(H, P^*_{Q,n})$		
	nevMOGA	Random	Exhaustive	nevMOGA	Random	Exhaustive
Maximum	0.0715	0.121	0.0778	0.122	0.169	0.103
Third quartile	0.0604	0.0750	0.0754	0.0903	0.140	0.103
Mean	0.0578	0.0676	0.0697	0.0790	0.133	0.0933
Median	0.0572	0.0661	0.0673	0.0717	0.130	0.101
First quartile	0.0551	0.0549	0.0668	0.0668	0.123	0.0840
Minimum	0.0447	0.0349	0.0665	0.0621	0.108	0.0777



(a) nevMOGA outcome set compared with the target set $P^*_{Q,n}$



(b) Random search outcome set compared with the target set $P^*_{Q,n}$



(c) Exhaustive search outcome set compared with the target set $P^*_{Q,n}$

FIGURE 9: A representative solution of each one of the three algorithms is compared with the target set $P^*_{Q,n}$ in the parameter space (right) and in the objective space (left). This representative solution has been chosen as the one whose values of Δ_p are closest to the average values of that metric in the objective space and in the parameter space.

of the metric Δ_p, leads us to conclude that nevMOGA accomplishes a better general performance on this benchmark.

This benchmark has also been solved by [18], where the set $P_{Q,\epsilon}$ is characterized. The algorithm tested in that work is called $P_{Q,\epsilon}$-MOEA. A comparison of the results of nevMOGA with those of $P_{Q,}$-MOEA shows that nevMOGA obtains a significantly smaller number of solutions (approximately four times smaller) without missing any

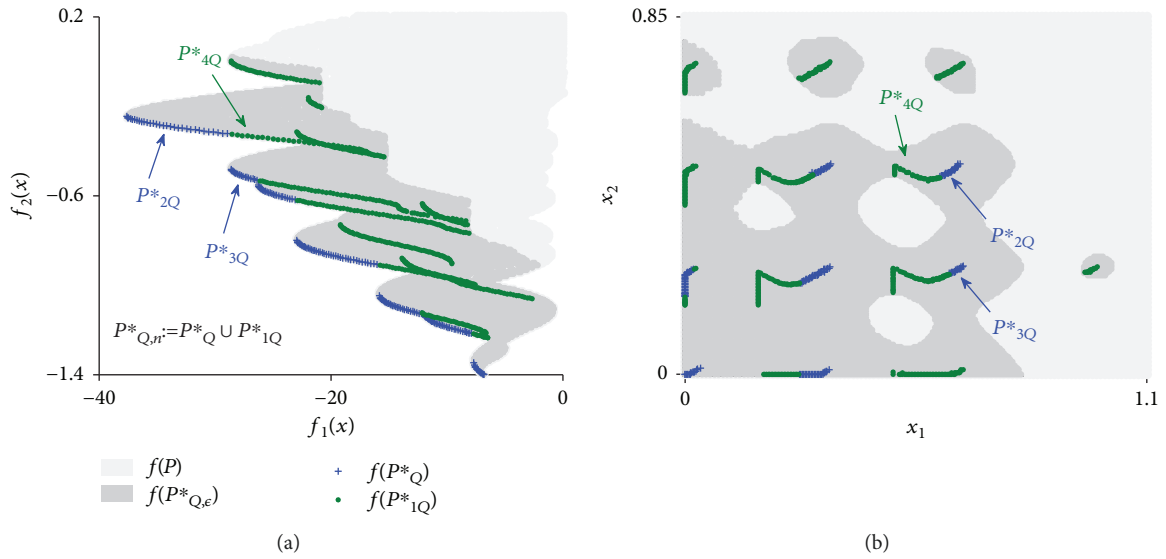


FIGURE 10: The target set $P^*_{Q,n}$ is formed by $P^*_Q \cup P^*_{1Q}$. The set P^*_{4Q} is dominated by the set P^*_{3Q}; however, they lie in distinct neighborhoods and, for this reason, both sets belong to the target set.

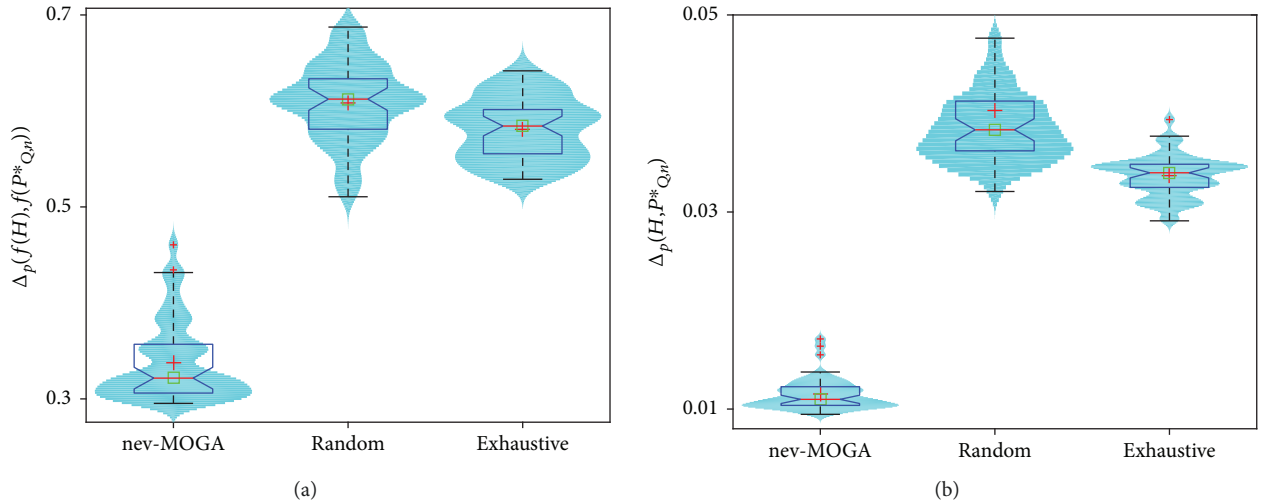


FIGURE 11: Boxplots for benchmark 2, comparing nevMOGA, random search, and exhaustive search. (a) $\Delta_p(f(H), f(P^*_{Q,n}))$ measures the approximation of the outcome set toward the Pareto front. (b) $\Delta_p(H, P^*_{Q,n})$ measures the approximation of the outcome set toward the Pareto set. The most significant values are shown in Table 4.

neighborhood containing nearly optimal solutions. This was our goal, namely, finding only those solutions that provide the most relevant information, and this with two aims: (1) not to slow down the algorithm and (2) to simplify the decision-making stage.

4.2. Benchmark 2. The statement of this second optimization problem was presented in Section 3.5.2. First, and with the aim of setting the two parameters of nevMOGA (ϵ and n), a tentative search for solutions is carried out within the defined range. This leads us to choose ϵ. Once $\epsilon = [7.7\ 0.3]$ has been set, we take a reference solution $x^R = [0.66\ 0.5]$ and follow the procedure detailed in Section 3.1 to obtain n. This results in $n = [0.15\ 0.15]$. The selection of ϵ and n leads to a particular target set $P^*_{Q,n}$, which is shown in Figure 10. As can be seen in the figure, P^*_{4Q} is dominated by the set P^*_{3Q} but they are in distinct neighborhoods. The set P^*_{4Q} does not have any solution dominated in its neighborhood and, therefore, it is part of the target set $P^*_{Q,n}$.

Figure 11 is similar to Figure 8 but for benchmark 2. Each algorithm is run 50 times and can evaluate the objective function 5000 times in each test run. Again, nevMOGA achieves better results than random and exhaustive searches with regard to the metric Δ_p, since it reaches values which are closer to zero. As we did in benchmark 1, we have also performed a multiple comparison test on these results. This comparison shows that the results obtained by the three search strategies are significantly different in both spaces. For these reasons, it can be concluded that nevMOGA

Table 4: Numerical values of the statistical results presented in Figure 11.

	$\Delta_p(f(H), f(P^*_{Q,n}))$			$\Delta_p(H, P^*_{Q,n})$		
	nevMOGA	Random	Exhaustive	nevMOGA	Random	Exhaustive
Maximum	0.460	0.687	0.641	0.0322	0.0697	0.0394
Third quartile	0.357	0.633	0.601	0.012	0.0412	0.0348
Mean	0.338	0.609	0.584	0.0119	0.0403	0.0337
Median	0.322	0.612	0.584	0.0110	0.0383	0.0340
First quartile	0.306	0.581	0.555	0.0104	0.0362	0.0325
Minimum	0.295	0.511	0.529	0.00946	0.0321	0.0291

significantly outperforms random and exhaustive searches on benchmark 2.

Figure 12 shows a representative solution (chosen in the same way as we did in benchmark 1) of each algorithm (nevMOGA, random, and exhaustive searches). A glance at the figure is sufficient to show that nevMOGA accomplishes a better approximation to the target set $P^*_{Q,n}$. Moreover, the outcome set obtained by nevMOGA has 73 solutions, whereas those obtained by random and exhaustive searches have 62 and 48 solutions, respectively. These two last facts (a better approximation to the target set $P^*_{Q,n}$ and a greater number of solutions) confirm the conclusion drawn before, when the results were statistically examined, namely, that nevMOGA outperforms both random and exhaustive searches.

4.3. Multiobjective Control Design Problem. In this section, we present the application of nevMOGA to a real design problem in the field of control engineering. The problem consists of optimizing the performance of two PI controllers, which operate a multivariable plant with two inputs and two outputs.

The model of the plant is [26]

$$Y(s) = \begin{pmatrix} y_1(s) \\ y_2(s) \end{pmatrix} = G(s) \begin{pmatrix} u_1(s) \\ u_2(s) \end{pmatrix}, \quad G(s) = \begin{pmatrix} \dfrac{5e^{-40}}{100\,\mathrm{s}+1} & \dfrac{1e^{-4}}{10\,\mathrm{s}+1} \\ \dfrac{-5e^{-40}}{10\,\mathrm{s}+1} & \dfrac{5e^{-40}}{100\,\mathrm{s}+1} \end{pmatrix}, \tag{29}$$

where y_1 and y_2 are the outputs of the plant, and u_1 and u_2 are the inputs (control actions). The time constants are all in seconds.

The control structure (also taken from [26]) is defined as a multiloop PI control which uses an off-diagonal pairing scheme, i.e., output y_1 is controlled by u_2 and y_2 by u_1. That is to say,

$$\begin{aligned} u_2(s) &= Kc_1 \left(e_1(s) + \left(\frac{1}{Ti_1} \right) \left(\frac{1}{s} \right) e_1(s) \right), \\ u_1(s) &= Kc_2 \left(e_2(s) + \left(\frac{1}{Ti_2} \right) \left(\frac{1}{s} \right) e_2(s) \right), \end{aligned} \tag{30}$$

where Kc_1 and Kc_2 are the proportional gains, Ti_1 and Ti_2 are the integral time constants, and $e_1 = r_1 - y_1$ and $e_2 = r_2 - y_2$ are the errors, where r_1 and r_2 are the setpoints for y_1 and y_2, respectively (see Figure 13).

The optimization problem is stated as follows:

$$\min_{\mathbf{x}} \quad f(x) = [f_1(x) f_2(x)], \tag{31}$$

where

$$\begin{aligned} f_1 &= \int_0^{t_f} e_1^2 + e_2^2 \Big|_{r_1=1, r_2=0} dt + \int_0^{t_f} e_1^2 + e_2^2 \Big|_{r_1=0, r_2=1} dt, \\ f_2 &= \int_0^{t_f} u_1^2 + u_2^2 \Big|_{r_1=1, r_2=0} dt + \int_0^{t_f} u_1^2 + u_2^2 \Big|_{r_1=0, r_2=1} dt, \end{aligned} \tag{32}$$

subject to

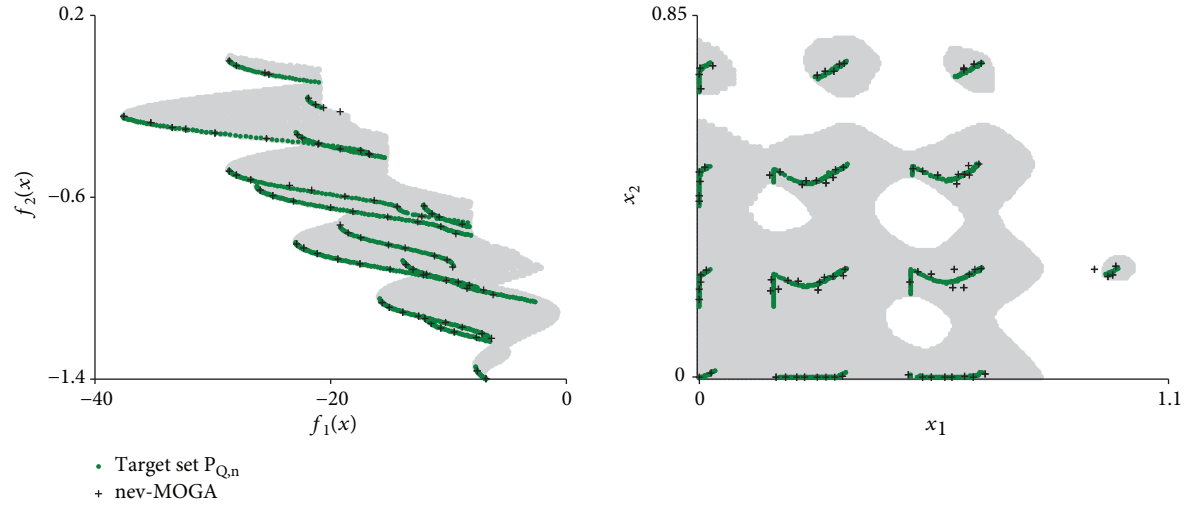
$$\begin{aligned} &\underline{x} \le x \le \bar{x}, \\ &f_1(x) < 300, \\ &f_2(x) < 200, \text{ stable in closed loop}, \end{aligned} \tag{33}$$

and where

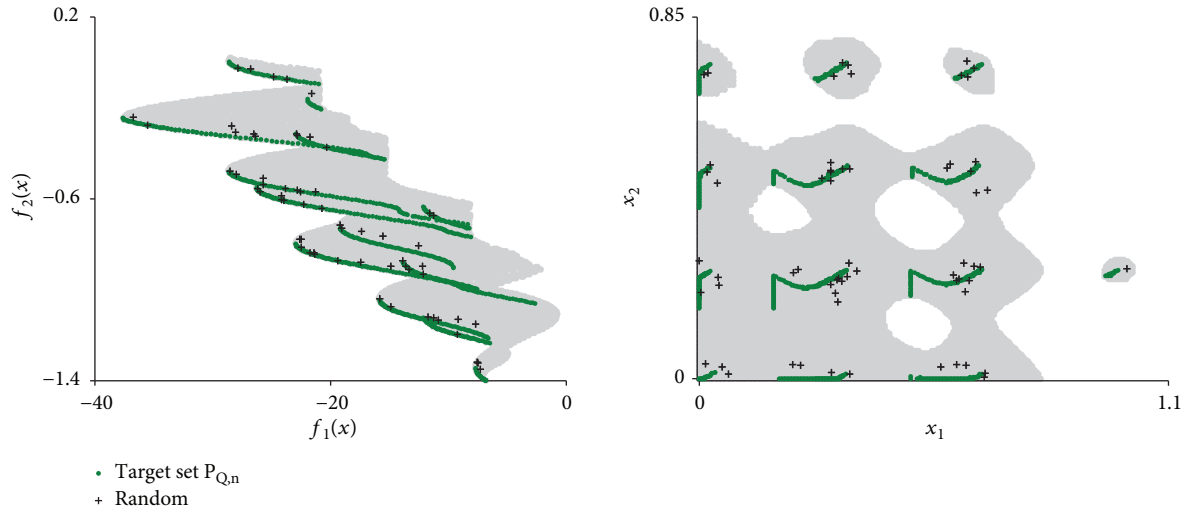
$$\begin{aligned} x &= [Kc_1, Ti_1, Kc_2, Ti_2] \underline{x} = [-1, 1, 0.1, 1], \\ \bar{x} &= [-0.1, 200, 10, 1000], \\ t_f &= 1000 \text{ seconds}. \end{aligned} \tag{34}$$

We launch two different simulations, sequentially. In the first one, a unitary step is applied to r_1, whereas r_2 is kept to zero. In the second one, the unitary step is now applied to r_2 and r_1 is kept to zero. In each simulation, the metric ISE (Integral Squared Error) is calculated for both outputs y_1 and y_2. Our first objective f_1 is then defined as the sum of the values of the ISE metrics of each simulation. So, the objective f_1 measures the setpoint tracking performance of the control system. The second objective f_2 is calculated in a similar manner but using the control action signals instead of the errors. Therefore, f_2 measures the control effort. Note that each objective (f_1 and f_2) can be seen as a sum of 4 subobjectives (see 32).

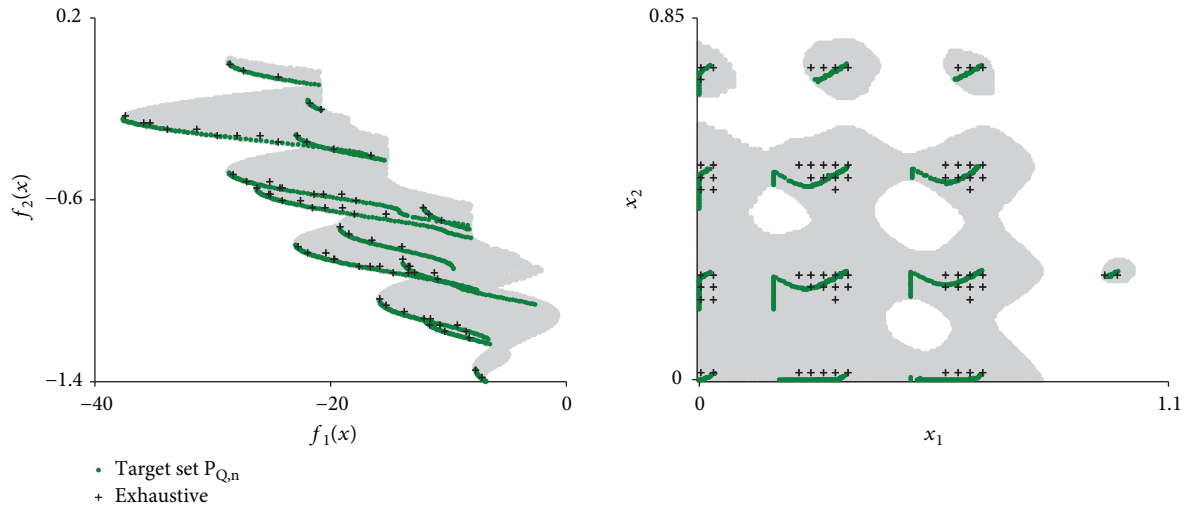
Once the optimization problem has been defined, the two parameters of nevMOGA (ϵ and n) have to be set. It is important to choose these parameters correctly, especially n. First of all, we have to decide what performance loss (with respect to



(a) nevMOGA outcome set compared with the target set $P^*_{Q,n}$



(b) Random search outcome set compared with the target set $P^*_{Q,n}$



(c) Exhaustive search outcome set compared with the target set $P^*_{Q,n}$

FIGURE 12: A representative solution of each one of the three algorithms is compared with the target set $P^*_{Q,n}$ in the parameter space (right) and in the objective space (left). This representative solution has been chosen as the one whose values of Δ_p are closest to the average values of that metric in the objective space and in the parameter space.

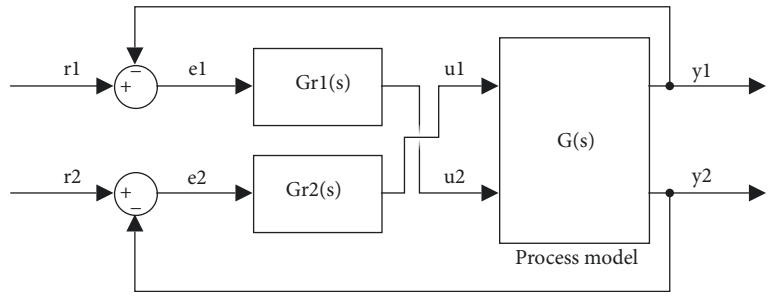
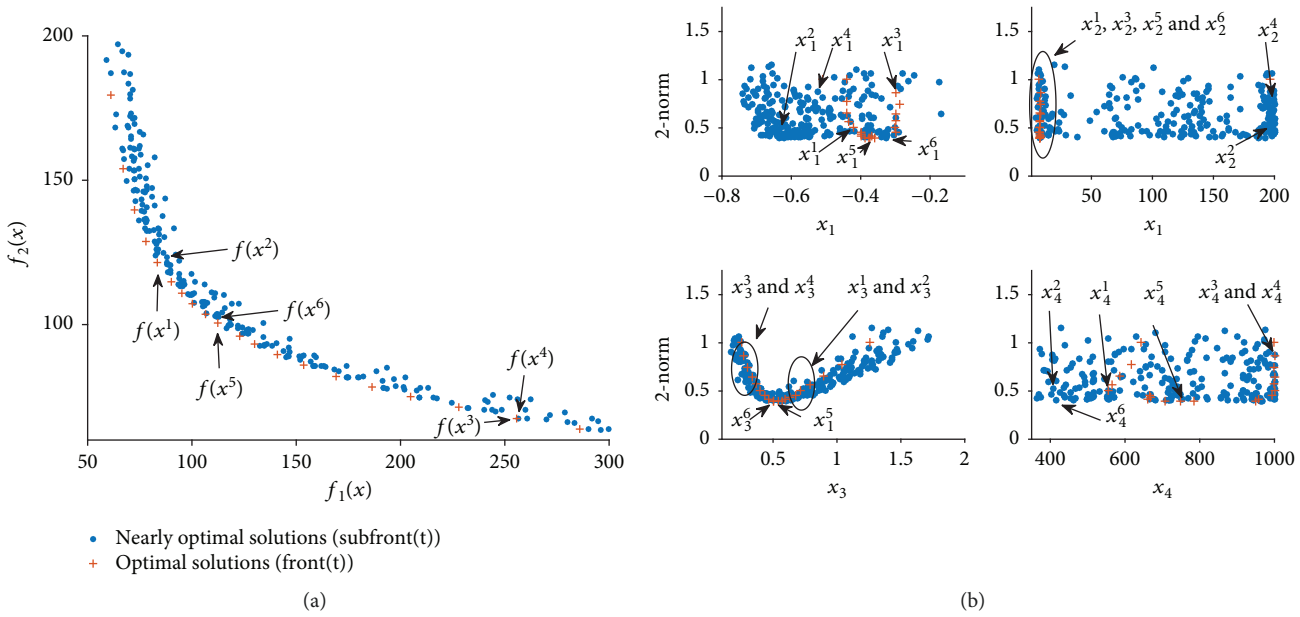


Figure 13: The control structure: a multiloop PI control which uses an off-diagonal pairing scheme.



(a) (b)

Figure 14: Set $P^*_{Q,n}$ found by nevMOGA. The optimal solutions (set P_Q) are shown in red and the rest (nearly optimal solutions nondominated in their neighborhood) in blue. x^1, x^3, and x^5 belong to P_Q (Pareto optimal solutions), whereas x^2, x^4, and x^6 are nearly optimal solutions.

the Pareto set) we are willing to accept (i.e., the parameter ϵ), in other words, what we are still going to regard as a nearly optimal solution. Choosing a high value would lead to a great number of possible solutions, whereas a low value would provide very few. In this application example, although the design objectives do have a physical sense, it is not self-evident how to choose ϵ. In order to solve this difficulty, we have proceeded in the following way. First, we have taken the values of f_1 and f_2 corresponding to the control designed in [26] as a reference solution. Then, we have pinpointed that solution on the Pareto front (which we have taken from [27]). Finally, on the basis of that information, we have decided to choose $\epsilon = [10\ 5]$. Note that there is an unavoidable subjective element concerning the way in which ϵ is chosen. Now, we have to assign a value to n, which is the most critical parameter of the algorithm. On the one hand, choosing an excessively low value would make most solutions be nonneighboring solutions, which in turn would cause the algorithm to tend to obtain the whole set $P^*_{Q,\epsilon}$. On the other hand, choosing an excessively high value would result in many solutions being neighboring solutions, which would cause the algorithm to tend to obtain only the set P^*_Q. This is why it is crucial that n be chosen carefully, supporting the choice with good reasons. Next, we describe how we have set n. First, we choose a reference controller $x = [-0.36, 7.86, 0.64, 628.4]$ (taken from [26]). Second, starting from the solution corresponding to this reference controller, the value of each of its parameters is increased and decreased independently, until reaching significant changes in the system response (see Figure 3). In this way, we finally get $n = [0.25\ 50\ 0.1\ 200]$; these are the values from which the system response is significantly different from the one provided by the reference controller.

As mentioned before, the objective functions include several terms which measure the performance and the control effort of each control loop (as if there were 4 sub-objectives added for each objective). For this reason, there may be solutions with the same objective value which are, however, completely different. This way of defining the objective functions is useful when the number of control loops is high, since it reduces the number of objectives and simplifies the decision-making stage. Yet, this approach has a shortcoming, namely, many interesting solutions may be

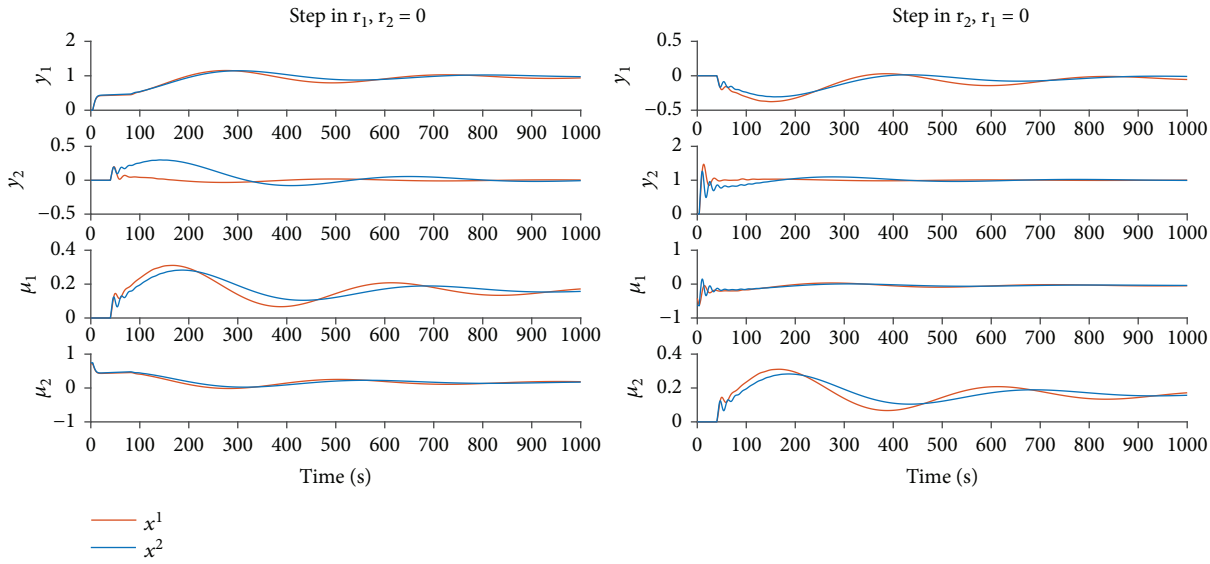


FIGURE 15: System responses for x^1 (optimal solution) and x^2 (nearly optimal solution). Although $f(x^2) \simeq f(x^1)$, both transient responses are different, since x^1 and x^2 are nonneighboring solutions.

neglected, because they may become nearly optimal solutions or multimodal solutions. Figure 14 shows the set of solutions $P^*_{Q,n}$ found by nevMOGA (for a given discretization of the objective space), where x^1, x^3, and x^5 are the optimal solutions, and x^2, x^4, and x^6 are the nearly optimal solutions. In order to show the decision variables, we employ a visualization tool called level diagrams ([28, 29]). For the ordinate axis, we have used the 2-norm. For the graphical representation of the objectives, we have not used this tool though, since there are only two and, in this case, plotting them on a single graph, where they can directly be compared, is preferable. Let us draw attention to the decision variables x^1 and x^2 (Figures 14(b)). Note how nevMOGA has been able to identify new areas in the search space containing solutions which may be interesting for the decision-maker and that would have been neglected in a traditional optimization approach, where only the set of Pareto optimal solutions P_Q is characterized.

Figure 15 shows the system responses for the solutions x^1 (optimal solution) and x^2 (nearly optimal solution), which have very similar values of f_1 and f_2 (see Figure 14(a)). These two solutions lie in distinct neighborhoods, and their corresponding responses are different too. This should not cause surprise since their corresponding controllers are significantly dissimilar from each other.

A more careful comparison of these two transient responses shows that x^1 presents a much better response than x^2 with respect to y_2 when the step is applied to r_1, but x^2 presents a better response than x^1 with respect to y_2 and y_1 (better in the sense that it displays less overshoot) when the step is applied to r_2. Therefore, it is not clear at all which solution should be chosen. In these cases, the decision-maker will have to include new criteria or preferences, in order to make the decision. Anyway, note that the nearly optimal solution x^2*could* ultimately be preferred over the optimal one x^1. This evidences that it would not have been wise to discard x^2 in the search stage and proofs the convenience of finding the nearly optimal solutions besides the optimal ones. This situation is very common when the objective functions are defined as an aggregation of several subobjectives, as is the case here. Additionally, it is worth noting that if this MOP had been addressed with a traditional approach and eight objectives (one for each term in 32, without adding objectives)—which is the ideal approach if one wants to find all the optimal solutions, although impractical in the decision-making stage—then x^2 would not be dominated by x^1 and both x^1 and x^2 could belong to the Pareto front. This is an additional evidence of the relevance of x^2, which nevMOGA has been able to identify.

Next, we assess the robustness of every solution found by the algorithm, that is to say, how well each controller maintains its performance when identification errors are contemplated. This analysis is typical in control design. In order to do that, first, we get 50 sets of plant parameter (gains, time constants, and time delays) by generating random values for each parameter within a range of ±20%. Each of this sets represents a deviation from the nominal plant. Then, we evaluate the performance (f_1 and f_2) of each controller for each one of those deviations. Including an additional objective which measures robustness and computing it in the optimization stage, along with the two other objectives, would have been very expensive computationally. This is why this robustness analysis is shifted to the decision-making stage. As an example of the results of this robustness analysis, Figure 16 shows the boundaries of the performance degradation (determined by the values of f_1 and f_2 when they are evaluated for each one of the 50 sets of plant parameters) experienced by two controllers (x^3 and x^4). Note that x^4 (a nearly optimal solution) experiences less degradation than x^3. This is a very valuable information and could make the decision-maker opt for x^4 instead of x^3.

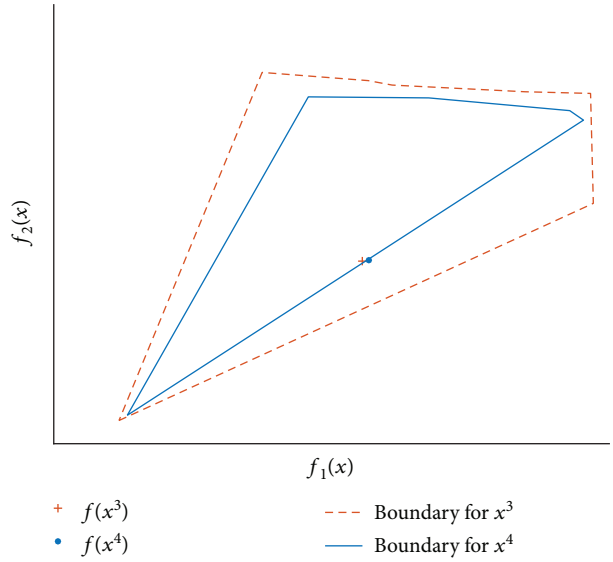


FIGURE 16: Boundaries of the performance degradation for x^3 (optimal solution) and x^4 (nearly optimal solution) when the plant parameters vary in ±20% from their nominal values. It must be highlighted that a traditional MOP approach would not have found the solution x^4.

Let us look at one last proof of the convenience of considering the nearly optimal solutions. If we compare x^5 (optimal solution) and x^6 (nearly optimal solution), we see that they have a similar performance with respect to both f_1 and f_2. However, x^6 has lower values (in absolute value) of the gains Kc_1 and Kc_2. From a design perspective, this fact may be an advantage, since this solution will present less sensibility to any measurement noise in the outputs of the plant and, therefore, it could be preferred over the optimal solution.

5. Conclusions

In this paper, we have presented a new multiobjective genetic algorithm, nevMOGA, which is capable of characterizing not only the Pareto set but also the set of nearly optimal solutions nondominated in their neighborhood. We think that this algorithm will be useful for designers who want to get a manageable number of solutions without neglecting the existing diversity in the characteristics of the solutions. In comparison with other algorithms that only return the Pareto front, our algorithm provides, in addition, new alternatives to be analyzed in the decision-making stage. Given that these solutions have a very similar performance to the optimal ones and significantly different values in the parameter space, they will be regarded by a designer adopting the specified approach as solutions worth studying. This is why obtaining these solutions, which we have called *potentially useful* solutions, can be very valuable in many multiobjective optimization problems.

nevMOGA has been compared with two other algorithms (random search and exhaustive search) on two benchmark problems, in order to assess its performance. It has not been compared to any other because no genetic algorithm of similar characteristics (one which is capable of finding nearly optimal solutions nondominated in their neighborhood) is available in the literature. In order to perform this comparison, the metric Δ_p has been employed, which measures the approximation quality of the outcome set of an algorithm to the target set $P^*_{Q,n}$ (the set of solutions to be found) in the objective space and in the parameter space. In addition, a statistical analysis has been conducted. The results reveal the outperformance of nevMOGA in characterizing the set of optimal and nearly optimal solutions in both spaces.

nevMOGA has also been applied to a real design problem in the field of process control engineering. This design problem consists of optimally adjusting the parameters of two PI controllers (K_c and T_i) which operate a plant with two inputs and two outputs. The control performance is evaluated by two widely used, conflicting metrics: f_1, which measures the setpoint tracking performance of the control system, and f_2, which measures the control effort. Each of these objective functions has been formulated as a sum of four subobjectives (see 32), which means that eight different objectives have been incorporated in the problem definition. Dealing with eight independent objective functions would have complicated enormously the analysis of the Pareto front and, consequently, the decision-making stage. This is why the eight objectives have been aggregated to form only two objective functions, whose Pareto front can be plotted on a two-dimensional graph and, thus, easily analyzed.

Aggregating objectives has, however, a major drawback if the optimization algorithm searches only for the optimal solutions: the resulting set of Pareto optimal solutions will lack many solutions that may be interesting and useful for the designer and, in some cases, even preferable to the optimal ones. nevMOGA, as it searches for nearly optimal solutions nondominated in their neighborhood too, is capable of finding those valuable solutions which are neglegted by a classical optimization algorithm. This means that the decision-maker will have a greater number of alternatives, all of them worth considering. For instance, if we compare x^2 (a nearly optimal solution) and x^1 (an optimal solution), we see that they have similar performances ($f(x^2) \simeq f(x^1)$) but their corresponding system responses are different, since they lie in distinct neighborhoods. The designer could choose the nearly optimal solution x^2 instead of the optimal solution x^1, since the former achieves a better system response when the step is applied to the setpoint r_2. A comparison between x^5 (optimal solution) and x^6 (nearly optimal solution) leads to the same conclusion: they have similar performances ($f(x^6) \simeq f(x^5)$) but x^6 has lower values of Kc_1 and Kc_2 (the gains of the PI controllers) than x^5 and, therefore, x^6 will be less sensitive to the measurement noises in the outputs, which is a considerable advantage in some applications. This could make the designer prefer x^6 to x^5.

For this same design problem, a robustness analysis has been performed on the optimal and nearly optimal solutions obtained in the optimization stage. The results of this analysis show how a nearly optimal solution (x^4) can be more robust than an optimal one (x^3) while providing a similar performance with respect to the objective

function ($f(x^3) \simeq f(x^4)$). This is due to the fact that they are nonneighboring solutions and, therefore, likely to present distinct characteristics. If we had included robustness as a metric in the definition stage in the same way as we did with f_1 and f_2, the computational cost would have increased drastically (around 50 times higher).

These examples demonstrate the pertinence of the criterion assumed and, consequently, the great convenience of taking into account the potentially useful solutions, that is to say, the nearly optimal solutions that are not dominated in their neighborhood. This approach is specially advisable when the MOP presents multimodal solutions, either because they are intrinsic to the problem or because they have arisen artificially as a result of aggregating several objectives into one single-objective function. Likewise, having the nearly optimal solutions available represents an advantage when there exist objectives which cannot be incorporated within the optimization process due to its high computational cost. When this happens, these objectives are excluded from the optimization stage and evaluated afterwards only for the optimal and nearly optimal solutions resulting from the search.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the Ministerio de Economía y Competitividad (Spain) Grant numbers DPI2015-71443-R and FPU15/01652, by the local administration Generalitat Valenciana through the project GV/2017/029, and by the National Council of Scientific and Technological Development of Brazil (CNPq) through the grant PQ-2/304066/2016-8.

References

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 1, Kluwer Academic Publishers, 1998.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16, John Wiley & Sons, 2001.

[3] G. Reynoso-Meza, J. Sanchis, X. Blasco, and M. Martinez, "Evolutionary algorithms for PID controller tuning: current trends and perspectives," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 10, no. 3, pp. 251–268, 2013.

[4] G. Reynoso-Meza, J. Sanchis, X. Blasco, and S. García-Nieto, "Physical programming for preference driven evolutionary multi-objective optimization," *Applied Soft Computing*, vol. 24, pp. 341–362, 2014.

[5] J. Sanchis, M. Martinez, and X. Blasco, "Integrated multiobjective optimization and a priori preferences using genetic algorithms," *Information Sciences*, vol. 178, no. 4, pp. 931–951, 2008.

[6] J. Branke, K. Deb, and K. Miettinen, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, vol. 5252, Springer Science & Business Media, 2008.

[7] K. Deb and A. Saha, "Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, pp. 447–454, Portland, Oregon, USA, July 2010.

[8] J. Liang, C. Yue, and B. Qu, "Multimodal multi-objective optimization: a preliminary study," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2454–2461, Vancouver, BC, Canada, July 2016.

[9] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 601–614, 2012.

[10] P. Loridan, "ε-Solutions in vector minimization problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 2, pp. 265–276, 1984.

[11] D. J. White, "Epsilon efficiency," *Journal of Optimization Theory and Applications*, vol. 49, no. 2, pp. 319–337, 1986.

[12] M. Vasile and M. Locatelli, "A hybrid multiagent approach for global trajectory optimization," *Journal of Global Optimization*, vol. 44, no. 4, pp. 461–479, 2009.

[13] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[14] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm," *TIK-Report*, vol. 103, 2001.

[15] J. M. Herrero, S. García-Nieto, X. Blasco, V. Romero-García, J. V. Sánchez-Pérez, and L. M. Garcia-Raffi, "Optimization of sonic crystal attenuation properties by ev-MOGA multiobjective evolutionary algorithm," *Structural and Multidisciplinary Optimization*, vol. 39, no. 2, pp. 203–215, 2009.

[16] V. Pareto, *Manual of Political Economy*, A. M. Kelley, New York, NY, USA, 1971.

[17] O. Schütze, C. A. C. Coello, and E. G. Talbi, "Approximating the ε-efficient set of an MOP with stochastic search algorithms," in *Lecture Notes in Computer Science*, pp. 128–138, Springer, 2007.

[18] O. Schütze, M. Vasile, and C. A. Coello Coello, "Computing the set of epsilon-efficient solutions in multi-objective space mission design," *Computing*, vol. 8, no. 3, pp. 53–70, 2011.

[19] J. M. Herrero, *Identificacion Robusta de Sistemas no Lineales mediante Algoritmos Evolutivos, [Ph.D. thesis]*, Editorial Universitat Politècnica de València, 2006.

[20] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[21] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello, "Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, 2012.

[22] X. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark functions for CE2013 special session and competition on niching methods formultimodal function optimization," in *RMIT*

University, Evolutionary Computation and Machine Learning Group, Tech. Rep, Australia, 2013.

[23] O. Schütze, C. A. C. Coello, E. Tantar, and E. G. Talbi, "Computing a finite size representation of the set of approximate solutions of an MOP," 2008, http://arxiv.org/abs/0804.0581.

[24] L. E. Toothaker, *Multiple Comparison Procedures*, Sage, 1993.

[25] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[26] T. Mc Avoy, Y. Arkun, R. Chen, D. Robinson, and P. D. Schnelle, "A new approach to defining a dynamic relative gain," *Control Engineering Practice*, vol. 11, no. 8, pp. 907–914, 2003.

[27] J. M. Herrero, G. Reynoso-Meza, C. Ramos, and X. Blasco, "Considerations on loop pairing in MIMO processes. A multi-criteria analysis," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4454–4459, 2017.

[28] X. Blasco, J. M. Herrero, J. Sanchis, and M. Martínez, "A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization," *Information Sciences*, vol. 178, no. 20, pp. 3908–3924, 2008.

[29] X. Blasco, J. M. Herrero, G. Reynoso-Meza, and M. A. Martínez Iranzo, "Interactive tool for analyzing multiobjective optimization results with level diagrams," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*, pp. 1689–1696, Berlin, Germany, July 2017.