Operations Research Perspectives xxx (xxxx) xxxx

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

A novel hybrid multi-objective metamodel-based evolutionary optimization algorithm

Enrique Gabriel Baquela*,[a], Ana Carolina Olivera[b]

[a] Universidad Tecnológica Nacional, Facultad Regional San Nicolás, San Nicolás de los Arroyos, Buenos Aires, Argentina
[b] Instituto para las Tecnologías de la Información y las Comunicaciones, UNCuyo - CONICET, Facultad de Ingeniería, Universidad Nacional de Cuyo, Mendoza, Argentina

ARTICLE INFO

Keywords:
Optimization via simulation
Metamodel
Multi-Objective optimization
Kriging
NSGA-II

ABSTRACT

Optimization via Simulation (OvS) is an useful optimization tool to find a solution to an optimization problem that is difficult to model analytically. *OvS* consists in evaluating potential solutions through simulation executions; however, its high computational cost is a factor that can make its implementation infeasible. This issue also occurs in multi-objective problems, which tend to be expensive to solve. In this work, we present a new hybrid multi-objective *OvS* algorithm, which uses Kriging-type metamodels to estimate the simulations results and a multi-objective evolutionary algorithm to manage the optimization process. Our proposal succeeds in reducing the computational cost significantly without affecting the quality of the results obtained. The evolutionary part of the hybrid algorithm is based on the popular *NSGA-II*. The hybrid method is compared to the canonical *NSGA-II* and other hybrid approaches, showing a good performance not only in the quality of the solutions but also as computational cost saving.

1. Introduction

Optimization via Simulation, also known as *Simulated Optimization, Simulation-Driven Optimization* or *SimHeuristic*, is a handy tool to optimize dynamic systems which are too complicated to model analytically or algorithmically [1,2]. It consists in solving an optimization problem in which its objective functions and/or constraints are evaluated using a simulator. The use of a simulator makes it possible to deal with non-linearity in a relatively simple way, but its disadvantage is that, in general, the computational cost of executing a simulation is usually much higher than the cost of evaluating a set of analytical functions. *OvS* has a vast field of application in *Operation Research* problems, where usually the type of simulation used is a *Discrete-Event Simulation*. Despite this, *OvS* is a general framework that can be used in any kind of optimization problem with any simulation paradigm [3–5].

In the multi-objective optimization scene, specifically in the *Pareto Frontiers* approximation one, algorithms often require a considerable amount of objective functions evaluations [6]. Worse, the number of points needed to represent a *Pareto Frontier* grows exponentially respect to the number of objectives [7]. In an *OvS* scheme, this means a considerable amount of simulations, which implies a high computational cost. In many occasions, this high cost discourages analysts from using a methodology like this, and the alternative of using an analytical and less accurate methodology is sometimes chosen. The most commonly used way of tackling this difficulty, without resign to use the *OvS* framework, is not to carry out all simulations required but try to approximate them by means of a model of its inputs and outputs (a metamodel) [8–10].

When a metamodel is used in an *OvS* scheme, on the basis of a set of inputs and outputs samples of the simulated process, an analytical model that adjusts inputs with outputs with the slightest error is calculated, allowing the estimation of outputs for new input values. Experimental data could also be used to improve the accuracy of the metamodel, but in general, this type of data is not available in an *OvS* problem. *Regression Models, Neural Networks* and *Response Surfaces*, among others, are used for this purpose. A prevalent type of metamodel in the *OvS* world, which was used in this work, is the one called *Kriging* [11–13].

From the wide variety of existing alternatives to implement a multi-objective *OvS* algorithm based on metamodels, the use of *Genetic Algorithms* represents a very good one due to the evidence of the good trade-off between quality in the *Pareto Frontier* approximation and computational cost that they can reach [14,15]. For instance, the *K-MOGA* algorithm uses a scheme of this kind [16].

In this paper, we present a hybrid multi-objective *OvS* algorithm based on the *Elitist Non-Dominated Sorted Genetic Algorithm (NSGA-II)* and the use of *Kriging* metamodels. Unlike other methods, in which the metamodel is used to evaluate the highest number of possible solutions,

* Corresponding author.
E-mail addresses: ebaquela@frsn.utn.edu.ar (E.G. Baquela), acolivera@conicet.gov.ar (A.C. Olivera).

https://doi.org/10.1016/j.orp.2019.100098
Received 9 April 2018; Received in revised form 26 November 2018; Accepted 5 January 2019

here it is used to accelerate the exploration of the solutions space in the firsts iterations, achieving a good trade-off between computational cost and quality of the solutions. Our proposal combines the two ways of using metamodels in a *OvS* framework (i.e., calculate the metamodel previously to the optimization process and calculate the metamodel inside the optimization process) with a last stage which does not require the metamodel, with the aims of obtaining both, quality in the results and speed in the calculation process.

The structure of this work is the following: the literature review is in Section 1.1. Section 2 shows the basis on which the developed algorithm is set: The *NSGA-II* algorithm and the metamodeling *Kriging* method. Section 3 explains *K-NSGA-II* algorithm, and in Section 4 the experiments carried out are presented. This work finishes with the conclusions in Section 5 and the list of all references used in this work.

1.1. Literature review

The multi-objective optimization is a convenient tool to solve problems from the real world, in which there exist several objectives in conflict [17]. From the multiple ways of facing this kind of problems, the search for the *Pareto Frontier* is the most complex of all but the one that shows the best degree of flexibility to the decision-maker [18]. *The Pareto Frontier* of a problem is the set of solutions that present values in its objectives in which the unique way of improving an objective is worsening another. From the decision-making process' point of view, all solutions of the *Frontier* are equivalent among them and are better than all the rest. There exist three main approaches for the *Frontier's* construction: assigning weights to the objectives iteratively [19,20], guiding the search with the aim of maximizing some quality indicator [21,22], or selecting non-dominated solutions [15,23–25]. The last approach is the one that has got more attention lately. Within this, the *NSGA-II* [14] is one of the most popular *Pareto Frontier* approximation algorithms.

The *OvS* is a general optimization methodology that had a significant development during the last two decades [1,2,5,11,26,27]. The use of simulations makes it possible to deal with problems that, analytically, would be very difficult to address. However, a drawback of it is that it requires several simulations and, in general, that is expensive, even more, if the problem to be solved is multi-objective [28]. For that reason, hybrid versions of *OvS* algorithms based on metamodels have been developed with the aim of avoiding the use of simulations whenever possible [11,29].

Sacks et al. [10] use *Kriging* metamodels for the first time to approximate the output of a simulation. A popular *OvS* algorithm based on *Kriging* metamodels is the *Efficient Global Optimization algorithm (EGO)*, which is an iterative algorithm that seeks the global optimum on the grounds of the mathematical properties of the metamodel. Knowles et al. [30,31] developed a multi-objective version of the *EGO* algorithm called *Pareto Efficient Global Optimization (ParEGO)*. It is based on the same principles as *EGO*, but its objective is to find the *Pareto Frontier* of a multi-objective problem. It is a weight-based optimization algorithm that linearly combines different objectives and tries to find the global optimum of such combination. *ParEGO* has received several refining in order to make it more efficient: Davins et al. [32] use a double-metamodeling scheme to accelerate the convergence, Aghamohammadi et al. [33] study the effects of variables scalarization in the *ParEGO* performance, and Hakanen et al. [34] use an interactive approach that takes into account the user's preferences. *ParEGO*, however, suffers from the basic flaw of the method based on weights: it is extremely difficult to determine which combinations of weights to choose, which tends to make the algorithm requires too many iterations.

Other developments in metamodeling have been focused on improving the quality of the metamodel itself. For example, Zhao et al. [35] proposed a *Dynamic Kriging* methodology in which different sets of estimation functions are used in different sets of points in order to manage the non-linearity of the model space. This method has shown very good results in complex optimization problems [36]. Volpi et al. [37] developed an algorithm based on *Dynamic Radial Functions* and compared it with a *Dynamic Kriging* algorithm, finding good results in high dimensional problems. Gu et al. [38] proposed an optimization algorithm which automatically selects appropriate metamodeling techniques during the searching process with the aim of improving searching efficiency. Yang et al. [39] used an adaptive version of *Kriging* model which upgrade the model with new points on each iteration, increasing in this way the quality of the estimations. Iuliano [40] explore several adaptatives strategies. And Diez et al. [41] designed a method where a *Dynamic Radial Basic Function* is combined with an adaptive sampling method with the aim of improving not only the current solution but also the metamodel.

A natural solution that makes it possible to use the vast development in the metaheuristics area consists in using the metamodels combined with multi-objective evolutionary algorithms [8]. Bittner et al. [42] suggest combining the use of *Kriging* metamodels with a multi-objective version of the well-known *Particle Swamp Optimization (PSO)*. Todoroki et al. [43] use metamodels to evaluate the objectives in a *Multi-Objective Genetic Algorithm (MOGA)*. Husain et al. [44] combine different metamodeling techniques with the well-known *NSGA-II* to solve a design problem in a heat sink, concluding that the *Kriging* metamodel shows the best performance when it is combined with this optimization algorithm. Choi et al. [45] create a metamodel and subsequently look for the metamodel's *Pareto Frontier* through *NSGA-II*, but they cannot correctly approximate the *Pareto Frontier* of the aeronautic design problem under study (because the model is not recalculated). Voutchkov et al. [46] combine *Kriging* with *NSGA-II*, calculating the real value (i.e., the one generated by the simulation) of the solutions selected as the best ones in each generation with a simulation, obtaining good results but high calculation time. Regis [47] developed an evolutionary algorithm which recalculates the metamodel in each iteration. Li et al. [16] develop the *K-MOGA* algorithm, an adaptive algorithm which uses a modified version of the *Non-Dominated Sorted Genetic Algorithm (NSGA)* combined with *Kriging* and uses the calculation of the metamodel variance to determine when it is necessary to recalculate it. *K-MOGA* gets a very good approximation in the problems under study, making in some cases around half of the simulation executions regarding the case of not using metamodels. Work carried out by Choi et al. [45], Li et al. [16], Yang et al. [39], Regis [47] and Diez et al. [41] conclude that, in the general case, to design a metamodel in advance does not ensure good results and it is convenient to refine the initial model as the optimization model is being executed.

One problem that the approaches which combine evolutionary algorithms and *Kriging* have to deal with is to decide when it is convenient to recalculate the model and how many samples must be used in such recalculation. If it is not frequently adjusted, the convergence to the *Pareto Frontier* is not so good, but if it is frequently modified, several additional simulations are required (and the computational cost rises). In general, the decision is made by using the statistical properties of it, detecting if the error in the calculation is significant (e.g., it is the method used by Li et al. [16] to recalculate the metamodel as the optimization process progresses). However, this becomes a tautology in which the model is used to make predictions about itself.

In this work, we develop a three-stage adaptive metamodel-based optimization algorithm called *K-NSGA-II* that combines *Kriging* with a modified *NSGA-II* in which the metamodel is improved in each generation. Additionally, the recalculation of such metamodel is determined by random sampling and the elitism operator is modified to avoid premature convergence. A deep analysis of our proposal is carried out in order to explore its potential. Seventeen literature benchmarks are used for comparison between Canonical NSGA-II, K-MOGA, K-NSGA-II (our first approximation) and K-NSGA-II-S3 based on K-NSGA-II with a tuning stage to improve the convergence.

The main contributions of our approach are the following:

- Development of a three-stage algorithm in which the metamodel is used to discard solutions far from the *Pareto Frontier*.
- Creation of a method to determine the need for metamodel recalculation not based on the model properties but through random sampling of simulations.
- Introduction of a *Negative Elitism* operator that decreases the probability of premature convergence.

2. NSGA-II and metamodels

In this section, the bases of the *K-NSGA-II* algorithm are presented: the canonical *NSGA-II* and the *Kriging-type* metamodels' construction for simulation outputs' estimation.

2.1. Canonical NSGA-II

The *NSGA-II* is an algorithm that belongs to the *Genetic Algorithms* family adapted to the multi-objective optimization [14]. It aims to achieve an approximation to the problem's *Pareto Frontier* as near as possible.

Unlike canonical genetic algorithm, in which each individual's fitness coincide with the value of the objective function, the criterion of non-dominance and the *Crowding Distance (CD)* are used here to determine when a solution is better than another. Each individual is qualified in relation to the position it has in the non-dominance hierarchy and, for those elements in the same level, in relation to the closeness to other solutions.

In each algorithm iteration, the values of all objectives of each individual are calculated and, based on these results, the non-dominance or Pareto ranking is built among all individuals. For this purpose, each individual is compared to the rest of the population so that it can be determined if it is dominated by any other individual. It if is not dominated by any, a ranking value 1 is assigned to it. After all individuals are compared, all non-dominated ones (the ones with ranking 1) are temporarily removed from the population, and then the process is repeated with the remaining individuals. Non-dominated individuals of this iteration are assigned value 2 in the ranking and are temporarily removed from the population. The process is repeated until every individual is assigned a value in the ranking. Each set of individuals carrying the same ranking value represents equivalent solutions that dominate all the ones with higher ranking and are dominated by all the ones with a lower ranking. With the aim of having an order criterion among solutions with the same ranking value, the crowding distance of each solution is calculated. For this purpose, solutions are ordered increasingly for each objective and, for each solution, the average distance to those solutions immediately before and immediately after is calculated.

A comparison operator $\prec_n$ called *Crowded-Comparison Operator* can be constructed with each individual ranked and with an assigned crowding distance in order to determine when a solution is better than another. Assuming that it is a minimization problem, $\prec_n$ is defined according to Eq. (1), where i_{rank} and j_{rank} are the Pareto Ranking values corresponding to the individuals i and j whereas $i_{crownDist}$ and $j_{crownDist}$ are the *Crowding Distances* of such individuals.

$$i \prec_n j \text{ if } (i_{rank} < j_{rank}) \text{ or } [(i_{rank} = j_{rank}) \text{ and } (i_{crownDist} > j_{crownDist})] \quad (1)$$

NSGA-II is similar to a *Genetic Algorithm* with tournament and elitism. In a generation m, the starting point is a population P_m with a n size and each individual is evaluated, the *Pareto Ranking* values, as well as the *Crowding Distance*, are calculated. Then, the Q_m descendants population is created by selecting individuals through a binary tournament, crossover, and mutation. A new population $R_m = P_m \cup Q_m$ is defined according to these two populations and the best n elements of R_m are selected using the operator $\prec_n$. These individuals form the P_{m+1} population. The process is repeated until the number of maximum generations or other defined termination criterion is reached.

2.2. Kriging-type metamodels

Kriging-type metamodels were developed by Krige and Matheron [12,13] in order to model problems belonging to the field of geostatistics and space data modeling and were implemented as a tool to model simulations by Sacks et al. [10]. In this type of metamodels, the function to be modeled (ideally, a simulation) is represented as a Gaussian stochastic process in the way it is presented in Eq. (2)

$$Y(x) = \sum_{j=1}^{k} \beta_j \cdot f_j(x) + Z(x) \quad (2)$$

where x is the input received by the simulator, Y is the output generated by that simulator, f is an adjustment function, β is the weight of f, k is the number of adjustment functions, and $Z(x)$ is a Gaussian process with zero mean whose covariance for two different elements (for instance, x and w) is defined by Eq. (3).

$$V(w, x) = \sigma^2 \cdot R(w, x) \quad (3)$$

where $V(w, x)$ is the covariance between w and x, σ^2 is the variance of the random process Z (estimated from the sample's variance), and $R(w, x)$ the correlation matrix of such process (square and with $rank = n_0$). There are several alternatives to choose matrix R, but a typical selection is the one shown in Eq. (4) [16].

$$R(w, x) = exp\left[-\sum_{n=1}^{N} \Theta\, |w_n - x_n|^2\right] \quad (4)$$

with Θ a vector of Θ_n parameters to calculate. The main idea after the modeling through Eq. (2) is that the original process can be interpreted as a random way around a regression of its inputs. The objective is to find the most appropriate β, f_j and $Z(x)$ based on a set of training data. Unlike the least squares method, the aim is not to minimize the calculation error in the training set but to make this error be zero for any point in the training set [9,11]. Assuming functions f as constant [10,16], the response y for any input x^* can be predicted by $\hat{y}$ through Eq. (5), where $f = f_c$ is a vector with all its components equal to 1 and r the correlations vector of element x^* with each element of the training set (Eq. (6)).

$$\hat{y} = \beta + r^t(x^*)R^{-1}(y - f_c\beta) \quad (5)$$

$$r = [R(x^*, x_1),\ ..., R(x^*, x_n)] \quad (6)$$

As R only depends on the observations set x and the unknown parameter Θ, β can be expressed according to them through Eq. (7), being the one expressed in Eq. (8) the best estimator of Θ_n.

$$\beta = (f_c^t R^{-1} f_c)^{-1} f_c^t R^{-1} y \quad (7)$$

$$\Theta_n = max \frac{-[n_0 \ln(\sigma^2) + \ln(R)]}{2} \quad (8)$$

To calculate Θ exactly (and R as a consequence) is usually complex; for this reason, numerical procedures and/or metaheuristics are used to carry out such calculation [11]. *Genetic Algorithms* and *Newton Methods* are usually chosen.

The main theoretical characteristic of the resulting metamodel is that it is an exact estimator for the x values used in training, whereas for those values internal to the *Hypervolume* defined by the most extreme x values, it works as an interpolator. This interpolation works quite well unless there exist very strong gradient change points among the training points. In practice, due to Θ is not calculated precisely, the quality of the resulting model strongly depends on the running time allowed for doing calculations, the characteristics of the original model and the method used to optimize the metamodel.

3. K-NSGA-II

In this section, our proposal is shown. Section 3.1 starts by showing the general structure of the *K-NSGA-II* algorithm, and Sections 3.2, 3.3 and 3.4 describe the three algorithm stages, respectively. Within the second stage description, Section 3.3.1 explains how metamodels are used in the *K-NSGA-II* algorithm, and Section 3.3.2 proposes an additional elitism procedure to avoid premature convergence.

3.1. K-NSGA-II Structure

The general structure of the *K-NSGA-II* algorithm is based on the canonical *NSGA-II*. The modifications of the canonical structure are done with the aim of adapting its dynamics to the evaluation by means of metamodels of the objective and constraints functions. The optimization scheme is divided into three stages:

- The first stage (Stage I) generates the initial metamodel in advance and carries out the first iteration of the genetic algorithm.
- The second stage (Stage II) consists in taking the *Genetic Algorithm* iterations forward, using the metamodel to evaluate the objective functions. This metamodel is updated at every generation, but the degree of this update depends on how good the estimations generated by it are. The result of this stage seems to be a reduction of the solutions space, with an irregular approximation of the *Pareto Frontier*.
- The last stage (*Stage III*) is a fine-tuning stage in which the objective functions are not evaluated through the metamodel but through simulations. Here the irregular approximation of *Stage II* is taken as the initial population for a canonical *NSGA-II* algorithm.

Fig. 1 summarises the sequence of steps to follow at each generation for each stage. The global structure is similar to a canonical *NSGA-II*, but in *Stage I* a set of metamodels is built and in Stage II, before the evaluation of all individuals in the current population, there is a test to decide if evaluate the population using metamodel or by running all simulations.

The reason for using a metamodel to calculate the value that every objective takes instead of using the simulator itself is that, although the construction of such metamodels is expensive regarding computational issues, the execution of a simulation has a similar or higher cost so, when avoiding the execution of several simulations, the saving is really significant (in terms of time and necessary memory) [11]. The optimal situation would be to construct a good metamodel with few training data (few simulations) and that its errors are not significant, which is something that does not usually happen. In general, the calculations are very good for points close to the training ones, but the error increases as we move away from such points.

As it will be seen in Section 3.3, *K-NSGA-II* uses the tendency to the convergence of *Genetic Algorithms*. In the first generations, a *Genetic Algorithm* is an algorithm mainly exploratory that deals with solutions which, on average, have a significant distance among them. In this situation, the prediction errors of the metamodel are not so critical, and the value of the operator $\prec_n$ can be appropriately estimated using metamodels. In the last generations, however, the average distance among the solutions is much smaller, and the errors in the prediction of $\prec_n$ are more significant, that is why it is preferred to carry out the evaluation exactly and not approximately (i.e., using simulation).

The scheme used in the *K-NSGA-II* merges the three approaches used for dealing with metamodels: estimate metamodel prior optimization process, estimate and update metamodel during the optimization process, and not to use metamodel in function evaluation.

3.2. K-NSGA-II: Stage I

The first stage consists of the creation of the initial population, the evaluation of solutions, the calculation of metamodels (one per each objective) and the creation of the second generation of solutions. Basically, this stage is a canonical *NSGA-II* with only 1 generation, plus the metamodel's construction.

The generation of the initial population is carried out at random. Being $p_s = |P|$ the selected size of the population, $2p_s$ solutions are created stochastically, and all of them are evaluated through simulation. With these results, the first metamodel is calculated (hereinafter, be it noticed that when we say metamodel, we are mentioning the whole set of metamodels, one associated to each objective and to each constraint).

Then, from the $2p_s$-size set, p_s solutions are chosen randomly, being these ones the solutions that form the first generation. The processes of selection, crossover, mutation, and elitism are applied and as a result, the second generation is obtained. So as to improve the accuracy of *K-NSGA-II* algorithm, the initial population could be created by a more specific method, like *Latin Hypercube*.

In this first stage, the selection through the operator $\prec_n$ is carried out exactly, considering that all results of simulations of all solutions to be evaluated are available. Additionally, a metamodel with a good start accuracy is reached when considering the double of solutions for its calculation.

3.3. K-NSGA-II: Stage II

This is the most complex stage and the one that generates more savings. It consists in moving forward from generation 2 until the beginning of *Stage III* evaluating the solutions through the metamodel. Again, it works as a canonical *NSGA-II* except for the selection procedure and the elitism process. In both, the changes made consist in that the evaluation is carried out through metamodels and that the elitism is divided in two, one positive and another negative (prone to include bad solutions).

3.3.1. Selection

The main idea of *K-NSGA-II* consists in evaluating the solutions using a metamodel instead of the original functions but ensuring that the selection of individuals through the operator $\prec_n$ is the same as using the original functions.

As an approximation model, *Kriging* has error margin in its calculation. Even as a sub-product of the metamodel calculation, an estimator for such error is obtained for each solution to be evaluated. Most optimization algorithms based on *Kriging* use this estimator to determine when the metamodel prediction is acceptable or not. This criterion is usually necessary in mono-objective optimization, but in multi-objective optimization, for the purpose of selecting individuals to create the population of the following generation, the accuracy in the calculation of the objective functions values is not so important. The important thing is the calculation of which solution of each pair of solutions is the non-dominated one. So long as the metamodel allows estimating the result of the operator $\prec_n$ correctly, the metamodel is valid to continue to be used (even if the error margin of each individual prediction is high); if the calculation of the application of $\prec_n$ is incorrect, the solutions must be evaluated through simulation, and the metamodel must be recalculated.

To predict how good the metamodel is, we use a boolean function $I(i, j)$ which gives back 1 if the result of $i \prec_n j$ is the same using the metamodel and calculating the value of the objectives for i and j using the true objective functions (i.e., simulating the solutions). Since $I(i, j)$ cannot be evaluated for all individuals (as that would require simulation of all solutions), a sample sized k_{check} is taken from the current population. Then, pairs of individuals (i, j) are formed and every pair is evaluated. As $I(i, j)$ takes the complementary values of $I(j, i)$, the number of checks to be carried out is of $N_{check} = k_{check} * (k_{check} - 1)/2$. The decision of using the metamodel to evaluate or not all solutions is made according to the proportion of results 1 of the function $I(i, j)$ regarding

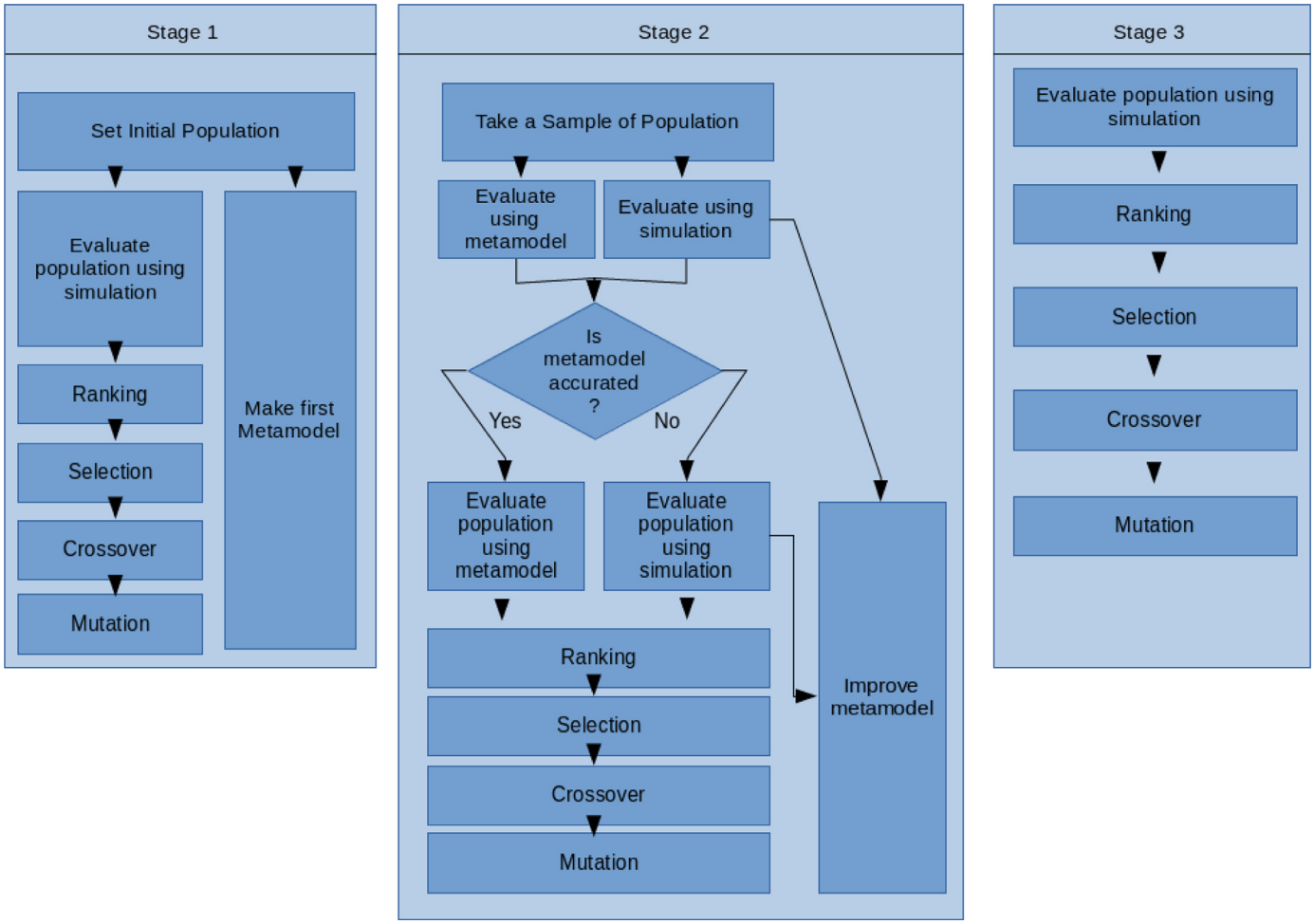


Fig. 1. K-NSGA-II stages scheme.

the total amount of analyzed couples (Eq. (9)).

$$\frac{\sum_i \sum_{j>i} I(i,j)}{N_{check}} \tag{9}$$

If the proportion is higher than a critical value α it is considered that the metamodel errors are not so significant and the rest of solutions are not simulated. However, if it is lower than α, all solutions of the population are simulated. In both cases, the metamodel is recalculated: in the first case, the solutions used for the test are added; in the second case, all the solutions of the population are added. For this reason, regardless of the result of the current generation, the following one will use a more accurate metamodel [39].

3.3.2. Negative elitism

The random method of evaluation of the metamodel quality (Section 3.3.1) has the problem of including a possible bias in the functioning of the ranking process and solutions selection based on the individuals selected in the sample. Since the process is random, for the checking process through the function $I(i, j)$ there is a risk of selecting only individuals for which the model works well and not evaluating any in which the model does not work well (or vice versa). This implies that the process of generation of the new population can discard good solutions and include bad ones. To ease that effect, the process of elitism is modified in order to include a greater diversity, at the expense of decreasing the convergence speed [48]. On the basis of the population R_m, the process to generate P_{m+1} is the following:

1. The population P_{m+1}^{good} is created by selecting the best $n - n_{ea}$ individuals from R_m through the application of the operator $i \prec_n j$.
2. The population P_{m+1}^{bad} is created by selecting the worst n_{ea} individuals from R_m through the application of the operator $i \prec_n j$.
3. It is made $P_{m+1} = P_{m+1}^{good} \cup P_{m+1}^{bad}$

This variation of the conventional elitism has the effect of reducing the probability of premature convergence of the algorithm.

3.4. K-NSGA-II: Stage III

As it was previously mentioned, *Stage II* has as its primary goal to reduce the size of the space of solutions to be explored. In many cases, the approximation of the *Pareto Frontier* at the end of such stage is good, but in other cases, it only takes the form of the Frontier with dominated solutions or else it cannot even take the form of it. This is due to the dynamics of the genetic algorithms: the average distance between solutions tends to decrease as the generations move forward and it is more difficult for the metamodel to achieve a correct prediction. Therefore, the final population of *Stage II* is taken as the initial population in *Stage III* and few generations of a canonical *NSGA-II* are executed (with selection without metamodels and traditional elitism). This stage works as a fine-tuning stage in which a narrowly defined region of the space of solutions is explored to finish shaping the approximation of the Pareto Frontier.

In the next section and in Section 5, when it is applied only *Stage I* and *Stage II* we refers to *K-NSGA-II* and the application of the three stages is that what we call *K-NSGA-II-S3*.

4. Numerical tests

In this section, all numerical tests to evaluate the performance of our proposal are described. *K-NSGA-II* and *K-NSGA-II-S3* are compared against canonical *NSGA-II* and *K-MOGA*. *K-MOGA*, as was indicated in Section 1.1, is another algorithm based in *Genetic Algorithm* and *Kriging* models, designed with the same purpose of our algorithm.

Section 4.1 describes the optimization problems used in the tests and how the benchmarks *Pareto Frontiers* for all of them are generated. Section 4.2 indicates the metrics used to evaluate how good the approximations to the *Pareto Frontiers* are. Section 4.3 shows the configuration of tested algorithms. In Section 4.4 is analyzed how many evaluations without metamodel are required by each algorithm for a fixed level of performance. Section 4.5 shows an analysis of the effect of *Negative Elitism* in the algorithm performance. Section 4.6 contains an analysis of k_{check} and α parameters. Finally, Section 4.7 shows some guidelines to set the new set of parameters.

4.1. Tested problems

To evaluate the proposed algorithm performance, a set of benchmark problems that are popular in the multi-objective optimization literature were selected to be optimized [14]. Tables 1 and 2 show, for each problem, the objective functions, the bounds of each variable and the constraint functions.

To generate the *Pareto Frontiers* of all benchmark problems, the following two procedures were performed:

(1) Obtain 1000000 points from the objective space of the target problem, sampling the decision space using random uniform probability distributions. In case the point was infeasible, this point is re-sampled.
(2) Run 100 different *NSGA-II* instances over the problem, using a random initial population for each one.

Procedure (1) allows obtaining a large sample of the objective space and Procedure (2) gives a set of 100 *Pareto Frontier* estimations. The final step to obtain the benchmark *Pareto Frontier* consists on merging all points obtained in both procedures and recalculating the *Pareto Frontier* of this large set of points.

4.2. Metrics for performance measurement

To evaluate the performance of the *K-NSGA-II* and *K-NSGA-II-S3*, we used two types of indicators: quality indicators and cost indicators. As quality indicators, we have chosen the following four [49–53]:

- *Generational Distance*
- *Generalized Spread*
- *Dominated Hypervolume*
- *Epsilon Indicator*

The *Generational Distance* indicator is calculated by averaging the minimum Euclidean distance of each point of the curve to measure with the *Pareto Frontier* used as benchmark curve. Its calculation is shown in Eq. (10):

$$GD = \frac{\sqrt{\sum_{i=1}^{N} d_i^2}}{N} \tag{10}$$

where N is the number of solutions to be compared with the *Pareto Frontier* and d_i is the minimum distance from solution i to the Pareto Frontier. The indicator is a measure of closeness to the benchmark frontier. The smaller it is, more uniformly distributed are the points of the estimated *Pareto Frontier* respect to the benchmark curve. That is, when the value of this indicator is big, the estimated *Pareto Frontier* has many points clustered, and it does not approximate the entire benchmark curve.

Generalized Spread is calculated by Eq. (11):

Table 1
Benchmark Problems - Part 1.

Problem Name	Function	Domain	Constraints
Belegundu	$f_1(x,y) = -2{\cdot}x + y$ $f_2(x,y) = 2{\cdot}x + y$	$0 \le x \le 5$ $0 \le y \le 3$	$0 \ge -x + y - 1$ $0 \ge x + y - 7$
Binh1	$f_1(x,y) = x^2 + y^2$ $f_2(x,y) = (x-5)^2 + (y-5)^2$	$-5 \le x, y \le 10$	-
Binh2	$f_1(x,y) = 4{\cdot}x^2 + 4{\cdot}y^2$ $f_2(x,y) = (x-5)^2 + (y-5)^2$	$-5 \le x, y \le 10$	$0 \ge (x-5)^2 + y^2 - 25$ $0 \ge -(x-8)^2 - (y+3)^2 + 7.7$
Binh3	$f_1(x,y) = x - 10^6$ $f_2(x,y) = y - 2{\cdot}10^6$ $f_3(x,y) = x{\cdot}y - 2$	$10^{-6} \le x, y \le 10^6$	-
Deb3	$f_1(x,y) = 4{\cdot}x$ $f_2(x,y) = g(y){\cdot}h(f_1(x,y), g(y))$ *with*: $g(a) = (4 - 3{\cdot}\exp(-((a-0.2)/0.02)^2))$ $h(b,c) = 1 - (\frac{b}{c})(0.25 + 3.75{\cdot}(b-1))$	$0 \le x, y \le 1$	-
Fonseca1	$f_1(\vec{x}) = 1 - \exp[-1{\cdot}(x_1-1)^2 - 1{\cdot}(x_2+1)^2]$ $f_2(\vec{x}) = 1 - \exp[-1{\cdot}(x_1+1)^2 - 1{\cdot}(x_2-1)^2]$	$-1 \le x_i \le 1,$ $i = 1 \ . \ n$	-
Fonseca2	$f_1(\vec{x}) = 1 - \exp[-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2]$ $f_2(\vec{x}) = 1 - \exp[-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2]$	$-4 \le x_i \le 4,$ $i = 1 \ . \ 2$	-
Hanne1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$	$-4 \le x_i \le 4,$ $i = 1 \ . \ 2$	$\sum_{i=1}^{i=2} x_i - 5 \le 0$
Hanne2	$f_1(\vec{x}) = (x_1)^2$ $f_2(\vec{x}) = (x_2)^2$	$-4 \le x_i \le 4,$ $i = 1 \ . \ 2$	$\sum_{i=1}^{i=2} x_i - 5 \le 0$
Hanne3	$f_1(\vec{x}) = \sqrt{x_1}$ $f_2(\vec{x}) = \sqrt{x_2}$	$-4 \le x_i \le 4,$ $i = 1 \ . \ 2$	$\sum_{i=1}^{i=2} x_i - 5 \le 0$

Table 2
Benchmark Problems - Part 2.

Problem Name	Function	Domain	Constraints
Hanne4	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$	$-4 \le x_i \le 4$, $i = 1 \quad . \quad 2$	$x_2 - 5 + 0.5 \cdot x_1 \cdot \sin(4 * x_1) \le 0$
Hanne5	$f_1(\vec{x}) = trunc(x_1) + 0.5 + c_2(\vec{x}) \cdot \sin(c_1(\vec{x}))$ $f_2(\vec{x}) = trunc(x_2) + 0.5 + c_2(\vec{x}) \cdot \sin(c_1(\vec{x}))$ *with* $c_1(\vec{x}) = 2 \cdot \pi \cdot (x_1 - trunc(x_1))$ $c_2(\vec{x}) = x_1 - trunc(x_1)$	$0 \le x_i \le 1$, $i = 1 \quad . \quad 30$	$x_2 - 5 + 0.5 \cdot x_1 \cdot \sin(4 * x_1) \le 0$
Jimenez	$f_1(\vec{x}) = 5 \cdot x_1 + 3 \cdot x_2$ $f_2(\vec{x}) = 2 \cdot x_1 + 8 \cdot x_2$	$10^{-6} \le x, y \le 10^6$, $i = 1 \quad . \quad 2$	$x_1 + 4 \cdot x_2 - 100 \le 0$ $3 \cdot x_1 + 2 \cdot x_2 - 150 \le 0$ $200 - 5 \cdot x_1 - 3 \cdot x_2 \le 0$ $75 - 2 \cdot x_1 - 8 \cdot x_2 \le 0$
VNT	$f_1(\vec{x}) = \frac{(x_1)^2 + (x_2)^2}{2} + \sin((x_1)^2 + (x_2)^2)$ $f_2(\vec{x}) = (3 \cdot x_1 - 2 \cdot x_2 + 4)^{\frac{1}{4}} + (x_1 - x_2 + 1)^{\frac{2}{27}} + 15$ $f_3(\vec{x}) = \frac{1}{(x_1)^2 + (x_2)^2 + 1} - 1.1 \exp(-(x_1)^2 - (x_2)^2)$	$-3 \le x_1 \le 3$ $-2 \le x_2 \le 2$	-
ZDT1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ *with* $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - \sqrt{\frac{u}{v}}$	$0 \le x_i \le 1$, $i = 1 \quad . \quad 30$	-
ZDT2	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ *with* $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - (\frac{u}{v})^2$	$0 \le x_i \le 1$, $i = 1 \quad . \quad 30$	-
ZDT3	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ *with* $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - \sqrt{\frac{u}{v}} - (\frac{u}{v}) \cdot \sin(10 \cdot \pi \cdot f_1(\vec{x}))$	$0 \le x_i \le 1$, $i = 1 \quad . \quad 30$	-

$$\Delta = \frac{\sum_{i=1}^{M} d(e_i, S) + \sum_{i=1}^{N} |d_i - \bar{d}|}{\sum_{i=1}^{M} d(e_i, S) + \bar{d} \cdot N} \tag{11}$$

where e_i are the M extreme points in the benchmark Pareto Frontier, $d(e_i, S)$ is the minimum Euclidean distance from the curve to measure S to the point e_i, d_i is the distance between the solution i and the solution $i + 1$ from the curve to measure, $\bar{d}$ is the average of N distances d_i and $N = |S|$ is the number of points in the curve to measure. This indicator measures how much spread the solutions found by the tested algorithm are. The smaller it is, the more uniformly distributed the obtained solutions are respect to the benchmark Pareto Frontier. The higher it is, the more clustered the obtained solutions are.

The *Dominated Hypervolume* is a quality measurement that evaluates, at the same time, the approximate *Pareto Frontier* distance against the benchmark *Pareto Frontier* as well as the space of the calculated frontier. It condenses the spread and convergence measurements in the same indicator. This metric calculates the volume of the objectives' space whose points are dominated by the curve under evaluation. The higher its value, the better the approximation is. The maximum level of this indicator is the *Dominated Hypervolume* by the real *Pareto Frontier* of the problem. To calculate it, a point W external to the curve is defined (the worst solution found, for example), and, per each point i in the approximate frontier Q, a v_i volume hypercube is constructed. The union of all hypercubes determines the total *Hypervolume* (Eq. (12)).

$$HD = volume(U_{i=1}^{|Q|} v_i) \tag{12}$$

The *Additive Epsilon Indicator* gives a factor by which a curve is worse than other considered all objectives. In our context, the factor can be considered as a scaling factor to be applied to transform the curve to measure into the Pareto Frontier. Its calculation is shown in Eq. (13), where R is the benchmark Pareto Frontier, A is the *Pareto Frontier* approximation obtained by the algorithm to test and d the number of objectives. As smaller is it, better is the approximation of A respect R.

$$EPS_{(+)} = \max_{r \in R} \min_{a \in A} \max_{i \in \{1..d\}} (a_i - r_i) \tag{13}$$

Additive Epsilon Indicator and *Hypervolume* are indicators that condense in one value the quality of the *Pareto Frontier* approximation respect to the benchmark Pareto Frontier. In most cases, evaluate an algorithm with one or another allows obtaining the same conclusion. However, in some cases, *Additive Epsilon Indicator* and *Hypervolume* can trigger different results, ie the first says that curve A approximates the benchmark curve in a better way than curve B, but the second says the opposite [51]. So, both indicators have been considered in this analysis. In regard of the selection of *Additive Epsilon Indicator* and Hypervolume, we added *Generalized Spread* and *Generational Distance* to our analysis with the aims to understand where the proposed algorithm performs better, in the minimization of the distance respect of the benchmark curve or in the coverage of all benchmark curve.

Respect to cost indicators, we have chosen the number of evaluations of the objective function made without the metamodel. If the objective function is evaluated without the metamodel, this number is equivalent to the number of simulations carried out in an OvS scheme. Depending on the nature of the objectives, all of them could be evaluated with the same simulation model, or could need different simulation models for each one. In the tested cases on this works, was assumed that all objectives are evaluated with the same model.

Table 3
Algorithms' parameters.

Parameter	Value
Population size	50
Tournament size	2
Crossover probability	0.8
Mutation probability	0.1
Elitism	0.1

4.3. Algorithms' configuration

To be able to evaluate the quality of the *K-NSGA-II* and *K-NSGA-II-S3*, its performance was compared with the metaheuristics *NSGA-II* [14], and *K-MOGA* [16]. *NSGA-II*, as well as *K-MOGA, K-NSGA-II* and *K-NSGA-II-S3*, were executed using stopping criteria consisting in finishing the optimization when the Dominated *Hypervolume* of the *Pareto Frontier* approximation is greater or equal to 95% of the benchmark *Pareto Frontier*. This criterion allows us to compare all algorithms in the same conditions. In Table 3, the general configuration of the algorithms is shown. In the case of *K-NSGA-II* and *K-NSGA-II-S3*, $k_{check} = 10$, $\alpha = 0.90$ and $n_{ea} = 0.10$. Finally, for the *K-NSGA-II-S3*, its number of generations was estimated in order to use the last 20% generations for *Stage III*. To realize the estimation, a set of executions of the algorithm were done, using different numbers of generations, and the number of generations to match the goal of the 20% required was interpolated. Each algorithm run was repeated 100 times, with different random seeds.

4.4. Algorithms comparison

In this section, the comparison between the algorithms is shown. The number of true evaluations needed (that is, not through a metamodel) is shown in Tables 4–11. In these tables, the first column is the tested function, the second is the indicator name (mean, percentile 0.05, and percentile 0.95), and the columns three to six correspond to the values reached by *NSGA-II, K-MOGA, K-NSGA-II*, and *K-NSGA-II-S3*, respectively. Values in **bold** indicate that an algorithm needs the least number of evaluations to reach the same quality obtained by *NSGA-II* for an indicator, and values in *italics* show the second least number of evaluations.

Tables 4 and 5 show the estimated number of function evaluations needed to reach the 95% of the *Dominated Hypervolume* of the respective *Pareto Frontier*. *K-NSGA-II-S3* performs well in problems like *Belegundu* and *ZDT1*, saving almost half of the required evaluations by the canonical *NSGA-II*. In *Belegundu*, canonical *NSGA-II* performs satisfactory, requiring a small number of evaluations to obtain the *Hypervolume* goal. However, *K-NSGA-II-S3* performs betters, using a smaller set of evaluations to reach the same goal. *ZDT1* requires a lot of evaluations in both algorithms, but *K-NSGA-II-S3* still performs better than canonical *NSGA-II*.

On the other side, *K-NSGA-II-S3* performs as *NSGA-II*, or slightly

Table 4
Number of evaluations to reach the 95% of Pareto Frontier's Dominated Hypervolume - Part 1.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 0.05	348	*204*	218	**192**
	Mean	397	245	*226*	**207**
	Percentil 0.95	421	293	*287*	**248**
Binh1	Percentil 0.05	753	625	**536**	*573*
	Mean	1520	*681*	799	**630**
	Percentil 0.95	2474	**763**	948	*775*
Binh2	Percentil 0.05	691	*554*	548	**502**
	Mean	759	*603*	626	**588**
	Percentil 0.95	854	*691*	703	**624**
Binh3	Percentil 0.05	1417	*1309*	1539	**1106**
	Mean	1752	*1486*	1805	**1308**
	Percentil 0.95	2003	*1960*	2158	**1897**
Deb3	Percentil 0.05	**368**	415	507	*395*
	Mean	**401**	507	703	*413*
	Percentil 0.95	*487*	670	878	**481**
Fonseca1	Percentil 0.05	806	**437**	1047	*478*
	Mean	1483	*574*	1378	**515**
	Percentil 0.95	2521	*807*	2042	**759**
Fonseca2	Percentil 0.05	742	*591*	1329	**382**
	Mean	1461	*854*	1787	**462**
	Percentil 0.95	1763	*1203*	2177	**703**

Table 5
Number of evaluations to reach the 95% of Pareto Frontier's Dominated Hypervolume - Part 2.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne1	Percentil 0.05	948	*608*	836	**535**
	Mean	1104	*745*	973	**623**
	Percentil 0.95	1572	*871*	1168	**715**
Hanne2	Percentil 0.05	*857*	948	1045	**801**
	Mean	1073	*902*	1231	**865**
	Percentil 0.95	1298	*1036*	1318	**960**
Hanne3	Percentil 0.05	1063	*976*	1202	**905**
	Mean	1136	**1085**	1382	**993**
	Percentil 0.95	1504	*1135*	1567	**1076**
Hanne4	Percentil 0.05	939	*746*	978	**675**
	Mean	1076	*940*	1024	**710**
	Percentil 0.95	1194	*1029*	1130	**885**
Hanne5	Percentil 0.05	852	**817**	934	*827*
	Mean	*968*	986	1056	**956**
	Percentil 0.95	1143	*1114*	1148	**1089**
Jimenez	Percentil 0.05	**564**	742	789	*675*
	Mean	*767*	825	934	**734**
	Percentil 0.95	1273	*1087*	1240	**948**
VNT	Percentil 0.05	**309**	390	643	*376*
	Mean	*543*	603	751	**538**
	Percentil 0.95	*875*	**859**	881	923
ZDT1	Percentil 0.05	1308	*1204*	1335	**1034**
	Mean	1793	*1407*	1593	**1152**
	Percentil 0.95	2367	*1838*	2071	**1354**
ZDT2	Percentil 0.05	862	*738*	1154	**704**
	Mean	1457	*963*	1265	**835**
	Percentil 0.95	2054	*1266*	1749	**1176**
ZDT3	Percentil 0.05	1375	*1070*	1290	**932**
	Mean	1865	*1291*	1493	**1049**
	Percentil 0.95	2459	*1522*	1757	**1168**

Table 6
Number of evaluations to reach NSGA-II Additive Epsilon Indicator value - Part 1.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne1	Percentil 0.05	348	*204*	218	**192**
	Mean	397	245	*226*	**207**
	Percentil 0.95	421	293	*287*	**248**
Binh1	Percentil 0.05	753	*580*	625	**546**
	Mean	1520	*681*	814	**651**
	Percentil 0.95	2474	**763**	975	*808*
Binh2	Percentil 0.05	691	*548*	554	**502**
	Mean	759	*603*	626	**588**
	Percentil 0.95	854	*691*	703	**624**
Binh3	Percentil 0.05	1417	*1309*	1554	**1136**
	Mean	1752	*1486*	1862	**1378**
	Percentil 0.95	2003	*1960*	2259	**1937**
Deb3	Percentil 0.05	**368**	415	507	*395*
	Mean	**401**	507	703	*413*
	Percentil 0.95	*487*	670	878	**481**
Fonseca1	Percentil 0.05	806	**437**	1047	*478*
	Mean	1483	*574*	1378	**515**
	Percentil 0.95	2521	*807*	2042	**759**
Fonseca2	Percentil 0.05	742	*591*	1329	**382**
	Mean	1461	*854*	1787	**462**
	Percentil 0.95	1763	*1203*	2177	**703**

Table 7
Number of evaluations to reach the same NSGA-II Additive Epsilon Indicator value - Part 2.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne1	Percentil 0.05	948	*608*	836	**535**
	Mean	1104	**623**	745	*642*
	Percentil 0.95	1572	*871*	1168	**715**
Hanne2	Percentil 0.05	857	*848*	1045	**801**
	Mean	1073	*902*	1231	**865**
	Percentil 0.95	1298	*1036*	1318	**960**
Hanne3	Percentil 0.05	1063	*976*	1202	**905**
	Mean	1136	*1085*	1382	**993**
	Percentil 0.95	1504	*1135*	1567	**1076**
Hanne4	Percentil 0.05	939	*746*	978	**675**
	Mean	1076	*940*	1024	**710**
	Percentil 0.95	1194	*1029*	1130	**885**
Hanne5	Percentil 0.05	*852*	**817**	934	877
	Mean	**968**	986	1056	**956**
	Percentil 0.95	1143	*1114*	1178	**1089**
Jimenez	Percentil 0.05	**564**	742	789	*675*
	Mean	*767*	825	934	**734**
	Percentil 0.95	1273	*1087*	1240	**1048**
VNT	Percentil 0.05	**309**	390	643	*376*
	Mean	*543*	603	751	**538**
	Percentil 0.95	*875*	**859**	1081	923
ZDT1	Percentil 0.05	1308	*1234*	1335	**1034**
	Mean	1793	*1452*	1593	**1152**
	Percentil 0.95	2367	*1838*	2071	**1354**
ZDT2	Percentil 0.05	862	*738*	1154	**704**
	Mean	1457	*963*	1265	**835**
	Percentil 0.95	2054	*1266*	1749	**1176**
ZDT3	Percentil 0.05	1375	*1070*	1290	**932**
	Mean	1865	*1291*	1493	**1049**
	Percentil 0.95	2459	*1522*	1757	**1168**

Table 8
Number of evaluations to reach the same NSGA-II Generational Distance Indicator value - Part 1.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 0.05	348	*204*	218	**192**
	Mean	397	267	*226*	**207**
	Percentil 0.95	421	306	*287*	**248**
Binh1	Percentil 0.05	753	625	**536**	*573*
	Mean	1520	*681*	799	**630**
	Percentil 0.95	2474	**763**	948	*825*
Binh2	Percentil 0.05	691	554	*548*	**502**
	Mean	759	*603*	626	**588**
	Percentil 0.95	854	*691*	703	**624**
Binh3	Percentil 0.05	1417	*1356*	1539	**1106**
	Mean	1752	*1503*	1805	**1308**
	Percentil 0.95	2003	*1984*	2158	**1897**
Deb3	Percentil 0.05	**368**	415	507	*395*
	Mean	**401**	507	703	*413*
	Percentil 0.95	*487*	670	878	**481**
Fonseca1	Percentil 0.05	806	**437**	1047	*482*
	Mean	1483	*534*	1378	**515**
	Percentil 0.95	2521	**726**	2042	*778*
Fonseca1	Percentil 0.05	742	*591*	1329	**382**
	Mean	1461	*854*	1787	**462**
	Percentil 0.95	1763	*1203*	2177	**703**

Table 9
Number of evaluations to reach the same NSGA-II Generational Distance Indicator value - Part 2.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne1	Percentil 0.05	948	*608*	836	**535**
	Mean	1104	**623**	973	*635*
	Percentil 0.95	1572	*871*	1168	**715**
Hanne2	Percentil 0.05	857	*848*	1045	**801**
	Mean	1073	*902*	1231	**865**
	Percentil 0.95	1298	*1036*	1318	**960**
Hanne3	Percentil 0.05	1063	*976*	1202	**905**
	Mean	1136	*1085*	1382	**993**
	Percentil 0.95	1504	*1135*	1567	**1076**
Hanne4	Percentil 0.05	939	*746*	978	**675**
	Mean	1076	*940*	1024	**710**
	Percentil 0.95	1194	*1029*	1130	**885**
Hanne5	Percentil 0.05	852	**817**	934	*827*
	Mean	*968*	984	1056	**956**
	Percentil 0.95	1143	*1114*	1148	**1089**
Jimenez	Percentil 0.05	742	**675**	789	*693*
	Mean	*767*	825	934	**734**
	Percentil 0.95	1273	*1087*	1240	**1048**
VNT	Percentil 0.05	**309**	390	643	*376*
	Mean	*543*	603	751	**538**
	Percentil 0.95	*875*	**859**	881	923
ZDT1	Percentil 0.05	1308	*1107*	1335	**1034**
	Mean	1793	1452	*1357*	**1152**
	Percentil 0.95	2367	*1612*	2071	**1354**
ZDT2	Percentil 0.05	862	*746*	1154	**704**
	Mean	1457	*855*	952	**835**
	Percentil 0.95	2054	*1208*	1749	**1176**
ZDT3	Percentil 0.05	1375	*1132*	1290	**932**
	Mean	1865	*1376*	1493	**1049**
	Percentil 0.95	2459	*1645*	1757	**1168**

Table 10
Number of evaluations to reach the same Generalized Spread Indicator value - Part 1.

Function	Measure	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 0.05	348	*204*	232	**197**
	Mean	397	*222*	238	**212**
	Percentil 0.95	421	**258**	300	*263*
Binh1	Percentil 0.05	753	617	*586*	**543**
	Mean	1520	**638**	809	*644*
	Percentil 0.95	2474	**782**	963	*789*
Binh2	Percentil 0.05	691	**513**	558	*517*
	Mean	759	**595**	637	*606*
	Percentil 0.95	854	**629**	710	*633*
Binh3	Percentil 0.05	1417	*1144*	1579	**1130**
	Mean	1752	**1336**	1831	*1347*
	Percentil 0.95	2003	**1920**	2185	*1932*
Deb3	Percentil 0.05	**368**	*400*	520	407
	Mean	**401**	*424*	717	471
	Percentil 0.95	**487**	*496*	884	519
Fonseca1	Percentil 0.05	806	*497*	1061	**484**
	Mean	1483	**529**	1387	*570*
	Percentil 0.95	2521	*812*	2052	**771**
Fonseca2	Percentil 0.05	742	**390**	1335	*403*
	Mean	1461	**471**	1802	*489*
	Percentil 0.95	1763	*754*	2185	**708**

worse, for problems like *Deb3, Hanne5, Jimenez*, and *VNT*. Metamodels could not estimate the new solutions generated in each generation. In those cases, the metamodels were recalculated in almost all generations, working the algorithm like a canonical *NSGA-II*. This behavior, naturally not good from the saving cost point of view, has a positive point: in the worst problems tested, where metamodels did not work, *K-NSGA-II-S3* obtains the *Hypervolume* goal without increasing cost significantly respect canonical *NSGA-II*. That is, at least in the problems tested in this work, *K-NSGA-II-S3* obtains always results equal or better than canonical *NSGA-II*.

K-NSGA-II-S3 seems to have a similar degree of randomness than canonical *NSGA-II*. Amplitudes between 5% and 95% percentile have similar value for both algorithms. It is important to notice that, in such problems where *K-NSGA-II-S3* is by far better, ranges $5\% - 95\%$ are not overlapped with the same ranges for canonical *NSGA-II*. That is, the worse performance of *K-NSGA-II-S3* in those problems is better than the better performance of canonical *NSGA-II*.

It can be seen how *K-NSGA-II* (without the *Stage III*) needs many calculations to reach the expected value. Cost savings are small or negatives (that is, the algorithm is more expensive than canonical *NSGA-*

Table 11
Number of evaluations to reach the same level as NSGA-II in Generalized Spread Indicator-Part 2.

Function	Measure	*NSGA-II*	*K-MOGA*	*K-NSGA-II*	K-NSGA-II-S3
Hanne1	Percentil 0.05	948	650	842	547
	Mean	1104	*732*	979	**638**
	Percentil 0.95	1572	*921*	1177	**725**
Hanne2	Percentil 0.05	857	*824*	1054	**813**
	Mean	1073	*878*	1237	**820**
	Percentil 0.95	1298	*1071*	1324	**967**
Hanne3	Percentil 0.05	1063	*945*	1219	**909**
	Mean	1136	*1061*	1410	**1030**
	Percentil 0.95	1504	*1183*	1604	**1097**
Hanne4	Percentil 0.05	939	*716*	1020	**693**
	Mean	1076	*798*	1060	**741**
	Percentil 0.95	1194	*967*	1148	**904**
Hanne5	Percentil 0.05	*852*	860	963	**838**
	Mean	*968*	990	1077	**927**
	Percentil 0.95	*1143*	1152	1184	**1131**
Jimenez	Percentil 0.05	**564**	*706*	812	742
	Mean	*767*	794	956	**762**
	Percentil 0.95	1273	*1093*	1280	**1070**
VNT	Percentil 0.05	**309**	*395*	668	408
	Mean	**543**	569	791	*567*
	Percentil 0.95	*875*	**789**	951	914
ZDT1	Percentil 0.05	1308	**1063**	1371	*1085*
	Mean	1793	*1285*	1620	**1140**
	Percentil 0.95	2367	*1481*	2097	**1390**
ZDT2	Percentil 0.05	862	**726**	1187	*735*
	Mean	1457	**857**	1293	*964*
	Percentil 0.95	2054	*1307*	1774	**1217**
ZDT3	Percentil 0.05	1375	*1060*	1311	**956**
	Mean	1865	*1276*	1513	**1069**
	Percentil 0.95	2459	*1500*	1783	**1189**

II). It can be supposed that metamodels are useful in the first generations, where genetic algorithms have an exploration behavior [7]. There is no need to use a lot of simulations to evaluate the solution, and also *Negative Elitism* seems to add a degree of variety in populations that benefit exploration. However, in the last stages, when exploitation is the usual behavior, metamodel approximations are not good, and *Negative Elitism* could add an unnecessary extra effort to converge to *Pareto Frontier*. It is not wrong to suppose that the net effect of *Stage II* is to generate a good starting population for the *Stage III* (in fact, a canonical *NSGA-II* running across a small number of generations). That effect can be interpreted as a type of search space reduction. Due to the fact that in a *Genetic Algorithm* each generation is generated through a stochastic transformation over the previous generation, solutions evaluated in first generations of *Stage III* should not have so much diversity as the solutions evaluated in first generations of *Stage II*.

Regarding *K-MOGA*, the algorithm performed well, needing less number of evaluations than canonical *NSGA-II* in the same problems where *K-NSGA-II-S3* performed well. However, it required a few more evaluations than *K-NSGA-II-S3* in most cases.

Tables 6 and 7 show the number of evaluations needed to reach the same *Additive Epsilon Indicator* value than a canonical *NSGA-II* running the same number of evaluations than in the *Hypervolume* test. The results reported in the tables demonstrate that *K-NSGA-II-S3* reduce the mean of evaluations significantly in fifteen functions of the seventeen to obtain the same level of quality in *Additive Epsilon Indicator* follow by *K-MOGA* in twelve of these functions. Moreover, in the two functions that not obtain the best performance was the second best. It is interesting to note that for *Fonseca1* and *Fonseca2* the saving of evaluations reaches more than 50% in comparison to *NSGA-II*.

Similar occurs in Tables 8 and 9 for *Generational Distance*. In fifteen functions *K-NSGA-II-S3* obtains the least number of evaluations follow by *K-MOGA* with the second best.

In Tables 10 and 11 for *Generalized Spread* it can observe differences. *K-NSGA-II-S3* obtains the best saving for nine of the seventeen functions and the second best for seven functions. It can be interpreted as the *K-NSGA-II-S3* has some bias in its calculation process which induces to cluster the solution generation and evaluation process.

In order to have a general indication of how good is each algorithm, the distribution of the global average of mean evaluations was estimated using *Bootstrap* [54]. It was performed through *Monte Carlo Case Re-sampling* with 5000 repetitions over the set of 17 test problem. The result of this process is an estimation of the probability of each possible value of the mean of an indicator, under the strong assumption that we are dealing with a bag of problem containing all tested problems or a fiction problem which partially averages all characteristics of the tested problem. It is a biased distribution that clearly does not allow to make predictions but allows to compare *NSGA-II, K-MOGA*, and *K-NSGA-II-S3* in a cleaner way than only watching an indicator per problem.

For Fig. 2a–d Light-grey histogram and KN boxplot are the *K-NSGA-II-S3* algorithm whereas Grey histogram and KM boxplot represent the *K-MOGA* algorithm. Dark-grey bars indicate common values for *K-MOGA* and *K-NSGA-II-S3*. The horizontal axis indicates the value of the quotient between then mean numbers of evaluation done with the target algorithm in a bag of problems generated re-sampling the bag of the tested problems, and the mean of evaluation done by the *NSGA-II* for the same bag. For example, a value of 0.7 indicates that the target algorithm does 70% of the evaluations done by *NSGA-II*, on average. Histograms show the frequency of each value, and each box plot indicates the distribution of the values, with the most improbable values plotted outside the boxplot itself. Value 1.0 in the horizontal axis is, by definition, the mean of *NSGA-II*.

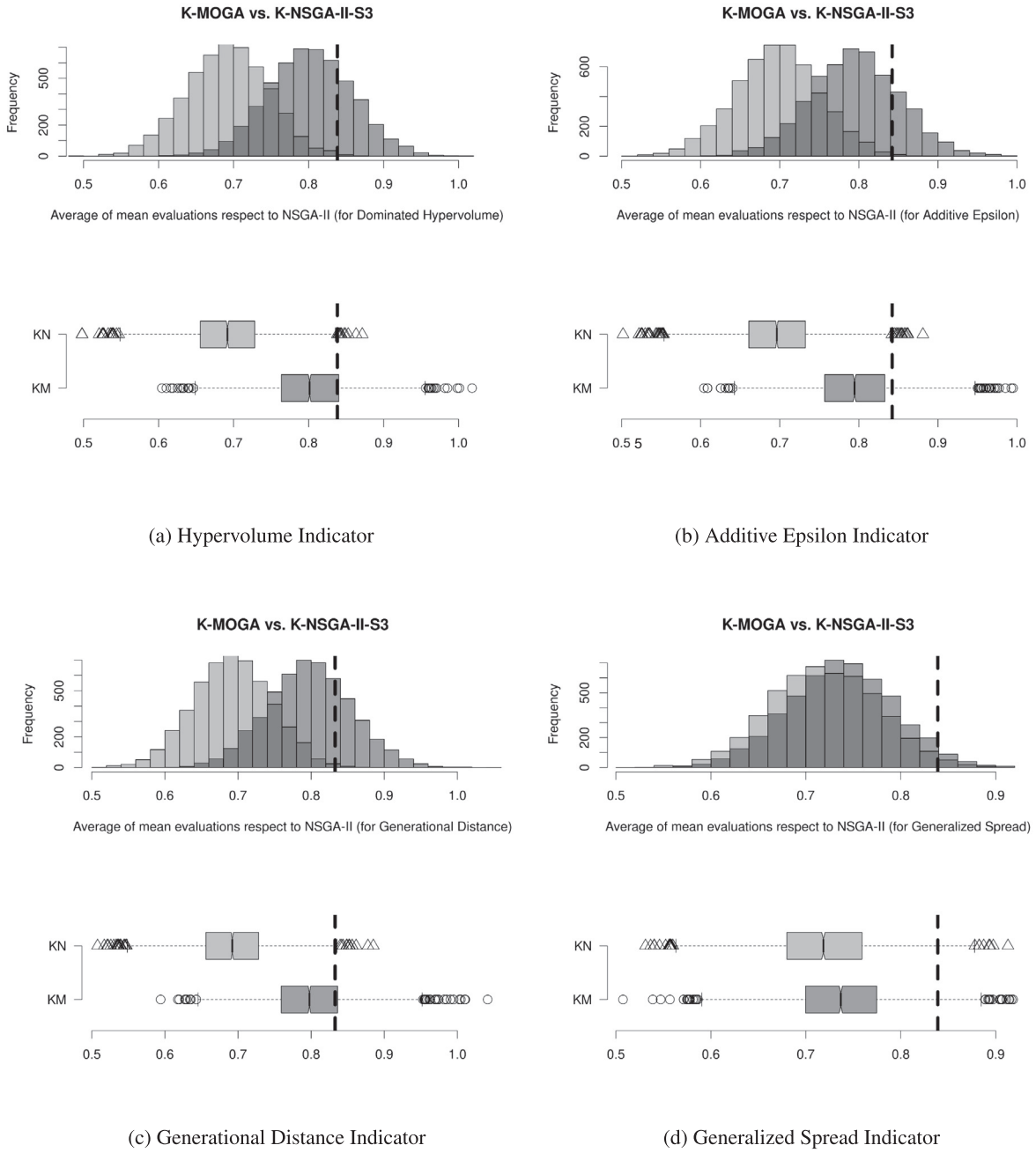
Fig. 2a shows the comparison between *K-MOGA* and *K-NSGA-II-S3* for the *Dominated Hypervolume* indicator. It can be seen that both algorithms have, in general, a mean behavior better than *NSGA-II*. *K-NSGA-II-S3* reaches better mean values than *K-MOGA*. Analyzing boxplots, it can be seen that, on average, the first 75% of all *K-NSGA-II-S3* observation are better than more of the 75% of all *K-MOGA* observations. The vertical dash-line indicates the percentile 0.05 of mean evaluations for *NSGA-II*. As can be seen, mean values for *K-NSGA-II-S3* are significantly betters than mean values of *NSGA-II* (in fact, only in *Deb3* was registered the opposite situation). Graphical analysis for this indicator shows similar results for the other indicators. Fig. 2b and c show that both, *Additive Epsilon* and *Generational Distance*, have a similar behavior than *Dominated Hypervolume*. Both have a better performance than *K-MOGA* and *NSGA-II*. But, in the case of *Generalized Spread* (Fig. 2d), the differences between *K-MOGA* and *K-NSGA-II-S3* disappear. It could be evidencing some kind of bias in *K-NSGA-II-S3*.

In general, the four indicators used show a good performance for *K-NSGA-II-S3*, but not for *K-NSGA-II*. *Generalized Spread* seems to be decoupled respect to the other three indicators in its behavior. Also, compared with *NSGA-II*, performance is better when problems need more iterations to be solved. To test this, the correlation between the following two indicators was calculated:

- $niter_{nsga-ii} = number\ of\ iterations\ needed\ by\ NSGA{-}II$
- $niter_{k-nsga-ii-s3} = \dfrac{number\ of\ iterations\ needed\ by\ K{-}NSGA{-}II{-}S3}{niter_{nsga-ii}}$

Correlation reaches a value of -0.55 for the case of *Dominated Hypervolume*, -0.52 for *Additive Epsilon*, -0.54 for *Generational Distance*, and for *Generalized Spread* a value of -0.52. These results imply that, although there are more factors that determine how many evaluations needs *K-NSGA-II-S3*, there exists some kind of inverse relation with the effort needed by the canonical *NSGA-II*. In other words, as more evaluations needs *NSGA-II*, more important are the percentual savings generated by *K-NSGA-II-S3*.

It is interesting to note that metamodels spare a significant number of evaluations for *ZDT1, ZDT2*, and *ZDT3* functions. Although they have a domain of 30 dimensions, the approximation via *Kriging* allows modeling the behavior of the objective functions adequately.



(a) Hypervolume Indicator

(b) Additive Epsilon Indicator

(c) Generational Distance Indicator

(d) Generalized Spread Indicator

Fig. 2. Comparison of the mean number of evaluation needed to reach the same indicator value.

4.5. Negative elitism analysis

In order to measure the effect of the negative elitism, *K-NSGA-II-S3* executes 50 generations for each problem, with and without the usage of negative elitism, in a new set of numerical tests. Table 12 shows the mean percentage difference in the *Dominated Hypervolume* indicator for the case without *Negative Elitism* respect the other case. *Negative Elitism* was set to values 0.05, 0.10 and 0.20. Its effects appear to be variable. Firstly, the impact of *Negative Elitism* seems to be strongly dependent on the individual characteristics of the *Pareto Frontier* to estimate. For example, if n_ea is set to 0.10, in *Fonseca1* the usage of *Negative Elitism* reduced the quality of the results, but in *ZDT1* it improved the results. It is interesting to points that, in the set of evaluated functions, mean improvements caused by *Negative Elitism* are greater than mean drawbacks caused by it, when $n_{ea} = 0.10$. Secondly, the usage of *Negative Elitism* in those cases when it has a positive impact should be constrained to small values of n_{ea}. In all cases, when $n_{ea} = 0.20$, the results were worse than $n_{ea} = 0.10$.

4.6. Parameters analysis

K-NSGA-II-S3 introduces several new parameters that need to be defined before running it. To determine how they impact on the optimization process, all the experiments of *K-NSGA-II-S3* were repeated with different sets of parameters. Table 13 shows the configuration tested and results are shown in Tables 14 and 15.

Due to k_{check} individuals are always evaluated in each generation, change the value of this parameter has a direct effect on the computational cost. Decreasing the value of k_{check} increases slightly the number of evaluations needed because of the loss in the model

Table 12
Mean Percentage Variation of *Hypervolume* when *Negative Elitism* is not used.

Problem Name	$n_{ea} = 0.05$	$n_{ea} = 0.10$	$n_{ea} = 0.20$
Belegundu	− 2%	− 2%	5%
Binh1	− 3%	− 4%	1%
Binh2	1%	0%	7%
Binh3	1%	2%	12%
Deb3	− 3%	− 3%	1%
Fonseca1	2%	7%	22%
Fonseca2	5%	12%	35%
Hanne1	− 2%	1%	6%
Hanne2	1%	3%	11%
Hanne3	0%	− 1%	3%
Hanne4	− 2%	− 2%	6%
Hanne5	1%	1%	5%
Jimenez	1%	3%	14%
VNT	− 6%	− 8%	− 3%
ZDT1	− 21%	− 23%	− 17%
ZDT2	− 5%	− 9%	− 1%
ZDT3	− 11%	− 15%	− 9%

Table 13
Algorithms' parameters.

Parameter	Value
k_{check}	{5; 10; 20}
α	{0.80; 0.90; 0.95}

Table 14
Number of evaluations needed to obtain a 95% of Dominated *Hypervolume* for different values of k_{check}.

Problem Name	$k_{check} = 5$	$k_{check} = 10$	$k_{check} = 20$
Belegundu	237	207	312
Binh1	727	651	729
Binh3	1428	1378	1484
Deb3	512	413	503
Fonseca1	606	515	616
Fonseca2	534	462	529
Hanne1	645	623	729
Hanne2	952	847	923
Hanne3	1118	1012	1093
Hanne4	742	724	776
Hanne5	1050	968	1183
Jimenez	795	744	898
VNT	704	562	722
ZDT1	1400	1302	1521
ZDT2	960	846	1025
ZDT3	1276	1121	1274

accuracy. Increasing the value of this parameters also shows an increase in computational cost.

Increasing α values implicates more needs of metamodel recalculation, which directly impacts on the computational cost. But reduce its value also increase the computational cost. In that case, because more generations are needed to reach the same quality.

Despite the difficulty of estimate these parameters, a procedure like *iRace*[55] can be used to define the best parameter selection.

4.7. Parameters proposed values

One drawback that many metaheuristics have, like Genetic Algorithms, is that the metaheuristic user needs to set the value of all parameters involved in the algorithm calculations. *K-NSGA-II-S3* add a new set of parameters: k_{check}, α, n_{ea}, the length of *Stage II* and the length of *Stage III*. Despite the fact that each individual problem has its own set of optimal parameter values, we ran an *iRace* [55] procedure over all tested problem in order to determinate what set of parameters has the best average behavior across all test set, setting the number of generations to 50. *iRace* got this set of values.

Table 15
Number of evaluations needed to obtain a 95% of Dominated *Hypervolume* for different values of α.

Problem Name	$\alpha = 0.80$	$\alpha = 0.90$	$\alpha = 0.95$
Belegundu	263	207	356
Binh1	690	651	1034
Binh2	643	588	723
Binh3	1429	1378	1682
Deb3	491	413	492
Fonseca1	482	515	761
Fonseca2	430	462	608
Hanne1	640	623	735
Hanne2	859	847	1009
Hanne3	1062	1012	1176
Hanne4	703	724	861
Hanne5	1005	968	1247
Jimenez	841	744	1089
VNT	635	562	958
ZDT1	1379	1302	158
ZDT2	889	846	1249
ZDT3	1232	1121	1426

- k_{check}: 10
- α: 0.90
- n_{ea}: 0.10
- Length of Stage III: $TotalGenerations \cdot 0.20$
- Length of Stage II: $TotalGenerations - LengthofStageIII - 1$

Said that, we suggest to run, if it is possible, the iRace procedure over a small instance of the problem to solve, in order to tune the parameters values to the problem structure.

5. Conclusions

With the aim to accelerate the convergence to the *Pareto Frontier* in Multi-Objective Problems solved by means of *Optimization via Simulation*, a novel algorithm called *K-NSGA-II-S3* was proposed in this work. *K-NSGA-II-S3* achieves satisfactory the trade-off between Quality Indicators and Number of Evaluations required in the problems analyzed. In the problems tested, the Quality Indicators reaches a similar level of what was obtained by the canonical *NSGA-II*, carrying out 50% of simulations or less in many cases. Also, in all instances where metamodels could not estimate well the objective functions, the *K-NSGA-II-S3* had a behavior similar to canonical *NSGA-II*, that is, it needed a similar amount of evaluations to reach similar results.

Compared to an algorithm that use of genetic algorithms combined with *Kriging* (*K-MOGA*) our proposal overcomes or equalize it for all instances.

A *Negative Elitism* procedure was introduced in order to prevent premature convergence in Stage II of the algorithm. The result of this procedure seems to have a strong dependence respect to the structure of the evaluated problem. Due to the absolute value of its effect is greater in the case where the results are improved than in the opposite case, it looks like a promissory path of research.

The sample-based process to evaluate if the fit of the metamodel respect to the real function to evaluate is good or bad had satisfactory results. With the proposed values of its parameters, this procedure could decide with a high accuracy when recalculating the model or not. Evidence of this the upper limit attain for the number of evaluations needed, similar to the number of evaluations done by the canonical *NSGA-II*.

As future work, is pending to analyze the dynamic behind the Negative Elitism. Due to the incidence of the problem structure on the behavior of Negative Elitism, how this procedure affects the search in

the solution space should be analyzed.

Performance respect to *Generalized Spread* have to be analyzed, because *K-NSGA-II-S3* seems to have some type of bias.

This work did not have into account the effect of noise on the performance of Stage II. This should be analyzed in future work, also the benefits of using *Kriging* instead of other methods to build meta-models.

The mechanism of how Stage II generate a good start point for Stage III should be studied more, in order to validate that hypothesis. Also, the length of this stages is currently defined a priori. A dynamic way to define when finish *Stage II* and start *Stage III* could improve the quality of the algorithm.

Finally, a multidimensional analysis of all parameters involved in *K-NSGA-II-S3* should be done.

Acknowledgements

The work of E. G. Baquela was supported by the Universidad Tecnolgica Nacional, under PID TVUTNSN0003605. A. C. Olivera thanks to Facultad de Ingeniería and Instituto para las Tecnologías de la Información y las Comunicaciones both from Universidad Nacional de Cuyo (UNCuyo). In the same way Olivera thanks to the Consejo Nacional de Investigaciones Cientficas y Técnicas (CONICET, Argentine).

References

[1] Amaran S, Sahinidis NV, Sharda B, Bury SJ. Simulation optimization: a review of algorithms and applications. 4OR 2014;12(4):301–33.
[2] Fu M. Optimization via simulation: a review. Ann Oper Res 1994;53:199–247.
[3] Koziel S, Ogurtsov S. Antenna design by simulation-Driven optimization. Springer International Publishing; 2014.
[4] Simulation-Driven modeling and optimization. In: Koziel S, Leifsson L, Yang X-S, editors. Springer Proceedings in Mathematics and Statistics, 153. Springer Int; 2014.
[5] Handbook of simulation optimization. In: Fu MC, editor. International Series in Operations Research & Management Science, 216. Springer Science + Business Media New York; 2015.
[6] Zhang W, Lu J, Zhang H, Qian Z, Gen M. Proceedings of the ninth International conference on management science and engineering management. Springer-Verlag Berlin Heidelberg; 2015. p. 963–76.
[7] Deb K, Saxena DK. On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. Tech. Rep.. Kanpur Genetic Algorithms Laboratory (KanGAL); 2005.
[8] Díaz-Manríquez A, Toscano G, Barron-Zambrano JH, Tello-Leal E. A review of surrogate assisted multiobjective evolutionary algorithms. Comput Intell Neurosci 2016.
[9] Kleijnen JP. Kriging metamodeling in simulation: a review. Eur J Oper Res 2009;192(3):707–16.
[10] Sacks J, Welch W, Mitchell T, Winn H. Design and analysis of computer experiments. Statist Sci 1989;4(4).
[11] Kleijnen JP. Design and analysis of simulation experiments. second edition International Series in Operations Research & Management Science 230. Springer International Publishing; 2015.
[12] Krige D. A statistical approach to some basic mine valuation problems on the wit-watersrand. J Chem, Metallurg Mining Soc South Africa 1951;52(6):119–39. https://doi.org/10.2307/3006914.
[13] Matheron G. Principles of geostatistics. Econ Geol 1963;58(8).
[14] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: nsga-ii. IEEE Trans Evol Comput 2002;6(2):182–97. https://doi.org/10.1109/4235.996017.
[15] Nedjah N, de Macedo Mourelle L. Evolutionary multi objective optimisation: a survey. Int J Bio-Inspire Comput 2015;7(1):1–25.
[16] Li M, Li G, Azarm S. A kriging metamodel assisted multi-objective genetic algorithm for design optimization. J Mech Des 2008. https://doi.org/10.1115/1.2829879.
[17] Massimiliano C, Paolo D. Multi-objective management in freight logistics. Increasing capacity, service level and safety with optimization algorithms. London: Springer London; 2008. p. 11–36.
[18] Deb K. Multi-objective optimization. Boston, MA: Springer US978-1-4614-6940-7; 2014. p. 403–49.
[19] Kim IY, de Weck O. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. Struct Multidiscip Optim 2006;31(2):105–16.
[20] h. Ryu J, Kim S, Wan H. Pareto front approximation with adaptive weighted sum method in multiobjective simulation optimization. Proceedings of the 2009 Winter Simulation Conference (WSC). 2009. p. 623–33. https://doi.org/10.1109/WSC.2009.5429562.
[21] Denysiuk R, Gaspar-Cunha A. Evolutionary multi-criterion optimization - 9th international conference proceedings. Lecture Notes in Computer Science 10173. Springer; 2017. p. 176–90.
[22] Elarbi M, Bechikh S, Said LB, Datta R. Recent advances in evolutionary multi-objective optimization. Springer; 2017. p. 1–30.
[23] Kumar V, Minz S. Multi-objective particle swarm optimization: an introduction. Smart Comput Rev 2014;4:335–53.
[24] Reyes-Sierra M, Coello CAC. Improving pso-based multi-objective optimization using crowding, mutation and e-dominance. Evolutionary Multi-Criterion Optimization - Third International Conference. Springer Berlin Heidelberg; 2005. p. 505–19.
[25] Reyes-Sierra M, Coello CAC. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput IntellRes 2006;2:287–308.
[26] Fu MC. Optimization for simulation: theory vs. practice. INFORMS J Comput 2002;14(3):192–215.
[27] Pellegrini R, Serani A, Leotardi C, Iemma U, Campana EF, Diez M. Formulation and parameter selection of multi-objective deterministic particle swarm for simulation-based optimization. Appl Soft Comput 2017.
[28] Gosavi A. Simulation-Based optimization: parametric optimization techniques and reinforcement learning. 2 Springer Science + Business Media New York; 2015.
[29] Cristescu C, Knowles J. Surrogate-based multiobjective optimization: Parego update and test. Workshop on Computational Intelligence (UKCI). 2015.
[30] Knowles J. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. Tech. Rep.. University of Manchester; 2004.
[31] Knowles J, Hughes E. Multiobjective optimization on a budget of 250 evaluations. Lecture Notes in Computer Science. 3410. 2005. p. 176–90.
[32] Davins-Valldaura J, Moussaoui S, Pita-Gil G, Plestan F. Parego extensions for multi-objective optimization of expensive evaluation functions. J Global Optim 2017;67(1):79–96. https://doi.org/10.1007/s10898-016-0419-3.
[33] Aghamohammadi NR, Salomon S, Yan Y, Purshouse RC. On the effect of scalarising norm choice in a parego implementation. 9th International Conference on Evolutionary Multi-Criterion Optimization - Volume 10173. New York, NY, USA: Springer-Verlag New York, Inc.978-3-319-54156-3; 2017. p. 1–15.
[34] Hakanen J, Knowles JD. On using decision maker preferences with parego. Evolutionary Multi-Criterion Optimization. Cham: Springer International Publishing978-3-319-54157-0; 2017. p. 282–97.
[35] Zhao L, Choi K, Lee I, Gorsich D. A metamodel method using dynamic kriging and sequential sampling. 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. 2010.
[36] Lee I. Sampling-based rbdo using the dynamic kriging (d-kriging) method and stochastic sensitivity analysis. Struct Multidiscip Optim 2011. https://doi.org/10.1007/s00158-011-0659-2.
[37] Volpi S, Diez M, Gaul NJ, Song H, Iemma U, Choi KK, et al. Development and validation of a dynamic metamodel based on stochastic radial basis functions and uncertainty quantification. Struct Multidiscip Optim 2015;51(2):347–68.
[38] Gu J, Li GY, Dong Z. Hybrid and adaptive meta-model-based global optimization. Eng Optim 2012;44(1):87–104.
[39] Yang Q, Huang J, Wang G, Karimi HR. An adaptive metamodel-based optimization approach for vehicle suspension system design. Math Problem Eng 2014;2014.
[40] Iuliano E. Application of surrogate-based global optimization to aerodynamic design. Springer International Publishing; 2016. p. 25–46.
[41] Diez M, Volpi S, Serani A, Stern F, Campana EF. Advances in evolutionary and deterministic methods for design, optimization and control in Engineering and sciences. Computational methods in applied sciences. Springer, Cham; 2019. p. 213–28.
[42] Bittner F, Hahn I. Kriging-assisted multi-objective particle swarm optimization of permanent magnet synchronous machine for hybrid and electric cars. Electric Machines & Drives Conference (IEMDC), 2013 IEEE International. 2013. p. 15–22.
[43] Todoroki A, Sekishiro M. Modified efficient global optimization for a hat-stiffened composite panel with buckling constraint. AIAA J 2008;46(9):2257–64.
[44] Husain A, Kim K-Y. Enhanced multi-objective optimization of a microchannel heat sink through evolutionary algorithm coupled with multiple surrogate models. Appl Therm Eng 2010;30(13):1683–91. https://doi.org/10.1016/j.applthermaleng.2010.03.027.
[45] Choi S, Alonso JJ, Chung HS. Design of a low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method. Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. 2004.
[46] Voutchkov I, Keane A. Multiobjective optimization using surrogates. Proceedings of the International Conference on Adaptive Computing in Design and Manufacture. 2006.
[47] Regis RG. Evolutionary constrained optimization. Springer India; 2015. p. 51–94.
[48] Vasconcelos JA, RamÃrez JA, Takahashi RHC, Saldanha RR. Improvements in genetic algorithms. IEEE Trans Magnet 2001;37(5).
[49] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 1999;3(4):257–71. https://doi.org/10.1109/4235.797969.
[50] Zitzler E, Brockhoff D, Thiele LT. The hypervolume indicator revisited: on the design of pareto-compliant indicators via weighted integration. In: Obayashi S, Deb K, Poloni C, Hiroyasu T, Murata T, editors. Evolutionary multi-criterion optimization. Berlin, Heidelberg: Springer Berlin Heidelberg978-3-540-70928-2; 2007. p. 862–76.
[51] Jiang S, Ong Y, Zhang J, Feng L. Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Trans Cybern 2014;44(12):2391–404. https://doi.org/10.1109/TCYB.2014.2307319.
[52] Riquelme N, Lucken CV, Baran B. Performance metrics in multi-objective

optimization. XLI Latin American Computing Conference. 2015.
[53] Liefooghe A, Derbe B. A correlation analysis of set quality indicator values in multiobjective optimization. Genetic and Evolutionary Computation Conference (GECCO 2016). 2016.
[54] Davison A, Hinkley D. Bootstrap methods and their application. Cambridge, United Kingdom: Cambridge University Press; 1997.
[55] López-Ibánez M, Dubois-Lacoste J, Cáceres LP, Stützle T, Birattari M. The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 2016.