Qual Quant (2010) 44:315–332
DOI 10.1007/s11135-008-9201-z

Robust design in multivariate systems using genetic algorithms

Hector Allende · Daniela Bravo · Enrique Canessa

Published online: 9 September 2008

Abstract This paper presents a methodology based on genetic algorithms, which finds feasible and reasonably adequate solutions to problems of robust design in multivariate systems. We use a genetic algorithm to determine the appropriate control factor levels for simultaneously optimizing all of the responses of the system, considering the noise factors which affect it. The algorithm is guided by a desirability function which works with only one fitness function although the system may have many responses. We validated the methodology using data obtained from a real system and also from a process simulator, considering univariate and multivariate systems. In all cases, the methodology delivered feasible solutions, which accomplished the goals of robust design: obtain responses very close to the target values of each of them, and with minimum variability. Regarding the adjustment of the mean of each response to the target value, the algorithm performed very well. However, only in some of the multivariate cases, the algorithm was able to significantly reduce the variability of the responses.

Keywords Robust design · Taguchi methods · Genetic algorithms · Desirability functions

1 Introduction

Robust design is a technique which tries to adjust some parameters of a system (control factors) so that some output variables of the system (responses) get as close as possible to corresponding target values, even in the presence of factors which cannot be controlled (noise

H. Allende · D. Bravo · E. Canessa (✉)
Facultad de Ciencia y Tecnología, Universidad Adolfo Ibáñez, Balmaceda, 1620 Vina del Mar, Chile
e-mail: ecanessa@uai.cl

D. Bravo
e-mail: dbravo@uai.cl

H. Allende
Departamento de Informática, Universidad Técnica Federico Santa María, CP 110, Valparaiso, Chile
e-mail: hallende@uai.cl

factors). In the case of production processes, the less sensitive the process is to noise factors, the smaller the variability the products manufactured by the process will have around the desired target values, which will increase their quality.

To accomplish those goals, robust design prescribes the use of two methods: tolerance design and parameter design. Tolerance design seeks to determine how much variability of the input parameters to the system will be allowed, balancing the levels of tolerance that may be obtained, against the benefits of achieving those levels. Parameter design is a procedure which tries to reduce variability at a low cost (Allende et al. 2005). The fundamental concept behind parameter design is the classification of control factors in dispersion factors (x_D), which mainly affect the variability of the response, and adjustment factors (x_S), which can be used to adjust the mean of the response to the target value. When we wish to obtain a response as close as possible to the target value, we use a nominal-the-better (NTB) quality characteristic (Taguchi 1991).

According to Taguchi's methods, the NTB quality characteristic has associated a quality loss function that may be expressed as $L(y) = kMSD$, that is, the multiplication of a quality loss coefficient (k) with the mean square deviation (MSD). The MSD is a measure of the difference between the target value of the response (T) and the obtained response (y), averaged over n observations, as can be seen in the next expression:

$$L(y) = k\ MSD = k\left[\frac{1}{n} * \sum_{i=1}^{n} (y_i - T)^2\right] = k\left[\sigma^2 + (\mu - T)\right] \tag{1}$$

We can see from the above expression, that to reduce the loss, we must minimize MSD. Equivalently, we may reduce variance and get the mean closer to the target value. Taguchi also uses another measure of quality, in which he uses the concept of signal-to-noise ratio, borrowed from the telecommunications field (Taguchi 1991). This measure corresponds to:

$$SNR \approx 10 \log\left(\frac{\overline{y}^2}{s^2}\right), \tag{2}$$

where $\overline{y}^2$ is the square of the average of the observations of the response of the system (y) and s^2 is the sample variance. Note that the SNR is defined so that we always try to maximize it. Thus, the parameter design attempts to find the combination of control factor levels, which will minimize MSD or maximize SNR, and at the same time, will obtain a response as close as possible to the target value. For doing that, Taguchi suggests using a two-stage optimization process (Allende et al. 2005):

(a) Determine x_D^* so that $\min_{x_D}(MSD(x_D))$, or, $\max_{x_D}(SNR(x_D))$, where x_D^* is the vector of optimum dispersion factors.
(b) Determine x_S^* so that $\min_{x_S} |\mu(x_S) - T|$, where x_S^* is the vector of optimum adjustment factors.

From the above expressions, we can see that Taguchi's optimization process does not assure that the goal of parameter design will be achieved. Attaining a local optimum at each stage, does not guarantee that we will obtain the overall optimum of minimizing the loss function. For that to happen, the mean and variance of the response must be independent of each other, so that we can decrease variance and then adjust the mean toward a target value, without increasing the variance. Generally, that condition is only partially met or is fulfilled by specific types of systems. That limits the applicability of Taguchi's process (Leon et al. 1987).

On the other hand, the two-stage optimization process stated by Taguchi can be extended to systems which have more than one response. For doing that, we must find expressions that allow us to calculate the quality loss function, the mean square deviation, and the signal-to-noise ratio for multivariate systems. In the case of k-dimensional systems with $k = 2, 3$ (bivariate and trivariate systems), Maghsoodloo and Chang (2001) developed the corresponding expressions. However, that work shows the theoretical difficulties of doing that. The difficulty to solve the equations of the theoretical approach increases as the dimensionality of the system increases (k bigger than 3).

One alternative to that approach if the system has a small number of factors, is to separately optimize each of the responses and then determine the regions where the target values are simultaneously achieved for each response (Myers and Montgomery 2002). However, for problems with a large number of factors and responses, that analysis becomes increasingly difficult or even impossible to carry out (Pignatiello 1988; Ortiz et al. 2004). Because of that, we can use the method proposed by Del Castillo et al. (1996), who suggest combining all of the system's responses in one expression, attaining a single target value, which they call the *desirability function* or *FD*, as we will explain in Sect. 2.2. Using that approach, we can then use a genetic algorithm to optimize the system's responses (Ortiz et al. 2004).

2 Proposed algorithm

Given the difficulties of applying robust design to multivariate systems, we propose a method which uses the desirability function for combining the different responses of a multivariate system in a single scalar function to optimize (Del Castillo et al. 1996). Then, we employ a genetic algorithm to find an adequate combination of control factors, which gets the responses as close as possible to their corresponding target values, and at the same time, decreases as much as possible the variability of each response. An additional advantage of our approach is that it works with multivariate systems which may have a large number of factors and levels for each factor (Bravo 2005). In the following subsections, we explain the details of the desirability function and genetic algorithm (GA) that we developed.

2.1 Representation and codification of the solutions in the GA

In a robust design experiment, we will have k control factors that may take s different levels (values). One solution (chromosome) will be composed of a combination of different levels of each factor, which corresponds to a particular treatment of the experiment.

Let f_{lj} be the factor j of chromosome l, with $j = 1, 2, \ldots, k$ and $l = 1, 2, \ldots, N$. Each f_{lj} can take the value of a given level of a factor, according to an integer codification, that is $f_j = \{1, 2, \ldots, s\}$. One chromosome (or solution) is expressed as a row vector (see Eq. 3). The matrix representing the total population of solutions X will be composed of N chromosomes (see Eq. 4).

$$x_l = [f_{l1}, f_{l2}, \ldots, f_{lk}] \tag{3}$$

$$X = [x_1, x_2, \ldots, x_N]^T \tag{4}$$

For the initial population ($i = 0$), the population matrix will correspond to the experiment carried out and will evolve both in its content as well as in its size, according to the iterations performed by the algorithm. In the ith generation we will have combinations of treatments

represented by expression (5) (from now on we will interchangeably use the terms solutions, chromosomes, combination of treatments or simply combinations). The population matrix of the ith generation is represented by expression (6), where each x_l^i represents the combination of treatments l of the corresponding generation.

$$x_l^i = \left[x_{l1}^i, x_{l2}^i, \ldots, x_{lk}^i\right] \tag{5}$$

$$X^i = \left[x_1^i, x_2^i, \ldots, x_N^i, \ldots\right]^T \tag{6}$$

To build the $i+1$ generation, we need to use the genetic operators of reproduction, crossover and mutation (Holland 1974). But before defining those operators, we need to describe the fitness function which we will optimize.

2.2 Fitness function of the algorithm

Each ith generation will have a combination of treatments x_l^i which will give a corresponding value for the fitness function $\phi\left(x_l^i\right)$. If we use a fitness function based on the *MSD*, then minimizing the *MSD* is equivalent to maximizing its reciprocal:

$$\text{Fitness function} = \phi\left(x_l\right) = \left(\frac{1}{MSD_l}\right) = \frac{1}{\sigma_{lr}^2 + \left(\overline{y}_r\left(x_l\right) - T_r\right)^2} \tag{7}$$

On the other hand, if we use the SNR, then:

$$SNR_l = 10\log\left(\frac{\left(\overline{y}_r\left(x_l\right)\right)^2}{\left(\sigma_{lr}\right)^2}\right) \Rightarrow \text{Fitness function } = \phi\left(x_l\right) = \frac{\left(\overline{y}_r\left(x_l\right)\right)^2}{\left(\sigma_{lr}\right)^2} \tag{8}$$

In multivariate systems we will use a total fitness function that will combine all the functions $\phi\left(x_l\right)$ of each response.

Let $y_{mr}\left(x_l\right)$ be replication m corresponding to the r response of the combination or chromosome x_l, with $m = 1, 2, \ldots, M$ $r = 1, 2, \ldots, R$ and $l = 1, 2, \ldots, N$.

Following the work of Ortiz et al. (2004), the total fitness function for multivariate systems will be expression (9) (from now on we eliminate the use of the superindex i, which represents the generation to facilitate the understanding of the expressions), which consists of a desirability function $D(\phi(x_l))$ and a penalty function $P_l((y(x_l)))$:

$$D_l^*\left(x_l\right) = D_l\left(\phi_r\left(x_l\right)\right) - P_l\left(y\left(x_l\right)\right) \tag{9}$$

where $y\left(x_l\right)$ is the generic form of designating all the responses of the treatment combination x_l. Thus, the penalty function will depend on all the replications of all the responses of the system.

Each element of expression (9) can be decomposed into expressions (10) and (11) according to Ortiz et al. (2004):

$$D_l\left(\phi\left(x_l\right)\right) = \left(d_{1l}\left(\phi_1\left(x_l\right)\right)^* d_{2l}\left(\phi_2\left(x_l\right)\right)^* \ldots^* d_{Rl}\left(\phi_R\left(x_l\right)\right)\right)^{\frac{1}{R}} \tag{10}$$

$$P_l\left(y\left(x_l\right)\right) = \left[\left(p_{l1}\left(y_l\left(x_l\right)\right)^* p_2\left(y_2\left(x_l\right)\right)^* \ldots^* p_R\left(y_R\left(x_l\right)\right)\right)^{\frac{1}{R}} - c\right]^2 \tag{11}$$

where $y\left(x_l\right)$ is the generic form of designating all the responses of the treatment combination x_l.

Moreover, each element of the penalty function (11) can be expressed as (Ortiz et al. 2004):

$$p_{r1}\left(y_l\left(x_l\right)\right) = \begin{cases} c + \left(\frac{L_r - \overline{y}_r(x_l)}{H_r - L_r}\right), & \overline{y}_r\left(x_l\right) \leq L_r \\ c, & L_r \leq \overline{y}_r\left(x_l\right) \leq H_r \\ c + \left(\frac{\overline{y}_r(x_l) - H_r}{H_r - L_r}\right), & \overline{y}_r\left(x_l\right) \geq H_r \end{cases} \tag{12}$$

No matter the form we use to determine the tolerance limits of a response y_r, each of them has a target value T_r and a lower and upper limit given by L_r and H_r, respectively, in which $L_r < T_r < H_r \quad \forall r$. The constant c avoids the penalty function becoming zero if infeasible cases arise and is assigned a value of 0.0001, which does not influence the value of the final solution calculated by the algorithm according to Ortiz et al. (2004).

The feasibility of a solution being within the tolerance limits selected for the system for all of the responses $(L_r \leq \overline{y}_r\left(x_l\right) \leq H_r)$, determines the feasibility of chromosome x_l.

Regarding the desirability function, it must have a lower and upper limit for each of the elements $d_{rl}(\phi_r(x_l))$ [see expression (10)], called the desirability or acceptability limits (Ortiz et al. 2004). Expression (13) represents the desirability function, where b_r corresponds to the most desirable case and a_r to the least desirable case, but even so a feasible one:

$$d_{rl}\left(\phi_r\left(x_l\right)\right) = \begin{cases} 0, & \phi_r\left(x_l\right) \leq a_r \\ \frac{\phi_r(x_l) - a_r}{b_r - a_r}, & a_r \leq \phi_r\left(x_l\right) \leq b_r \\ 1, & \phi_r\left(x_l\right) \geq b_r \end{cases} \tag{13}$$

The value of those two parameters (b_r, a_r) depends on the fitness function we use (*MSD* or *SNR*). If the fitness function is based on the *MSD*, then according to Eq. 7, we will obtain a maximum value for the fitness function when the denominator of expression (7) reaches a minimum. This will happen when the response equals the target value and the variability of the response is a minimum. Thus, parameter b_r may be expressed by Eq. 14, using the minimum variability attained in the treatment combination of the initial experiment performed:

$$b_r = \frac{1}{\left(\left(\sigma_{lr}\right)_{\min}\right)^2}, \quad \forall r, l \tag{14}$$

The expression (14) will tend to infinity every time the variability tends to zero. To avoid that, we may simply change Eq. 14, as shown in Eq. 15. In (15), when the variability tends to zero, parameter b_r tends to one, which is the most desirable case according to the definition of the desirability function:

$$b_r = \frac{1}{\left(1 + \left(\sigma_{lr}\right)_{\min}\right)^2}, \quad \forall r, l \tag{15}$$

In the least desirable and feasible case, the maximum distance between the obtained value and the target one of a response, will be found when the response takes the value of one of the tolerance limits. Considering the limit which is farther away from the target value (T_r), and taking into account the maximum variability observed in the treatment combinations of the initial experiment, parameter a_r may be calculated using expression (16):

$$a_r = \frac{1}{\left(\left(\sigma_{lr}\right)_{\max}\right)^2 + \left(\max\left\{\left(H_r - T_r\right), \left(T_r - L_r\right)\right\}\right)^2}, \quad \forall r, l \tag{16}$$

On the other hand, if the fitness function is based on the *SNR*, then according to expression (8), the most desirable case will be found when the variability of the response is a

minimum and following the same reasoning as before, parameter b_r may be calculated as shown by (17):

$$b_r = \frac{\left(\overline{y}_r\right)^2}{\left(1 + (\sigma_{lr})_{\min}\right)^2}, \quad \forall r, l \tag{17}$$

The least desirable case will be found at maximum variability and thus:

$$a_r = \frac{\left(\overline{y}_r\right)^2}{\left((\sigma_{lr})_{\max}\right)^2}, \quad \forall r, l \tag{18}$$

In both cases, the selected numerator is the average of all the replications of all the original treatment combinations of response r, $\overline{y}_r$.

We must note that in the calculation of the fitness value for each treatment combination, we need to know the response of each of them. However, it may happen that some of those responses are not known, since the corresponding treatment combination was not part of the original experiment. Thus, since the performed experiment did not consider those treatments, we will not have the corresponding data. Moreover, it can happen that those treatments are precisely the ones which may obtain a good solution, even better than the treatments really used in the actual experiment carried out. Thus, in order to evaluate those treatments, we need to estimate the responses we would have obtained, had we actually tried those treatments in the experiment. That involves estimating both the mean and standard deviation of the responses corresponding to those treatments.

For estimating the standard deviation, we calculate it for each response, considering all the replications of all the treatments tried in the original experiment. In that form, we will obtain a conservative estimate of the standard deviation (bigger) than if we calculate it using the mean of the replications for each response.

For estimating the mean, we use two simple methods:

- Method one: mean of the observed responses (of the collected data). Each of the missing values will be replaced by the mean of the observations corresponding to the experiment that was carried out (Yuan 2006).
- Method two: estimation of the optimal response according to Taguchi's method. It consists in estimating the response using the mean of the responses and the values of the main effects of each level of the factors (Allende et al. 2005), as expression (19) shows:

$$y_{opt} = A_1 + A_2 + \cdots + B_1 + B_2 + \cdots + \overline{y}^* \,(\text{Number\_factors} - 1)\,, \tag{19}$$

where $\overline{y}$ is the mean of all the collected data, A_1 is the mean of all the observations corresponding to level 1 of factor A, A_2 is the mean of all the observations corresponding to level 2 of factor A, and so on for the other levels and factors considered in the design of the experiment.

2.3 Selection stage of the algorithm

The goal of this stage is to select the treatment combinations that will survive and be part of the next generation. Let q_l be the probability that combination l will pass to the next generation, with $l = 1, 2, \ldots, N$, that is, the selection probability of x_l^i. The corresponding cumulative probability is denoted by Q_l. Then, we can write:

$$q_l = \frac{D_l^*\left(x_l^i\right)}{\sum_{l=1}^{N} D_l^*\left(x_l^i\right)} \tag{20}$$

$$Q_l = \sum_{h=1}^{l} q_h \tag{21}$$

This stage consists of the following steps:

- Generate N random numbers sampled from an uniform distribution $U\,[0, 1]: u_1, u_2, \ldots u_N$
- For each u_l find h so that $Q_{h-1} < u_l \leq Q_h$, with $l = 1, 2, \ldots, N$ and $h = 1, 2, \ldots, N$
- Select x_h^i and build the new generation $X^{i*} = \left[x_1^{i*}, x_2^{i*}, \ldots, x_N^{i*}\right]^T$

Some of the combinations of the ith generation may be selected more than once and others never. At the end of the selection stage no new combination will be created; only some bad combinations will be eliminated (with low q_l).

This form of selecting the chromosomes which will be part of the next generation corresponds to a roullette or probabilistic selection, in which the individuals with a higher value for the fitness function will have a higher probability of being selected. The infeasible chromosomes have zero probability of being selected.

2.4 Crossover stage of the algorithm

The goal of this stage is to apply the crossover operator to the population $\left\{x_1^{i*}, x_2^{i*}, \ldots, x_N^{i*}\right\}$. Let p_c be the crossover probability, or the probability that each combination undergoes a crossover operation. This probability is set by the user and depends on the optimization problem being addressed (Haupt and Haupt 2004).

The steps of this stage are the following:

- Generate N random numbers from a uniform distribution $U\,[0, 1]: u_1, u_2, \ldots u_N$
- For each u_l, with $l = 1, 2, \ldots, N$
  - If $u_l > p_c$: do not perform crossover (x_l^{i*} does not change, just go on to the mutation stage)
  - If $u_l \leq p_c$: select x_l^{i*} and perform crossover:
    - Select at random another combination $x_{l'}^{i*}$ (different from the former one)
    - Generate an integer random number in the interval $[1, k]$, which will indicate the factor on which to perform the crossover (remember that we have k factors).

The above steps mean that, if the algorithm performs crossover, it will select two parent chromosomes x_l^{i*} and $x_{l'}^{i*}$, as shown below:

$$\begin{array}{cc} \text{Parent\_1}: & \text{Parent\_2}: \\ x_l^{i*} = [f_{l1}, f_{l2}, \ldots, f_{lcol}, \ldots, f_{lk}] & x_{l'}^{i*} = [f_{l'1}, f_{l'2}, \ldots, f_{l'col}, \ldots, f_{l'k}] \end{array}$$

Suppose that the factor chosen to perform the crossover is $E = col$, with $col \leq k$. In that case, the two descendants (children) will be:

$$\begin{array}{cc} \text{Child\_1}: & \text{Child\_2}: \\ x_l^{i**} = [f_{l1}, f_{l2}, \ldots, f_{l'col}, \ldots, f_{l'k}] & x_{l'}^{i**} = [f_{l'1}, f_{l'2}, \ldots, f_{lcol}, \ldots, f_{lk}] \end{array}$$

This type of crossover operation corresponds to a one-point crossover. From the two original treatment combinations, the algorithm will generate two children, that is the ratio

parent:children is 1:1 We arbitrarily decided to retain the parents as part of the population, since they already have a high value fitness function. Remember that the parents were already chosen to breed in the selection stage. Additionally, as we will explain in Sect. 2.6, it is useful to have as many good solutions as possible, so that the experimenter can choose the most convenient one to implement. However, to keep the size of the population within a manageable value, we use a limit on that value as the stopping criterion of the algorithm.

2.5 Mutation stage of the algorithm

The mutation operation is performed bit by bit (in our case factor by factor) in all the combinations created and kept after the crossover stage. Let p_t be the mutation probability, which is the probability that each factor undergoes mutation and is set by the user. The value of this probability depends on the optimization problem being addressed (Haupt and Haupt 2004).

For each combination x_l^{i**} , with $l = 1, 2, \ldots, N$, perform the following steps:

- Generate k random numbers from an uniform distribution $U\,[0, 1] : u_1, u_2, \ldots u_k$
- For each u_h, with $h = 1, 2, \ldots, k$:
  - If $u_h \leq pt$: mute the hth factor of x_l^{i**}
  - If $u_h > pt$: the hth factor remains unaltered

The mutation operation replaces the level of factor h with another level of the same factor. For a two-level factor, there is only one mutation possible: if the factor is at level one, it will change to level two and vice-versa. On the other hand, if the factor has more than two levels ($s \geq 3$), the level of the factor will change to another level selected at random. The algorithm keeps the original chromosome along with the mutated one.

2.6 Stopping criterion of the algorithm

The algorithm must have a stopping criterion. This criterion may be to achieve a certain difference between the value of the fitness function of a generation and the next one that should be smaller than ξ, or that the difference between the target value and the obtained solution is smaller than ξ (Haupt and Haupt 2004). Another criterion may be that the algorithm performs a fixed number of iterations or has achieved a given population size (Haupt and Haupt 2004). This is the criterion we use in this first version of the algorithm. In future work we will explore the benefits of using other stopping mechanisms.

Once the stopping criterion is achieved, we will have a set of possible solutions, since the genetic algorithm calculates good solutions, not necessarily the best one. From those solutions, the experimenter must select the one to implement in the real system. This decision may be based on different reasons. For example, the experimenter may consider the cost of implementing each solution in the real system or the time it will take to do so, among others.

Before showing the application of the algorithm to some real and simulated systems, we would like to emphasize that the use of the algorithm is part of the traditional Taguchi's methodology of robust design. Thus, as Fig. 1 shows, it shares many of the steps of robust design. Note that the method uses the same steps prescribed by robust design for designing the experiment and executing it in the real system. Once the data is collected, we apply the algorithm for obtaining the best treatment combinations, i.e. those which will adjust the responses of the system to their corresponding target values and simultaneously

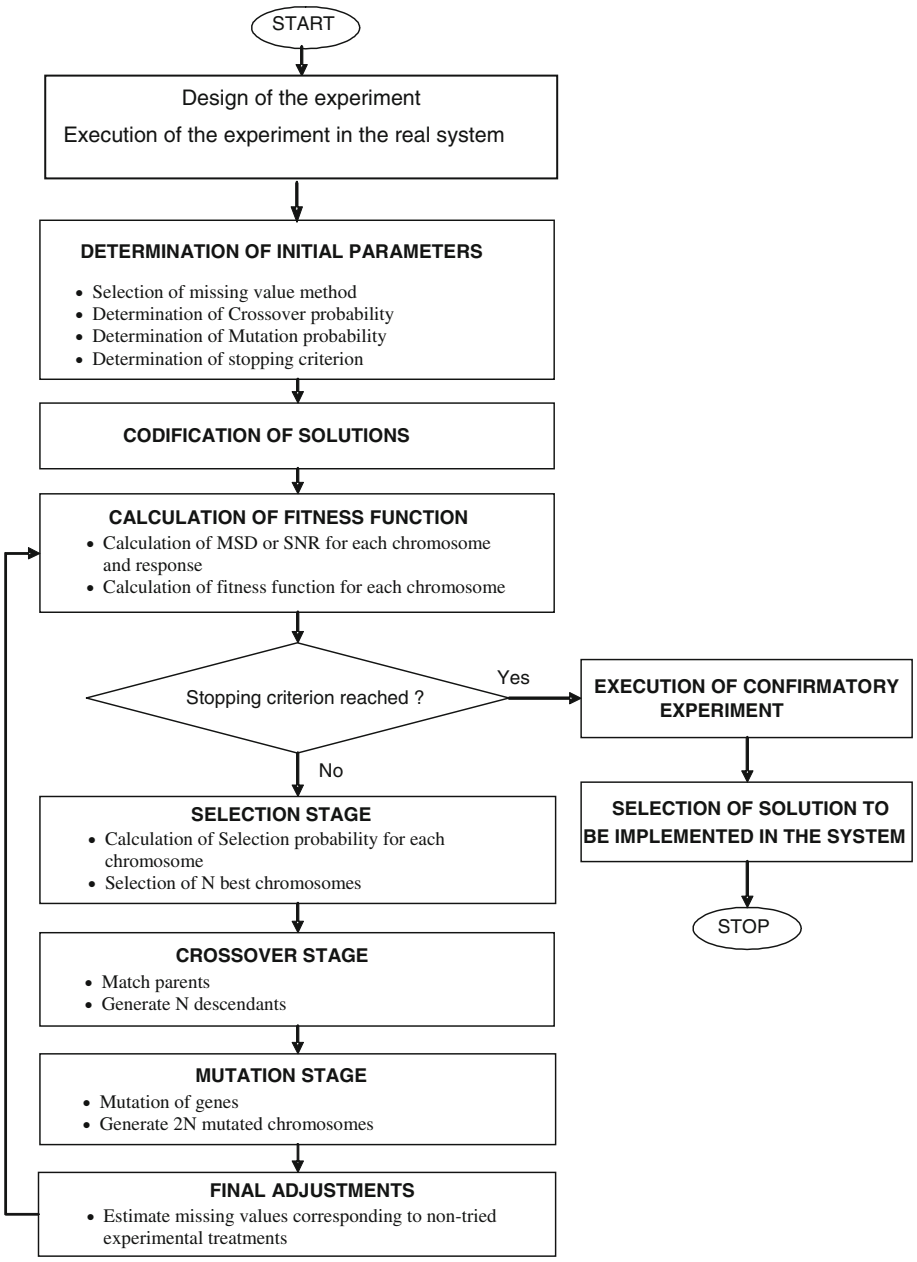

Fig. 1 Application of the algorithm to robust design in multivariate systems

decrease variability of the responses. Then, following the steps of robust design, one should carry out confirmation runs for validating the solution. Finally, the experimenter decides which will be the solution to be implemented, per the considerations outlined in the previous paragraph.

3 Application of the algorithm

To assess the soundness and performance of our method and algorithm, we used two different case studies. The first one corresponds to an application of robust design in a real univariate system, described in Vandenbrande (2000). The second case corresponds to a simulated multivariate system and its simulator is described in Appendix A. In Sects. 3.1, 3.2 and 3.3 we will discuss why we selected those cases to test the algorithm.

For both cases, we applied the algorithm using the fitness function based on the MSD as well as on the SNR. We also used both methods for estimating the missing values of the responses: mean of the observed values and estimation by Taguchi's method. Table 1 shows each of the possible four combinations along with the other parameters of the algorithm that we kept fixed.

3.1 Results obtained for the univariate real system

The case we use for this part of the study corresponds to the application of robust design to find the best treatment combination of four control factors for adjusting the automatic body paint process in a car manufacturing plant (Vandenbrande 2000). In this process, it is very important to attain a specific width of the each strip that is sprayed with paint. A narrow strip will leave parts of the car body with no paint and overspraying will result in waste of paint. This also means that the process should be very stable, i.e. the width of the paint strip should exhibit a small variability. In this study, the engineers and technicians identified four control factors each having three levels: type of spray gun used, speed of paint flow, fan airflow and atomizing airflow. The noise factors identified are three each with two levels: paint color, input air pressure and paint viscosity. For details see Vandenbrande (2000). The design of the experiment called for the use of an orthogonal array $L_9(3^4)$ for the control factors and a $L_4(2^3)$ for the noise factors. Given the small number of control and noise factors and levels per factor, the optimal solution (treatment) can be manually determined. However, we applied the algorithm since the data corresponds to a real system and since we knew the optimal solution, we could check the solution provided by our algorithm. The tolerance limits and target value we set up in the algorithm were: $L_r = 36.5$, $H_r = 41.5$ and $T_r = 40.0$ cm. The target value corresponds to the ideal paint strip width and the lower and upper tolerance limits were

Table 1 Alternatives and parameters used in the application of the algorithm

Name of alternative	Fitness function based on	Estimation method for missing values
Alternative 1	MSD	Mean of observed values
Alternative 2	MSD	Taguchi method
Alternative 3	SNR	Mean of observed values
Alternative 4	SNR	Taguchi method
Constant parameters	**Value**	**Observations**
Penalty function constant (c)	0.0001	Proposed by Ortiz et al. (2004)
Crossover probability (p_c)	0.3	Values set for initial experimentation. One should study their impact on the performance of the algorithm in future work
Mutation probability (p_t)	0.05	
Stopping criterion	Size of final population	Between 3,000 and 4,000 chromosomes

arbitrarily fixed. The influence of those parameters over the performance of the algorithm should be studied in future work. The data used were taken from Vandenbrande (2000).

The algorithm gave the same optimal solution that can be manually calculated and corresponds to the treatment combination $A = 2$, $B = 3$, $C = 1$ and $D = 2$ (chromosome $x = [2\ 3\ 1\ 2]$). For that combination, the width of the paint strip is 41.025 cm with a standard deviation of 1.439 cm. The width is only 2.56% broader than the target value. This result was obtained using all of the four alternatives for fitness function and missing value estimation method (see Table 1). Thus, we can say that the algorithm is effective in dealing with this small-size type of problem.

Additionally, the algorithm also delivered other feasible solutions. One of them corresponds to chromosome $x = [1\ 2\ 2\ 2]$, which generates a paint strip width of 41.14 cm, which is a 2.85% broader than the target value and exhibits a standard deviation of 2.871 cm. Another solution was the chromosome $x = [2\ 3\ 1\ 1]$ with a response of 39.9 cm, which is 0.21% narrower than the target value, but has a standard deviation of 7.37 cm. We can see the trade-off that exists between obtaining a good adjustment of the mean of the response to the target value and the decrease in variability. In that sense, the algorithm allows us to obtain a range of feasible solutions and the experimenter must choose the solution to implement based on the relative importance of adjusting the mean to the target value and reducing variability. Additionally, as already mentioned, the cost and time to implement each solution in the real system will be another important consideration when making that decision.

3.2 Results obtained for the univariate complex systems

For further testing the algorithm, we built a simulator which emulates a multivariate system with four responses. Before testing the algorithm with the multivariate system, we used the four responses of the simulator separately and independently of each other. That situation corresponds to applying the algorithm to four different univariate systems. Since each response is affected by 15 factors, the univariate systems are quite demanding. Consider that in the specialized literature, the univariate systems studied have between 3 and 8 control factors and about 3 noise factors, for a total of between 6 and 11 factors (Kackar and Shoemaker 1986; Allende et al. 2005). The 15 factors are divided into 10 control factors and 5 noise factors. The robust design considers using an inner array $L_{64}(4^{10})$ for the control factors and an outer array $L_{16}(4^5)$ for the noise factors. We must also note that each response has a minimum noise, which cannot be removed, normally distributed with a certain mean and standard deviation. Appendix A presents the details of the simulator and of the experimental

Table 2 Solutions found by the algorithm for the univariate complex systems

Response	Fitness function (FF) and estimation of missing value alternative (EMV)	Chromosome (solution)
y_1	1 and 2	1-4-4-4-4-4-4-4-4-1
y_1	3 and 4	1-3-3-3-3-1-1-1-1-3
y_2	1 and 2	1-2-2-2-2-1-1-1-1-2
y_2	3 and 4	1-2-2-2-2-2-2-2-2-1
y_3	All of them	3-3-1-2-4-4-2-1-3-2
y_4	1 and 2	3-2-4-3-1-3-1-2-4-4
y_4	3 and 4	4-4-1-3-2-2-3-1-4-3

Table 3 Comparison of means and standard deviation for the univariate complex systems

Response	FF and EMV alternatives	Response mean			Standard deviation		
		Optimum	Obtained	Diff.(%)	Minimum possible ($s_{\min}$)	Obtained (s_{obt})	$s_{\text{obt}}/s_{\min}$ (times)
y_1	1 and 2	201.5	200.3	−0.60	2.1	5.883	2.80
y_1	3 and 4	201.5	187.1	−7.15	2.1	4.176	1.99
y_2	1 and 2	50.8	50.1	−1.38	0.8	2.167	2.71
y_2	3 and 4	50.8	52.4	3.15	0.8	1.712	2.14
y_3	All of them	965.0	924.2	−4.23	36.0	108.958	3.03
y_4	1 and 2	512.0	482.8	−5.70	28.0	98.892	3.53
y_4	3 and 4	512.0	593.3	15.88	28.0	91.530	3.27

design we used, along with the tolerance limits and target values set in the algorithm for each of the four responses.

Table 2 shows the best solutions delivered by the algorithm for each of the four responses analyzed independently of each other. Table 3 presents the difference between the optimum value and the one achieved by the solutions delivered by the algorithm, both for the mean as well as for the standard deviation.

Before analyzing the results shown in Table 2, we must explain that the optimum value for each response differs from the corresponding target values (see Appendix A). For example, for response y_1, the target value is 200, while the optimum feasible value is 201.5. This happens because the systems have a minimum noise with a given mean, which cannot be removed, which in the case of response y_1 is 1.5. Thus, it would be unfair to judge the performance of the algorithm against the target value (200), since by construction and setting of the simulator, it can never be reached. That is the reason why we use the optimum value instead of the target value in evaluating the algorithm. The same happens with the standard deviation. Since each response has a noise with a minimum standard deviation, the solutions delivered by the algorithm will never be able to reduce the variability of the response to zero.

Analyzing the difference between the optimum value and the obtained value of the mean of each response in Table 3, we can see that for alternatives 1 and 2 (fitness function based on MSD), the solution found is practically equal to the optimum response of the systems. On the other hand, for alternatives 3 and 4 (fitness function based on SNR), those differences are bigger, but still sufficiently small to be adequate. Table 4 shows the difference between the optimum value and the obtained one expressed in terms of the minimum standard deviation of each response. As we can see, for all the responses, that difference is within three standard deviations. Thus, we can conclude that for univariate systems, the algorithm is effective in finding solutions that are close to the target value.

Regarding variability reduction, Table 3 shows that the standard deviations achieved using the fitness function based on the SNR (alternatives 3 and 4) are smaller than the ones achieved using the fitness function based on the MSD (alternatives 1 and 2). Thus, we may preliminarily say that the method based on the SNR might be more adequate in reducing variability than the one based on the MSD. Additionally, note that in general the obtained standard deviations are about three times the minimum standard deviation possible to get, which we regard as acceptable. Therefore, we think that for univariate systems, the algorithm seems adequate for reducing variability.

Table 4 Differences between the optimum value and the obtained response expressed in terms of standard deviations for the univariate complex systems

Response	FF and EMV alternatives	Response mean				
		Optimum	Obtained	Difference (Δ)	Minimum SD possible ($s_{\min}$)	$\Delta/s_{\min}$
y_1	1 and 2	201.5	200.3	−1.20	2.1	−0.57
y_1	3 and 4	201.5	187.1	−14.40	2.1	−6.86
y_2	1 and 2	50.8	50.1	−0.70	0.8	−0.87
y_2	3 and 4	50.8	52.4	1.60	0.8	2.00
y_3	All of them	965.0	924.2	−40.80	36.0	−1.13
y_4	1 and2	512.0	482.8	−29.20	28.0	−1.04
y_4	3 and 4	512.0	593.3	81.30	28.0	2.90

3.3 Results obtained for the multivariate complex system

In this case, we used the same simulator and the same experimental design as before, but we optimized the four responses at the same time. This means that we are optimizing a 4-dimensional multivariate system affected by 15 factors (10 control factors and 5 noise factors). We would like to note that in the specialized literature, we found studies involving multivariate systems with 2 and 3 responses and from 6 to 11 control factors, but with no noise factors (Roy 2001). This implies that performing robust design in multivariate high-dimensional systems is rather difficult. After executing and analyzing the solutions delivered by the algorithm, we saw that the feasible solutions are the same regardless of whether the algorithm used the fitness function based on the MSD or SNR. Table 5 presents those solutions.

Note that although we present all the solutions, some of them yield a very high standard deviation for the responses. On the other hand, the adjustment of the means of each response to the optimum value seems in all cases adequate. Thus, we present the solutions which

Table 5 Comparison of means and standard deviations of responses for the multivariate system

Resp.	Chromosome (solution)	Response mean			Standard deviation		
		Optimum	Obtained	Diff. (%)	Minimum possible ($s_{\min}$)	Obtained (s_{obt})	$s_{\text{obt}}/s_{\min}$ (times)
y_1	3-4-2-1-3-3-1-2-4-2	201.5	195.0	−3.23	2.10	6.33	3.01
	1-4-4-4-4-2-2-2-2-3		208.8	3.62		10.83	5.16
	2-2-1-4-3-4-3-2-1-3		207.4	2.93		5.59	2.66
y_2	3-4-2-1-3-3-1-2-4-2	50.8	51.7	1.77	0.80	2.52	3.15
	1-4-4-4-4-2-2-2-2-3		49.9	−1.77		2.38	2.98
	2-2-1-4-3-4-3-2-1-3		47.6	−6.30		2.90	3.63
y_3	3-4-2-1-3-3-1-2-4-2	965.0	910.0	−5.70	36.0	356.82	9.91
	1-4-4-4-4-2-2-2-2-3		950.7	−1.48		316.31	8.79
	2-2-1-4-3-4-3-2-1-3		1058.1	9.65		408.22	11.34
y_4	3-4-2-1-3-3-1-2-4-2	512.0	508.2	−0.74	28.0	117.63	4.20
	1-4-4-4-4-2-2-2-2-3		523.9	2.32		233.39	8.34
	2-2-1-4-3-4-3-2-1-3		505.2	−1.33		206.75	7.38

Table 6 Differences between the optimum value and the obtained one of the responses expressed in terms of standard deviations for the multivariate system

Resp.	Chromosome	Response mean				
		Optimum	Obtained	Difference (Δ)	Minimum SD possible ($s_{\min}$)	$\Delta/s_{\min}$
y_1	3-4-2-1-3-3-1-2-4-2	201.5	195.0	−6.5	2.10	−3.10
	1-4-4-4-4-2-2-2-2-3		208.8	7.3		3.48
	2-2-1-4-3-4-3-2-1-3		207.4	5.9		2.81
y_2	3-4-2-1-3-3-1-2-4-2	50.8	51.7	0.9	0.80	1.13
	1-4-4-4-4-2-2-2-2-3		49.9	−0.9		−1.13
	2-2-1-4-3-4-3-2-1-3		47.6	−3.2		−4.00
y_3	3-4-2-1-3-3-1-2-4-2	965.0	910.0	−55	36.0	−1.53
	1-4-4-4-4-2-2-2-2-3		950.7	−14.3		−0.40
	2-2-1-4-3-4-3-2-1-3		1058.1	93.1		2.59
y_4	3-4-2-1-3-3-1-2-4-2	512.0	508.2	−3.8	28.0	−0.14
	1-4-4-4-4-2-2-2-2-3		523.9	11.9		0.42
	2-2-1-4-3-4-3-2-1-3		505.2	−6.8		−0.24

exhibit a high standard deviation to show that in multivariate systems, it seems easier to adjust the means than reduce variance.

Regarding the adjustment of the means of the responses to their corresponding target values, Table 5 shows that almost all solutions achieve a value that is within 3% of the objective and in no case they exceed 10% of it. Globally, the chromosome [1 4 4 4 4 2 2 2 2 3] allows to simultaneously adjust the mean of the four responses to a range within 4% of the optimum value of each response.

Additionally, Table 6 shows that all the differences between the optimum values and the obtained responses are within four standard deviations and most of them within three standard deviations. Thus, we think that the algorithm delivers acceptable to good solutions with regard to mean adjustment.

Regarding variability reduction, Table 5 shows that the standard deviations of the responses are within 2.7 and 11.3 times the minimum standard deviation possible to achieve. For the responses y_1 and y_2 , the variability is within 2.7 and 5.2 times the minimum standard deviation, but for the responses y_3 and y_4 that range increases to within 4.2 and 11.3 times. Although the variability for y_1 and y_2 may be acceptable, the ones for responses y_3 and y_4 seem rather high. Globally, chromosome [3 4 2 1 3 3 1 2 4 2] allows to get responses with a low variability, except for response y_3. In general, we can say that the variability exhibited by the responses is high compared to the minimum variability possible to achieve. However, note that the design of the simulator considers a very difficult situation with respect to variability reduction. The variability of each response increases to the square of the difference between the optimum value of each noise factor (which delivers the minimum possible standard deviation) and the value set by the algorithm for each noise factor (see Appendix A).

Now, if we try to simultaneously adjust means and reduce variability for the four responses, the previous analyses and the figures presented in Table 5 indicate that chromosome [1 4 4 4 4 2 2 2 2 3] allows to adjust the means within a 4% around the optimum values and to get a standard deviation that is less than 9 times the minimum possible one. On the other hand, chromosome [3 4 2 1 3 3 1 2 4 2] adjusts the means within a 6% of the optimum value and reduces variability especially for responses y_1, y_2 and y_4. Thus, it seems that this chromosome may be the best possible global solution, except for reducing the variability of y_3.

However, if we tolerate a higher variability of the responses y_1, y_2 and y_4, the chromosome [1 4 4 4 4 2 2 2 2 3] might be the best.

4 Conclusions

This paper presented a methodology based on genetic algorithms for applying robust design to multivariate systems. The algorithm allows to find good combinations of treatments of the control factors, so that all the responses of the system are adjusted to the corresponding target values and variability is reduced as much as possible. The main contribution of the paper is the development of a tool which permits applying robust design to multivariate systems with many responses and a big number of factors and levels per factor. This was done using a desirability function, which integrates in one function all of the responses of the system. The desirability function was then used to build a fitness function, which guides the algorithm in finding a set of feasible solutions, which globally optimize the system.

The algorithm, regardless of the fitness function used (based on MSD or SNR), delivered solutions which allowed the responses to be close to their respective target values, both in univariate and multivariate systems. However, the reduction of variability in multivariate systems is only modest. For some of the responses of the multivariate system, the variability was still rather high. Although it seems that the fitness function based on the SNR is somewhat better for reducing variability than the function based on the MSD, both functions do not achieve consistently a high reduction in variability. However, we should again note that the simulator emulates a difficult situation in achieving a small variability.

Since the algorithm achieved modest results in variability reduction in complex situations, we think that our future developments of the algorithm should focus on that aspect. One possibility of improvement may be to change the fitness function. Studies have shown that using the SNR or MSD in robust design may be less efficient than directly using a measure of variability, such as the logarithm of the sample variance ($\log(s^2)$) and the mean ($\overline{y}$) (Pignatiello 1988). The main change to accommodate those expressions would involve modifying the desirability function presented in Sect. 2, so that it would simultaneously integrate $\log(s^2)$ and the mean of each response. An additional advantage of that change would be to eliminate the need of classifying the control factors in adjustment and dispersion ones, which would facilitate the application of Taguchi's two-stage optimization process. When performing the minimization of variability and adjustment of the mean, the algorithm would implicitly separate the control factors in those two categories.

Another pending work is to explore the efficiency of the algorithm for different kinds of response surfaces, analyzing its behavior as those surfaces become more complex. As part of that work, we should analyze the impact of the crossover and mutation probabilities on the efficiency of the algorithm for finding solutions which may adjust the mean and reduce variability of the responses, since it has been proved that those probabilities affect the performance of genetic algorithms (Golberg 1989). The same may be done with the lower and upper tolerance limits of the desirability function. We could even experiment with arbitrarily set a and b parameters, instead of calculating them using expressions (15) through (18).

Finally, it would be interesting to see whether the experimental design we used ($L_{64}(4^{10}) \times L_{16}(4^5)$) for applying the algorithm, might influence the solutions. In general, the more fractioned an experimental design, the more information one looses from the system. Thus, we should see that the more fractioned the design, the poorer the solutions should be.

However, we do not know for sure, how markedly the solutions would worsen as the fractioning increases.

All of those modifications to the algorithm may be done using a traditional robust design experiment, in which the control factors are the parameters of the algorithm and the noise factors are the different response surfaces and experimental designs to be used. In that form, we could determine the values of the parameters of the algorithm we should set given a specific experimental design we would like to use.

Acknowledgements We would like to thank Rick L. Riolo, Center for the Study of Complex Systems, The University of Michigan, and Claudio Moraga, European Centre for Soft Computing and FB Informatik Universität, Dortmund, for their valuable comments regarding the original version of this paper.

Appendix A

The simulator was built considering 15 factors, of which 10 are control factors and 5 are noise factors. The simulator has 4 responses. The equations which define the simulator are the following:

$$y_{Tr} = y_r + N_r \tag{22}$$

$$\begin{aligned} y_r = K_r &+ \sum_{i=1}^{10} A_{ir}\,(X_{ir} - O_{ir}) + B_{ir}(X_{ir} - O_{ir})^2 \\ &+ \sum_{i=1}^{4} \sum_{j=1, i<j}^{4} C_{ijr}\,(X_{ir} - O_{ir})\left(X_{jr} - O_{jr}\right) \\ &+ \sum_{i=5}^{8} \sum_{j=5, i<j}^{8} C_{ijr}\,(X_{ir} - O_{ir})\left(X_{jr} - O_{jr}\right) \end{aligned} \tag{23}$$

$$N_r = \mu_r + \sqrt{-2\ln(U_1)}\cos(2\pi U_2)\,\sigma_r \sum_{i=1}^{5} \left(Q_{ir}(W_{ir} - Z_{ir})^2 + 1\right) \tag{24}$$

where

- y_{Tr} : total value of response $r = 1, 2, 3, 4$
- y_r : value of r response without noise
- N_r : value of noise normally distributed $\mathrm{N}(\mu_r, \sigma_r^2)$ of each response r
- K_r : constant of each response r
- A_{ir} : linear coefficient of control factor i of response r
- X_{ir} : value of control factor i of response r
- O_{ir} : optimum value of control factor i of response r
- B_{ir} : quadratic coefficient of control factor i of response r
- C_{ijr} : interaction coefficient of the double interaction between control factor i and control factor j of response r
- μ_r : mean of the noise of response r
- U_1 : random number uniformly distributed between 0 and 1
- U_2 : random number uniformly distributed between 0 and 1
- σ_r : standard deviation of the noise of response r

Table 7 Values of coefficients set in the simulator

	y_1	y_2	y_3	y_4		y_1	y_2	y_3	y_4
Constant and linear coefficients					Quadratic coefficients				
K_r	200	50	1000	500	B_{1r}	0.002	−0.0003	0.008	0.005
A_{1r}	0.3	−0.08	2	1.5	B_{2r}	−0.05	0.001	0.09	0.002
A_{2r}	1.5	0.5	−2	0.8	B_{3r}	0.001	0.0002	0.008	0.005
A_{3r}	−0.15	0.08	1	0.3	B_{4r}	0.0005	0.00001	−0.0009	0.0008
A_{4r}	0.2	0.01	−0.5	0.1	B_{5r}	0.0024	−0.0002	−0.009	0.009
A_{5r}	0.5	−0.2	3.5	2.9	B_{6r}	0.00018	0.00008	0.0005	0.0001
A_{6r}	0.1	0.1	1.9	−0.5	B_{7r}	−0.0017	0.0005	0.0006	0.005
A_{7r}	−0.5	0.1	4.5	1.8	B_{8r}	0.0008	0.000015	0.005	0.0015
A_{8r}	0.8	−0.3	2.8	0.9	B_{9r}	−0.0003	0.00006	0.0008	0.001
A_{9r}	−0.3	0.09	−2.8	2	B_{10r}	0.028	0.003	−0.07	0.05
A_{10r}	2.2	−0.8	5.5	1.5					
Double interactions coefficients									
C_{12r}	0.0005	0.0001	0.002	−0.002	C_{56r}	0.00008	0.00002	0.0002	0.0001
C_{13r}	−0.0001	0.00005	0.00006	0.0005	C_{57r}	0.00001	0.000001	0.00008	0.00006
C_{14r}	0.0002	−0.000005	0.005	0.0005	C_{58r}	0.0009	−0.0001	0.006	−0.004
C_{23r}	0.0009	0.0002	0.002	0.0006	C_{67r}	0.0005	0.0003	0.0003	0.00001
C_{24r}	−0.00005	0.00004	0.0006	0.0003	C_{68r}	−0.002	0.0002	0.004	0.004
C_{34r}	0.0001	−0.00005	−0.0005	0.00008	C_{78r}	0.00004	0.0005	0.0008	−0.00007
Optimum value of control factor									
O_{1r}	75	45	12	60	O_{6r}	221	190	128	185
O_{2r}	12	12	18	8	O_{7r}	16	40	60	51
O_{3r}	18	50	95	72	O_{8r}	25	25	50	15
O_{4r}	110	75	25	65	O_{9r}	71	45	28	62
O_{5r}	50	55	82	40	O_{10r}	12	9	5	8
Noise factor coefficient					Optimum value of noise factor				
Q_{1r}	0.005	0.001	0.009	0.0008	Z_{1r}	8	12	10	9
Q_{2r}	0.0006	0.0002	0.0009	0.006	Z_{2r}	45	15	65	58
Q_{3r}	0.04	0.08	0.005	0.02	Z_{3r}	4	5	8	6
Q_{4r}	0.001	0.0004	0.008	0.0005	Z_{4r}	60	80	28	55
Q_{5r}	0.0003	0.0001	0.00005	0.0009	Z_{5r}	112	145	55	80
Mean of the noise					Standard deviation of the noise				
μ_r	1.5	0.8	−35	12	σ_r	2.1	0.8	36	28

Q_{ir} : quadratic coefficient of noise factor i of response r
W_{ir} : value of noise factor i of response r
Z_{ir} : optimum value of noise factor i of response r

The characteristics of the simulator defined by those equations are the following:

- The following double interactions exist between the control factors: 12, 13, 14, 23, 24, 34, 56, 57, 58, 67, 68, 78.
- There are no higher order interactions between control factors.
- There are no interactions between control and noise factors.
- There are no interactions between noise factors.

Table 7 shows the values we set for the coefficients for obtaining the data we used in the present study.

We must note that the response surfaces we used contain regions where all of the responses can be simultaneously achieved. This is a necessary condition to be able to find feasible solutions in the multivariate system.

Table 8 Target values and tolerance limits set in the algorithm

	y_1	y_2	y_3	y_4
Target value T_r	200	50	1,000	500
Lower tolerance limit L_r	190	47	800	400
Upper tolerance limit H_r	210	55	1,200	650

Table 8 shows the target values (T_r) and the lower (L_r) and upper (H_r) tolerance limits of the desirability function used in the algorithm, according to expression (12). We set those values around the value of the constant of each response and they might be changed to analyze the impact of them on the behavior of the algorithm.

The experimental design we selected corresponds to a product array $L_{64}(4^{10}) \times L_{16}(4^5)$, thus we used four levels for the control and noise factors. If someone decides to perform a full factorial experiment using four levels, he/she must run approximately 10^9 experiments, which is impossible to do in any real productive system.

References

Allende, H., Canessa, E., Galbiati, J.: Diseño de Experimentos Industriales, Edit. Universidad Técnica Federico Santa María, Valparaíso, Chile (2005)

Bravo, D.B.: Desarrollo de una Herramienta basada en Algoritmos Genéticos para resolver problemas de Diseño Robusto en Sistemas Multivariados. Master thesis, Universidad Adolfo Ibáñez (2005)

Del Castillo, E., Montgomery, D.C., McCarville, D.R.: Modified desirability functions for multiple response optimization. J. Qual. Technol. **28**, 337–345 (1996)

Golberg, D.E.: Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)

Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms. Wiley, New Jersey (2004)

Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI (1974)

Kackar, R.N., Shoemaker, A.C.: Robust design: a cost effective method for improving manufacturing processes. AT&T Tech. J. **65**(2), 39–50 (1986)

Leon, R., Schoemaker, A.C., Kackar, R.N.: Performance measures independent of adjustment: an explanation and extension of Taguchi's signal to noise ratios (with discussions). Technometrics **29**(3), 253–285 (1987)

Maghsoodloo, S., Chang, C.: Quadratic loss functions and SNR for a bivariate response. J. Manuf. Syst. **20**(1), 1–12 (2001)

Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 2nd edn. Wiley, New York, NY (2002)

Ortiz, F., Simpson, J., Pigniatello, J. Jr., Heredia-Langner, A.: A genetic algorithm Approach to multiple—response optimization. J. Qual. Technol. **36**(4), 432–449 (2004)

Pignatiello, J. Jr..: An overview of the strategy and tactics of Taguchi. IIE Trans. **20**(3), 247–254 (1988)

Roy, R.K.: Design of Experiments Using the Taguchi Approach, 1st edn. J. Wiley & Sons, New York, NY (2001)

Taguchi, G.: Systems of Experimental Design, vol. 1 and 2, 4th edn. American Supplier Institute, Dearborn, MI (1991)

Vandenbrande, W.: Make love, not war: combining DOE and Taguchi, ASQ's 54th annual quality congress. In: Proceedings, pp. 450–456 (2000)

Yuan, Y.: Multiple Imputation for Missing Data: Concepts and New Development. SAS Institute Inc., Rockville, MD (2006)