A new algorithm for the problem of robust single objective optimization

A. NORIEGA, R. VIJANDE, E. RODRÍGUEZ, J.L. CORTIZO AND J. M. SIERRA

University of Oviedo, Department of Mechanical Engineering, Campus Universitario, Edf. Oeste, Módulo 5 s/n, 33203 Gijón, Spain

noriegaalvaro@uniovi.es

Abstract

This paper propounds a new algorithm, the Sub-Space Random Search (SSRS) for the problem of single-objective optimization, with the aim of improving the robustness and the precision of classical methods of global optimization. The new algorithm is compared with a genetic algorithm (GA), on a set of four scaleable test functions and with the number of variables changing from 1 to 5. A new test function called Deceptive-bimodal (DB) is proposed. Results indicate that, with the same total number of function evaluations, SSRS is about 50% faster than GA. Moreover, SSRS shows a greater precision and similar ability to find the global optimum than GA with 1, 2 and sometimes 3 variables. But this advantage diminishes when the number of variables increases on multimodal and narrow-flat valley functions. Finally, SSRS is successfully applied to a problem of dynamical synthesis of a mechanism.

Keywords: *meta-heuristic, unconstrained optimization, stratified random search, synthesis of mechanisms*

A. Noriega., R Vijande, E Rodríguez, J-L Cortizo, J-M Sierra, A new algorithm for the problem of robust single objective optimization. Int. J. Simul. Multidisci. Des. Optim. 2, 223–229 (2008)

DOI 10.1051/ijsmdo:2008030

The original publication is available at www.ijsmdo.org

http://www.ijsmdo.org/index.php?option=com_toc&url=/articles/smdo/abs/2008/03/contents/contents.html

1. Introduction

Stratified Sampling Methods are widely used today in the design of experiments and allow the amount of important information which can be extracted from the output data of a process to be maximized, see [1]. The application of these methods with refinement on global optimization problems allows regions with a high probability of containing the global optimum to be identified. Moreover, the finer the stratification is, the minimum is delimited with more precision.
The paper propounds a meta-heuristic optimization algorithm based on a search space stratification similar to the methods cited before. But the sampling and the choice of the zone where the resolution must be increased is completely different to those methods.
The paper is organized as follows: In section 2, a detailed description of algorithm implementation and its basic concepts is given. In section 3, the experimental validation of this algorithm is described. Test conditions and functions are indicated in it, including a new test function proposed and an application to a problem of dynamical synthesis of a mechanism. In section 4, the results are shown and discussed, extracting some conclusions about the new algorithm and possible future work-lines.

2. Algorithm description

For this algorithm, it is supposed that the optimization problem is formulated as follows:

$$\min\ f(\vec{x}) \qquad (1)$$

with $\vec{x} \in \Re^n$ and $l_i \le x_i \le u_i$ being $i = 1,...,n.$

The algorithm Sub-Space Random Search (SSRS) is based on the hypothesis that there is a search space stratification in sub-spaces of equal size and whose union makes the initial space of every iteration. In every sub-space, a sample of individuals are generated and evaluated to get a measure of the behaviour of the objective function in this sub-space To increase the resolution in the optimum search, selecting the most promising sub-space and re-initiating the process on it is proposed. Figure 1 shows a graphic representation of this idea with an example of 2 variables.
However, to do this refinement process, it is necessary to determine a reference value which estimates the probability that a sub-space contains the global minimum. This reference value allows comparing it with other sub-spaces and thus being able to select the most suitable to continue the process. To calculate this reference value, a random sample of individuals is generated and evaluated in every sub-space because a random sample is the easiest way of generating individuals being the sample size is a parameter defined by the user. Furthermore, this random sample has the same average probability of approaching the unknown minimum of the sub-space as any other method of generating individuals as is shown in [2]. Finally, this randomized approach to estimate the reference value, is similar to Monte-Carlo techniques used in robust optimization [3].
Then, the minimum value of the individuals' evaluations can be taken as a reference value for the comparison of sub-spaces. Or the average value of the evaluations of all the individuals of the sample can be calculated. Even the average value of p % smallest values of the individuals' evaluations (p being a parameter) can be taken. A priori, it is not known which one is the best option to calculate the reference value because it depends on the landscape of the objective function in this sub-space. For a continuous and monotonous function, a sample with few individuals generates a probably correct estimation of the function behaviour in the sub-space. But the same thing does not happen with a multimodal function (with several minima) since there exists the probability of not generating individuals in the influence area of the global minimum of sub-space

when it represents a small percentage regarding the whole space. For the influence area of a minimum, we understand the minimum's neighborhood where the function is unimodal.

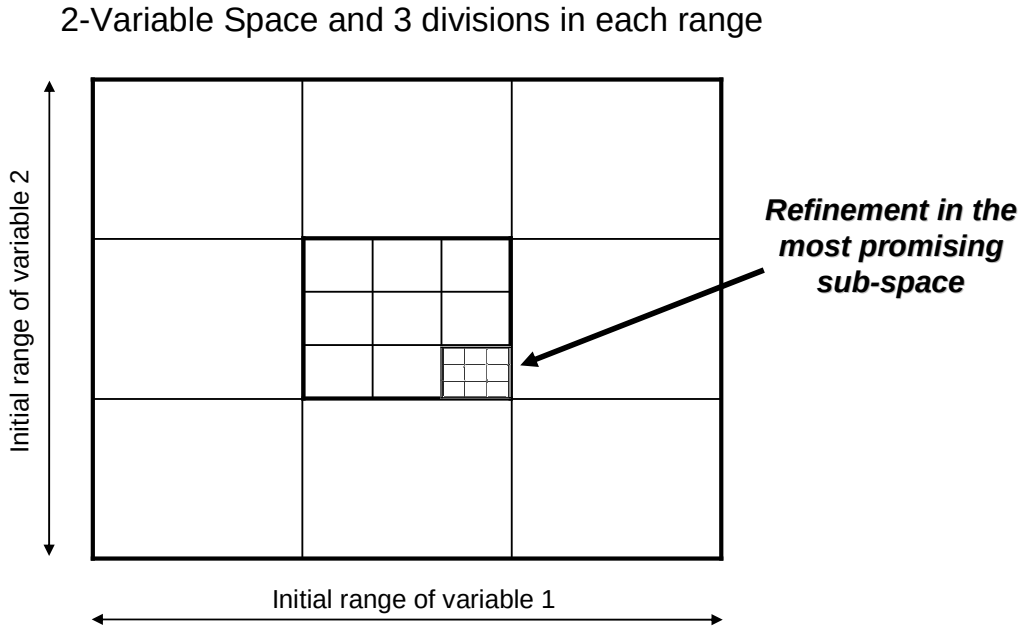

Figure 1: Reiterated stratification and selection of the most promising sub-space

The problem is that, for having a precise description of the behaviour of the function in the sub-space and, therefore, the high security that the election of the most suitable region is correct, it is necessary to increase the size of the random sample a great deal which will penalize the general efficiency of the search. However, this action improves the robustness of the search to Type II variations just as they are referred in [4].
The number of selected individuals to calculate the reference value (parameter *p*) can be modified to compensate that difficulty without increasing the size of the sample in every sub-space. If the sample is poor, a small number of individuals will be selected. In the case of having a big sample size, the number of selected individuals must increase to improve the security of the estimation.
But the landscape of the objective function in the sub-space also has influence on choosing the number of individuals selected to calculate the reference value. If there are sharp minima, it is necessary to generate a sample with a big number of individuals and select a small number of them to increase the probability of obtaining almost one individual closer to these minima.
These values directly influence the efficiency of the algorithm and its robustness and globality in an opposed way. However, it would also allow the search towards a type of certain minimum (more or less sharp) to be guided.
The stratification of the variable space is an important factor since the more divisions in the range of every variable, the finer the global stratification is. Therefore, the search is more global because the complexity of the function in every sub-space diminishes and, then, it is easier to have a correct evaluation of them. But it is a very delicate parameter since it has a negative influence on the efficiency of the algorithm, because it increases the number of sub-spaces exponentially with the number of variables. For instance, if there are three divisions in every variable's range of a two-dimensional space, the result is $3^2 = 9$ sub-spaces to study, but if the space is three-dimensional, the result is $3^3 = 27$ sub-spaces. Therefore, this is the most critical parameter in the algorithm and hence it would limit its application to optimization in spaces with many variables.

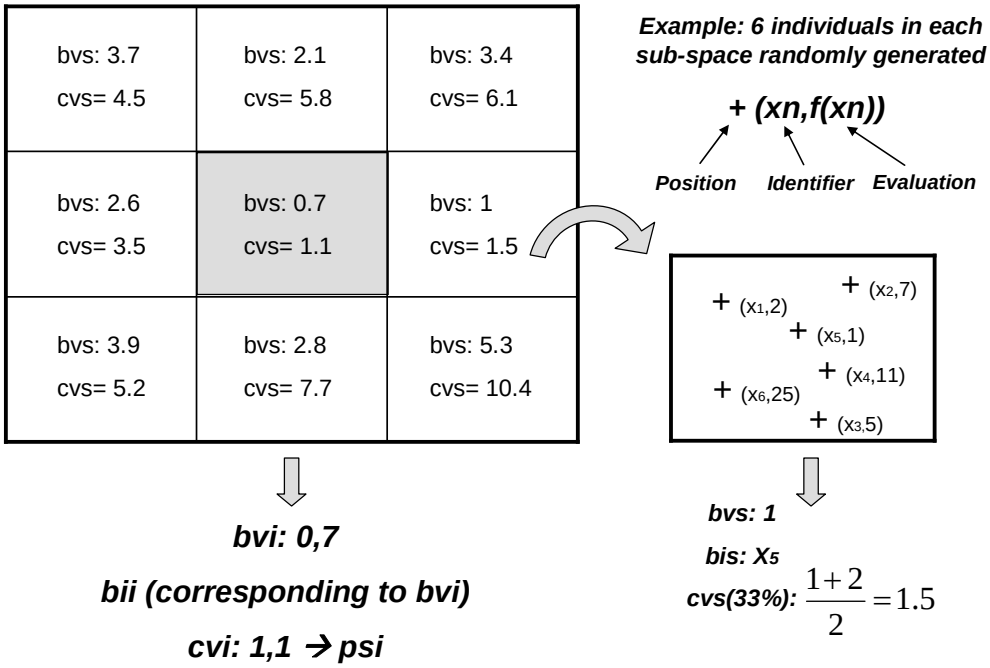

Figure 2: Sub-space evaluation and selection of the most promising one

Thus, the sample's individuals are evaluated to calculate three values in every sub-space: the smallest value obtained in the objective function (*bvs*), the individual of the sample corresponding to this minimum (*bis*) and the comparison value for the sub-space (*cvs*). This last value is calculated taking the mean of the *p* % of smallest values of the objective function, *p* being a parameter defined by the user.

Once these three values of every sub-space have been obtained, the sub-spaces are compared among them to obtain the final values of this iteration: the best individual obtained (*bii*), its corresponding value of the objective function (*bvi*) and the most promising sub-space to look for the global minimum of the function (*psi*). It can be seen in Figure 2 with an example.

The sub-space with smaller *cvs* is considered as the most promising and the process explained above is repeated again on it. The number of times that this refinement process is repeated, is externally set by the user, so that in every step, the precision of the obtained solution increases. The flowchart of the algorithm is shown in Figure 3.

Then, the proposed algorithm has 4 parameters:

Number of divisions per variable (*ndv*): This parameter is related with the complexity of the objective function and it is good that it is a small integer number because the number of sub-spaces depends on it by means of the following equation:

$$t = ndv^n \qquad (2)$$

where *n* is the number of variables.

Population size (*ps*): This parameter indicates the number of individuals randomly generated in every sub-space. It measures the quality of the function description in every sub-space and it is also related with the complexity of the objective function.

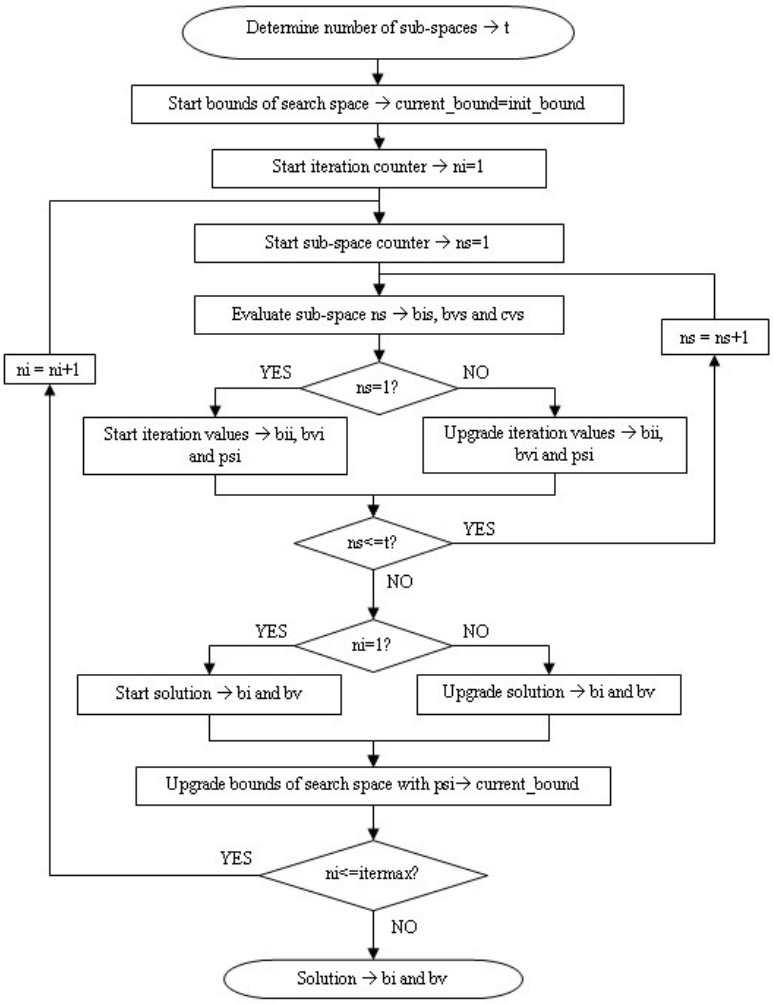

Figure 3: Flowchart of SSRS

% pop for reference value (*p*): This parameter indicates the % of sample smallest values taken to calculate the comparison value of every sub-space (*cvs*). This value is calculated by means of the average of selected values.
Number of iterations (*itermax*): This parameter indicates the number of iterations that will be carried out; it has to be not too big an integer because this algorithm converges very fast to the minimum.

The best individual obtained (*bii*) will usually be in the most promising sub-space (*psi*) but it may not be so. In this case, it is recommended to increase the population size because the function can have a very complicated landscape in the search space.
The whole explained process has the advantage of allowing a very simple implementation. Furthermore, it also reduces the number of the individuals' comparisons to small groups corresponding to the sub-spaces populations and later among sub-spaces. The result is that the general speed of this algorithm is increased.

3. Experimental validation

The validation of SSRS will be done by means of a benchmarking of this algorithm and another similar algorithm. In the field of robust optimization, there are two main philosophies [5]. The first one is called *simplification strategy* whose aim is the transformation of the problem to another that can be solved using standard techniques of mathematical programming. Its drawback is that these techniques are usually of local search and they need additional information like derivatives of objective function. On the other hand, there exist the *simulation optimization techniques* which are less efficient than the first ones but they do not need additional information and, moreover, they can use randomized approaches, like Genetic Algorithms, which can do a global search.
Because of the populational nature of SSRS and its general guidance, it is considered to compare it with an algorithm with similar features, selecting a Genetic Algorithm (GA) as a competitor. In this case, Genetic Algorithm and Direct Search Toolbox of MATLAB ® is used.

3.1 Test functions

To test the SSRS, two types of test are proposed. First, standard test funtions are used. To select them, the basic features of a function to optimize are identified. Then, four benchmark functions which represent some of these features have been selected [6]. These functions will be scaleable, i.e., the number of variables of the function (*n*) can be changed but the features are maintained. These features are:
Number of minima: The number of local and global minima and their relative values determine the complexity of the search and they allow the robustness of the optimization techniques to be verified. A distinction can be made between Unimodal (a single minimum) and Multimodal (two or more minima). For these types, De Jong's function F1 is selected as unimodal function (UM) and Rastrigin's function as multimodal function (MM).
Narrow and flat valley: The minimum is located in a very narrow and sharp valley with an almost flat bottom in which is very difficult to determine whether you have arrived or not at the proximity of the minimum. For this type, Rosenbrock's function (NFV) is selected.
Flat surfaces: Flat surfaces are obstacles for optimization algorithms based on derivatives, because they do not give any information of which direction is favourable.
Deception: This feature arises when in a multimodal function the global minimum influence area represents a small percentage of whole variable space which is dominated by local minima influence areas. This feature complicates the search a great deal and it allows the robustness of an optimization technique to be verified. For this type, the Deceptive-bimodal function (DB) is proposed, inspired by one in Deb [7] but with different parameters and modified to make it scaleable. It is a bimodal function with a local minimum in $x_i = 7$ with a big influence area and a global minimum in $x_i = 1$ with a small influence area and where *n* is the number of variables. A graphic representation in 2 variables is shown in Figure 4.

$$f(\vec{x}) = 1,2 - e^{-10\cdot\sum_{i=1}^{n}(x_i-1)^2} - 0,7\cdot e^{-0,1\cdot\sum_{i=1}^{n}(x_i-7)^2} \qquad (3)$$
$$0 \le x_i \le 10$$

The function value in the global minimum is $0,2 - 0,7\cdot e^{-3,6\cdot n}$.

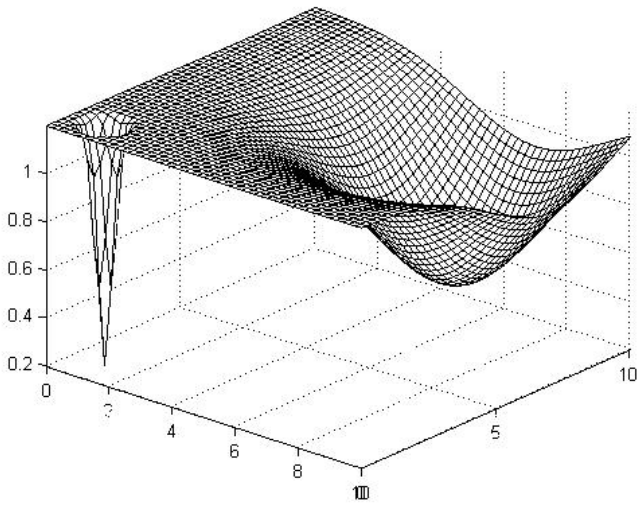

Figure 4: Deceptive-bimodal function in 2 variables

In second place, a real optimization problem is proposed. In this case an application to a dynamical synthesis of a mechanism is selected. The mechanism is a very simple system composed by a mass, a spring and a damper is proposed, see Figure 5. The mass can move along the Y axis by means of a vertical guide. The problem consists in adjusting the values of the mass (M), the spring stiffness (K) and the damper coefficient (C) so that the acceleration in Y of the mass has a certain response regarding time. To define this response and to ensure that the problem has a well-known solution, a previous simulation of the system is made with the following variables: M=30 *kg*, K=3 $N \cdot mm^{-1}$ and C=0.1 $N \cdot s \cdot mm^{-1}$. A graphical view of this response can be seen in Figure 5. The objective is to minimize the average of absolute errors in every simulation frame between the obtained and the desired responses. The bounds of the variables are the followings: $M \in [10\ 50]$, $K \in [1\ 10]$ and $C \in [0.01\ 1]$.

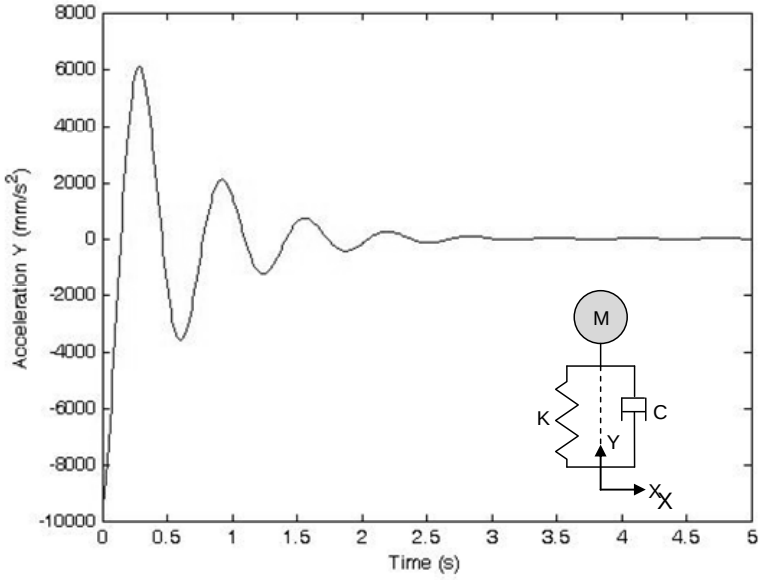

Figure 5: Dynamical system and desired response

3.2 Test conditions

A benchmark comparison between the SSRS and a GA is made on the group of test functions cited above. With the standard test funtions, the number of variables changes from 1 to 5 in a similar way as can be seen in Elbeltagi [8] except for Rosenbrock's function which varies from 2 to 5 variables because it is not defined for 1 variable. Experiments are repeated 100 times. Both the average error and its standard deviation and the average run time are calculated as in Deb [9].
Parameters which control the population size and the generations in both algorithms are tuned so that, keeping logical values for every test problem, the total number of function evaluations are equal in both algorithms. These parameters are shown in Table 1.
In the case of dynamical synthesis, there is a simulation of the behaviour of the system done with ADAMS ® in the objective function. Since, the time cost of the objective function evaluation is around 4 seconds, only one run with both algorithms will be done. The problem is solved with SSRS with the following parameters: *ndv=3, ps=20, p=10%* and *itermax=4,* using 2160 function evaluations while GA uses a population size of 48 individuals and 45 generations with the same total function evaluations.
The implementation of SSRS and benchmark test has been carried out in MATLAB ® 7.0. The computer used for the tests has a processor Intel ® Celeron ® 2.0 GHz and 512 Mb of RAM.

4. Results and discussions

The outcomes of the benchmarking with the four standard test functions are the mean and the standard deviation of the absolute error and the mean of runtime. The error mean gives an idea of the value of the minimum reached. It can be related with the globality and the precision of the algorithm while the standard deviation gives an idea of the repeatability of the algorithm. In the case of runtime value, comparison among the averages obtained by the algorithms gives an idea of the cost of their internal operations since the number of function evaluations are the same in both cases.
Thus, the most important result which can be seen in Table 2 is that SSRS is faster than GA for all the cases. SSRS run time is about 40-60% less than GA run time with an equal number of function evaluations over 3, 4 and 5 variables. This ratio is even better for 1 and 2 variables.
Watching mean and standard deviation of error, a different behaviour is observed on different test functions. On the UM function, algorithm SSRS shows a greater precision than GA. On NFV function, SSRS presents a better behaviour than GA for 1, 2 and 3 variables but for 4 and 5 this tendency is inverted and the advantage of GA over SSRS increases as the number of variables increases.
On the MM function something similar happens. With 1 and 2 variables, SSRS does a more global search and obtains better precision than GA but this behaviour changes when the number of variables increases. In this case, GA works better with more variables while SSRS shows the opposite behaviour.
On the DB function, both algorithms present a similar behaviour, converging to the local minimum almost every time but SSRS has better repeatability than GA.
To have an idea about the robustness it would be possible to see the evolution of the *cvs* in the most promising sub-space and to relate this value with the size of the sub-space in every iteration.
For the problem of dynamical synthesis, the SSRS obtains the best solution M=31.269, K=3.132 and C=0.103 with an error of 7.066. The GA obtains the best solution M=49.260, K=4.912 and C=0.167 with an error of 13.351. Then, the SSRS is clearly better than the GA obtaining the correct solution with more precision than the GA and in a more efficient way.

Table 1. Algorithm parameters

Function	Var.	SSRS				GA			Function Eval.
		ndv	ps	p	itermax	Pop	Gen	Elite	
UM	1	2	10	25	5	10	10	1	100
	2	2	10	25	5	20	10	1	200
	3	2	20	25	5	50	16	1	800
	4	2	50	25	5	100	40	1	4000
	5	2	100	25	5	200	80	1	16000
NFV	2	3	20	10	2	20	18	1	360
	3	3	50	10	2	50	54	1	2700
	4	3	100	10	2	100	162	1	16200
	5	3	200	10	2	200	486	1	97200
MM	1	3	20	10	2	10	12	1	120
	2	3	20	10	2	20	18	1	360
	3	3	50	10	2	50	54	1	2700
	4	3	100	10	2	100	162	1	16200
	5	3	200	10	2	200	486	1	97200
DB	1	5	20	1	2	20	10	2	200
	2	5	20	1	2	50	20	2	1000
	3	5	40	1	2	100	100	2	10000
	4	5	60	1	2	500	150	2	75000
	5	5	80	1	2	1000	500	2	500000

Table 2. Results

Function	Var.	SSRS			GA		
		Error Mean	Error Std	Time Mean	Error Mean	Error Std	Time Mean
UM	1	0.0023	0.004735	0.0175	0.0697	0.156271	0.1267
	2	0.0122	0.011414	0.0371	0.2137	0.298253	0.1356
	3	0.0172	0.011193	0.1223	0.2611	0.238042	0.2717
	4	0.0233	0.012047	0.5502	0.2076	0.154413	1.0549
	5	0.0277	0.012261	2.0028	0.2083	0.141052	3.8506
NFV	2	0.2501	0.351123	0.0498	0.2701	0.292721	0.1679
	3	0.8793	0.274383	0.3312	1.1223	0.875923	0.7321
	4	1.8731	0.398454	1.7983	1.3866	1.081146	3.8459
	5	3.0801	0.410591	10.8642	1.7645	0.969627	20.0491
MM	1	0.3578	1.133234	0.0183	1.5934	1.618239	0.1411
	2	2.1151	1.719442	0.0619	2.7411	2.079256	0.1869
	3	2.3418	1.001727	0.4152	1.7676	0.917453	0.8573
	4	3.3164	1.117348	2.5969	0.9946	0.664095	4.5358
	5	4.1791	1.206706	14.2511	0.3617	0.284541	25.9321
DB	1	0.1216	0.112855	0.0339	0.1381	0.099891	0.1133
	2	0.2011	0.000224	0.1725	0.1931	0.031975	0.3522
	3	0.2005	0.000288	1.6128	0.1985	0.017446	2.6995
	4	0.2009	0.000449	11.6342	0.2039	0.004432	19.1681
	5	0.2013	0.000572	75.7334	0.2052	0.014621	124.8826

5. Conclusions and future work-lines

A new meta-heuristic algorithm, the Sub-Space Random Search, is proposed for the global optimization problem and a benchmarking with a GA is made to estimate its behaviour. This benchmark is made on a basic casuistry of scaleable optimization problems with the number of variables changing from 1 to 5.
Results indicate that SSRS shows a greater precision than GA on unimodal functions. On multimodal and narrow and flat valley functions its precision is better on low dimensions but it get worse as the number of dimensions increases. On deceptive function the behaviour of both algorithms is similar. An application of SSRS to a problem of dynamical synthesis of mechanism also shows the better performance of this algorithm regarding a GA in problems with a few variables. Furthermore, SSRS is faster than GA in all the cases studied and it is worth noting the great simplicity of the code needed to implement this algorithm.
Then, the SSRS is showed as a good algorithm for relatively easy problems (unimodal and low multimodality functions) with few variables since it can get the global optimum and it also allows studying the robustness during the process.
At present, the authors are working on improving this behaviour when the number of variables increases and on implementing a parallel search in several equal-promising sub-spaces.

Acknowledgments
The first author is supported by the Ministry of Education of Spain (FPU grant AP-2004-6492)

References

[1] Tong C., Refinement strategies for stratified sampling methods, *Reliability engineering and system safety* **91** (2006) 1257--1265.

[2] Wolpert D.H., MacReady W.G., No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation*, **1** (1997) 67--82.

[3] Andradóttir S., A review of simulation optimization techniques, *Proceedings of the 1998 Winter Simulation Conference*, IEEE, Piscataway, NJ, (1998) 151--158.

[4] Chen W., Allen J., Tsui K.-L., Mistree F., A procedure for robust design: Minimizing variations caused by noise factors and control factors, *ASME Journal of Mechanical Design* **118** (4) (1996) 478--493.

[5] Beyer H-G., Sendhoff B., Robust optimization – A comprehensive survey, *Computer Methods in Applied Mechanics and Engineering* **196** (2007) 3190--3218.

[6] Goldberg D., Genetic algorithms in search, optimization, and machine learning, New York: *Addison-Wesley*, 1989.

[7] Deb K., Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, *Evolutionary Computation* **7** (3) (1999) 205--230.

[8] Elbeltagi E., Hegazy T., Grierson D., Comparison among five evolutionary-based optimization algorithms, *Advanced Engineering Informatics* **19** (2005) 43--53.

[9] Deb K., Pratap A., Agarwal S., Meyarivan T., A fast and elitism multi-objective genetic algorithm: NSGA-II, *Technical Report Nº 2000001*, Kanpur Genetic Algorithm Laboratory, 2000.