A sensitivity analysis indicator to adapt the shift length in a metaheuristic

Peio Loubière
CY Cergy Paris University
Pau, France
plo@eisti.eu

Astrid Jourdan
CY Cergy Paris University
Pau, France
aj@eisti.eu

Patrick Siarry
LISSI
UPEC
Vitry-sur-Seine, France
siarry@u-pec.fr

Rachid Chelouah
CY Cergy Paris University
Cergy, France
rc@eisti.eu

***Abstract*— Population based metaheuristics (e.g. Genetic Algorithm, Particle Swarm Optimization, …) deal with a dichotomy between exploration (discover unexplored areas) and exploitation (dig around a good solution). The consequence is a wide exploration of the search space. A lot of information about the link between the objective function and the input variables is collected during the algorithm. Sensitivity analysis methods allow to transform this information in order to characterize the effect of an input variable on the objective function: linear impact, nonlinear impact, negligible impact. We propose to integrate a sensitivity analysis method in the optimization process in order to increase or decrease the shift length when offsetting a variable according to its behavior. The offset of a variable with a nonlinear impact has to be small in order to catch possible local optima of the objective function. On the contrary, the offset of a variable with a linear impact has to be high in order to move faster the variable toward its best position. A toy example is used to illustrate the interest of the method.**

Keywords—optimization, metaheuristics, sensitivity analysis, convergence speed

I. Introduction

Metaheuristics are strategies that guide a global optimization process of nonlinear objective functions. The goal is to efficiently explore the search space in order to find near–optimal solutions. Metaheuristics deal with a dichotomy between exploration, to discover unexplored areas, and exploitation, to dig around a good solution. Population based metaheuristics (e.g. genetic algorithm, particle swarm optimization, …) ensure a wide exploration of the search space.

Starting from a set of initial points, the metaheuristic iterations randomly explore the neighborhood of each point. A neighbor is generated by offsetting a set of variables of the current point. The variables and the offset are randomly chosen. We propose to use the information gathered during the iterations for guiding the algorithm toward cleverer choices. The idea is to use the points evaluations to characterize the variables behavior (relevance and shape) thanks to a sensitivity analysis (SA) method [2][13].

Sensitivity analysis is the study of how the input variables affect an output variable. In optimization, SA methods are often used to eliminate non influential variables before the process, thereby reducing the dimension. In this paper, we present an another way to use the information given by a SA method for improving the convergence of the metaheuristic. We assume that proceeding in two steps may not be suitable. Removing variables definitively can be damaging to the optimization process. Irrelevant variables at the beginning of the algorithm may become relevant further in the search process and could discriminate the points (during the exploitation process).

Moreover, sensitivity analysis requires a lot of evaluations of the objective function, as the metaheuristic. Directly integrating a sensitivity analysis method in a metaheuristic saves evaluations and allows to focus on relevant variables. The goal is not the computation of accurate sensibility analysis indices, but to obtain enough information on the variable behavior in order to guide the optimization process.

According to their search process, metaheuristics can be split into three families: those which search along one direction (variable), those which select a subset of variables and those which search along all directions. For the two first families, SA would help to focus on the most influential variables. In all cases the information about variable behavior (monotony, non-linearity, ...) would help adapting the offset when generating a new neighbor.

In a previous work [9], Morris' sensitivity method [4], has been integrated in Artificial Bee Colony (ABC) algorithm [4][5]. ABC algorithm integrates fairly well Morris' method, because of its one-direction neighborhood search process. They both offset a point according to a single variable at time and analyze the impact on the objective function output.

Among all search processes implemented in metaheuristics, ABC neighborhood search is a particular case. Many metaheuristics algorithms search a neighbor in a hyper-sphere, offsetting various variables at a time (tabu search [1], differential evolution DE [11], swarm intelligence based metaheuristics such particular swarm optimization PSO [6][12]).

In a second work [10], we generalized this approach. We defined a new sensitivity analysis method adapted to a multidimensional neighborhood context. This method was successfully integrated in a second family algorithm (DE). The SA performed during the algorithm allows to compute for each variable a weight proportional to the impact of this variable on the objective function. The uniform random selection of the variables to offset is then replaced a random selection based on the weights. The most influential variables have more chance of being selected.

In these previous works, we only used the information given by the SA on the variable influence, but not the one about its behavior (linear or nonlinear). Moreover, the influence information is not relevant with algorithms of the third family, such as PSO. Indeed, this kind of algorithm generates a neighbor by offsetting all variables on the same time, regardless of their influence. In this paper, we propose to integrate as well the knowledge about the variables behavior to adapt the offset value for each variable. Non-linearity leads to a small shift and, on the contrary, monotony leads to a large shift.

Section 2 reminds the naive sensibility method defined in [11]. In Section 3, we explain how we use the SA information to automatically regulate the shift during the process. In section 4, we present a simple example to illustrate the method. Section 5 is dedicated to the conclusion and some future works.

II. A NEW INDICATOR FOR LINEARITY

In [11], we defined a naive sensitivity analysis method able to

- identify non influent variables,
- rank the variables,
- determine the behavior of each input variable,

in a multidimensional neighborhood context. The SA was defined according to the following constraints:

- the sensitivity analysis method has to cope with complex functions since it will be used in the context of hard optimization,
- the sensitivity analysis method must not have experimental design constraint since the evaluation points are given by the initialization and the iterations of the algorithm,
- the sensitivity analysis has to deal with a reasonable number of evaluations since metaheuristics usually have a limited number of function evaluations,
- the sensitivity indices must be very fast to calculate in order to be integrated into the optimization algorithm.

Our method is based on Morris' method [8] defined for screening important input variables of a nonlinear model with few evaluations of this model. The elementary effect method can detect influential variables with linear or nonlinear effects.

Considering $X = (X_1, \ldots, X_D)$ a point in the search space, the elementary effect of the j^{th} input variable on a model f is calculated by

$$EE_j = \frac{f(X_1,\cdots,X_j+\Delta,\cdots,X_D)-f(X)}{\Delta}$$

where Δ is an offset. A trajectory is composed of successive offsets, one input variable at a time (OAT). It means that, at each step, the offset shifts the point in only one dimension (as described in Figure 1). This figure is an example in a three-dimensional process. It iterates three times: at step j, an offset Δ_j is applied on an input variable X_j of a point X, producing another point X'. One trajectory allows to compute one elementary effect for each variable. The sensitivity of each variable is then evaluated by the absolute mean m_j^* and the standard deviation s_j of the elementary effects computed with N trajectories,

$$m_j^* = \frac{1}{N}\sum_{i=1}^{N}\left|EE_j^i\right| \quad \text{and} \quad s_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(EE_j^i - m_j^*\right)^2}\ .$$

A large value for m_j^* indicates a linear effect of X_j and a high value of s_j indicates a nonlinear effect. If X_j has no impact on f function the values m_j^* and s_j are small (Figure 2).

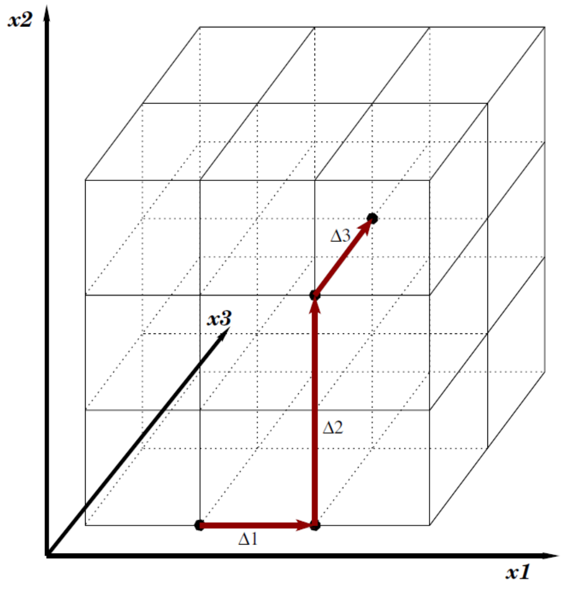


Fig. 1. *A trajectory for the Morris 'method in dimension D=3.*

In most of the optimization algorithm the offset of X implies more than one variable. The computation of the elementary effects is no more possible. The simple idea is to replace the elementary effects in the Morris' method by the linear (or rank) correlation coefficients computed in a local neighborhood of a set of evaluation points.

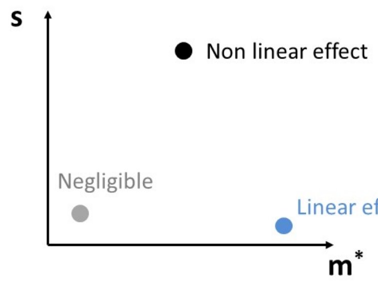
Given a set of N points in the D-dimensional search space. Denote Y the vector with the evaluations of the objective function at the N points. We randomly consider k points ($k < N$) and we select, for each point, the p nearest neighbors to define k neighborhoods. For each variable $X_j, j \in \{1, \ldots, D\}$, and each neighborhood N_i, $i \in \{1, \ldots, k\}$, we compute the linear (or rank) correlation coefficients :

$$\mathrm{CC}_{N_i}^j = \rho(X_{N_i}^j, Y_{N_i})$$

where $X_{N_i}^j$ and Y_{N_i} are the variables X_j and Y restricted to N_i neighborhood and ρ is the linear (or rank) correlation coefficient between the variables.

As done in Morris' method, the normalized mean (m_j^*) and standard deviation (s_j) of the local coefficients are computed, by variable. Figure 2 illustrates the meaning of m_j^* and s_j on a simple example. The upper part of the figure draws the function for each variable $X_j, j \in \{1,2,3\}$. The red points are the k points used to define the neighborhoods and the arrows represent the coefficients of correlation computed for these neighborhoods.

As for the Morris method, a large value of m_j^* indicates an influential variable with a monotonic effect, and a large value of s_j indicates an influential variable with a non-linear effect.



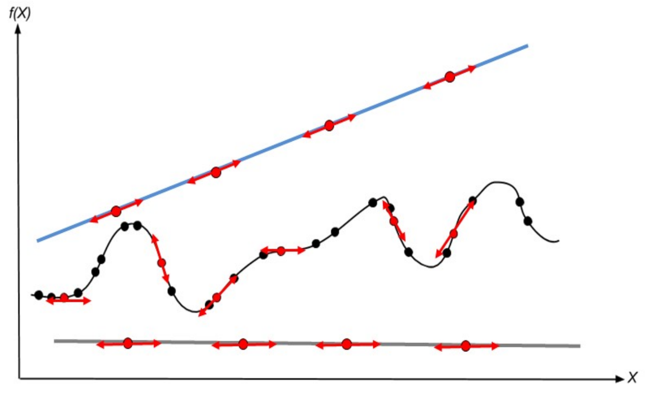


Fig. 2. *Bottom: A function with a negligible variable (grey), a linear impact of x (blue), a non-linear impact of x (black). Up: The corresponding position in the map (m^*, s).*

The indices m_j^* and s_j are very fast to calculate and give enough information for guiding the optimization process. In [11], we used them to evaluate whether a variable has an impact on the objective function during the iterations of the algorithm. The jth variable influence is given by the weighted distance d_j from the pair (m_j^*, s_j) to the origin

$$d_j = \sqrt{m_j^{*2} + s_j^2}\ .$$

The weight of the jth variable is given by

$$w_j = d_j / (\sum_{k=1}^{D} d_k)\ .$$

The weights replace the constant rate in DE algorithm and give higher probability to be selected for influential variables. Each variable is offset according to the equation,

$$u_j^{t+1} = \begin{cases} v_j^{t+1} & \text{if } random < w_j \\ x_j^t & \text{if } random > w_j \end{cases}$$

where $v^{t+1} = x_{r1}^t + k(x_{r2}^t - x_{r3}^t)$ and r_1, r_2, r_3 are three point indexes, k is the shift constant.

By doing this, the information about the linearity or non-linearity of the variable is lost. Indeed, a linear variable with a high m_j^* and small s_j (blue point in Figure 2) has the same weight than a nonlinear variable with a high s_j and small m_j^* (black point in Figure 2). A new indicator is defined by

$$\delta_j = \frac{m_j^{*2}}{s_j}\ .$$

Then, $\delta_j > 1$ (large m_j^* and small s_j) if the jth variable has a high linear effect, and $\delta_j < 1$ (small m_j^* and large s_j) if the jth variable has a high non-linear effect or if the jth variable is negligible.

III. Including the New Indicator in a Metaheuristic

Most of the optimization processes use a shifting strategy for the variable to converge toward the best solution. The idea is to increase the shift length when the variable has a linear effect on the objective function to go directly to the optimum position.

On the contrary, in the case of a non-linear impact, the length of the offset must be reduced in order to test any irregularities in the function. In order to illustrate this strategy, we integrate the indicator defined above in a Particle Swarm Optimization.

Particle Swarm Optimization (PSO) is a population based global optimization algorithm introduced by Eberhart and Kennedy [6]. For understanding the metaphor, the swarm of particles represents the population. Each particle has a position (the point) and a speed (a vector of shifts to be applied to each point's variable). Also, a particle has small memory but is highly communicant: it reminds its own best past position and shares position information with others to determine which particle belongs to the best position.

The search is processed by updating all information (communication step) and moving each particle across the search space according to a new speed computation (compute and apply the speed to the current position (Equation 1) and (Equation 2), and the global best position of the swarm is updated).

$$v^{t+1} = \omega v^t + c_1 r_1 (x_{best}^t - x^t) + c_2 r_2 (g_{best}^t - x^t) \qquad (1)$$

$$x_j^{t+1} = x_j^t + v_j^{t+1},\ \forall j \in \{1,\dots,D\} \qquad (2)$$

where ω, c_1, c_2 are constants, r_1 and r_2 are random numbers in [0,1], and x_{best} and g_{best} are individual and neighborhood best positions.

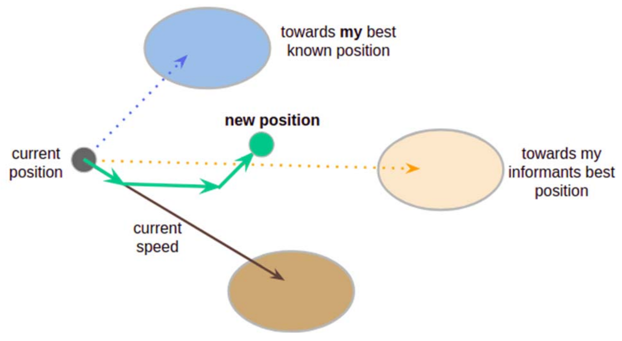

towards **my** best known position
new position
current position
towards my informants best position
current speed

Fig. 3. *PSO moving schema.*

As shown in Figure 3 and (1), the computation of the new speed is a linear combination of the current speed, the direction toward the particle personal best past position and the informants best position. All the variables will be shifted according to the corresponding speed vector values. In order to increase or decrease the shift length according to the variable behavior, we modify (2) as:

$$x_j^{t+1} = x_j^t + \boldsymbol{\delta}_j v_j^{t+1}, \forall j \in \{1,\ldots,D\} \quad (3)$$

If the jth variable has a high linear impact, then $\delta_j > 1$ and the shift length is higher in (3) than in (2),

$$\delta_j \left\| v_j^{t+1} \right\| > \left\| v_j^{t+1} \right\|.$$

The jth component of the current position x_t goes faster toward its best position. On the contrary, if the jth variable has a high nonlinear impact, then $\delta_j < 1$ and the shift length is smaller in (3) than in (2),

$$\delta_j \left\| v_j^{t+1} \right\| < \left\| v_j^{t+1} \right\|.$$

The jth component of the current position x_t moves slower toward its best position in order to catch possible irregularities of the objective function.

IV. A TOY EXAMPLE TO ILLUSTRATE THE SHIFTING STRATEGY

The usual benchmarks do not make it possible to illustrate the interest of this new shifting strategy. Indeed, the method is useful only if the function has linear and non-linear influential variables. We use the following toy example in dimension 5,

$$f(x) = 3.5(w_1 - 1) + \sum_{i=2}^{3} \left[(w_{i-1} - 1)^2 (1 + 20\sin^2(\pi w_i + 1))/10 + \sin^2(\pi w_i) \right] + 0 \times x_4 + 0 \times x_5$$

with $w_i = 1 + (x_i - 1)/4$ and $x_i \in [-15\,;15]$.

The minimum is $f(-15,1,\ldots,1) = -14$. As illustrated in Figure 4, x_1 has a high linear impact, x_2 and x_3 have a high non-linear impact and x_4 and x_5 are negligible.

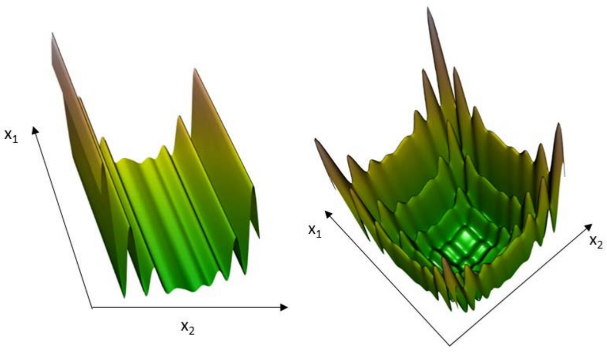


Fig. 4. Toy example. x_1 vs x_2 (up-left) and x_3 vs x_2 (up-right).

In order to estimate m^* *et* s^*, we evaluate the function f at 30 points x evenly spread in the search space according to a space-filling design [3]. Their position in the map (m^*,s) are given in Figure 5. The resulting indicators δ_j are given in Table 1. The indicator δ_1 is greater than one and increases the length shift in (3).

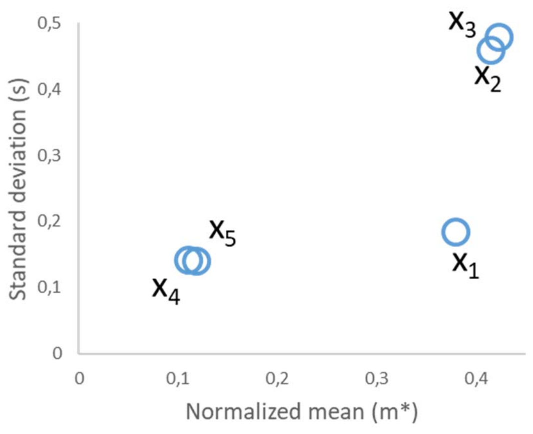


Fig. 5. *Map* (m^*,s) *for the toy example*

Since x_1 has a linear effect on the objective function, there is no need for many x_1 shifts to reach its optimal position. On the contrary, x_2 and x_3 have a non-linear impact, and small shifts are required. We can see in Table 1 that δ_2 and δ_3 are lower than one and decrease the length of the shift.

TABLE I. INDICATOR VALUES FOR THE TOY EXAMPLE

δ_1	δ_2	δ_3	δ_4	δ_5
1.20	0.36	0.38	0.10	0.09

We run the PSO with the toy example with 20 starting points and 175 iterations. The evaluations of m^* and s are computed after 25 iterations. We repeat the PSO 30 times in order to evaluate the fluctuations of m^* and s estimations. The 30 resulting δ_j are summarized in the boxplots Figure 6. We note an overestimation of δ_1 due to the fact that the points use to compute m and s are not evenly spread in the search space since they are determined by the optimization process.

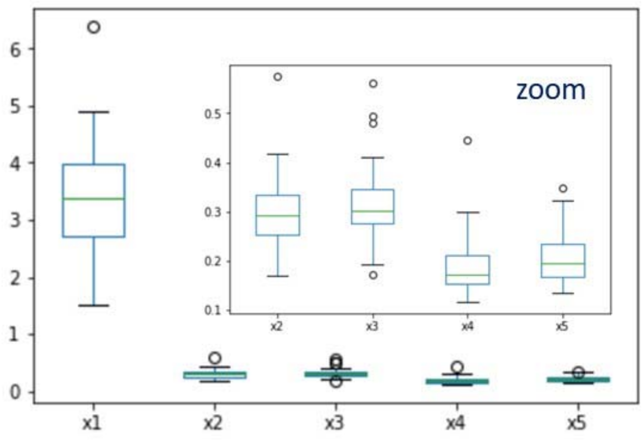


Fig. 6. *Boxplot computed with 30 estimated values of* $\delta_1,\ldots,\delta_5$.

In Figure 7, we compare the convergence of the standard PSO algorithm and the modified PSO algorithm with (3). We obtain the expected result. The introduction of the indicators δ_j after the 25th iteration accelerates the convergence.

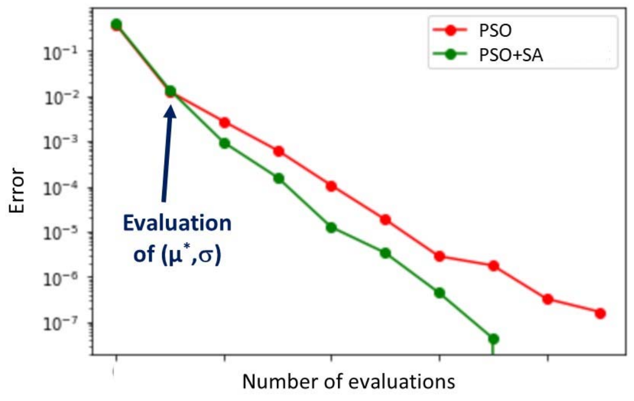


Fig. 7. *Convergence of the PSO algorithm (red) and modified PSO algorithm (green)*

The method is efficient if some input variables have a linear impact on the objective function. In the toy example, if we remove x_1, we only have two variables with nonlinear effect and two negligible effects. It can be seen in Figure 8 that the convergence curves are comparable with or without the introduction of the sensitivity indicator. There is a little benefice for the modified PSO, certainly due to the fact that the shift length is reduce for the non linear variables.

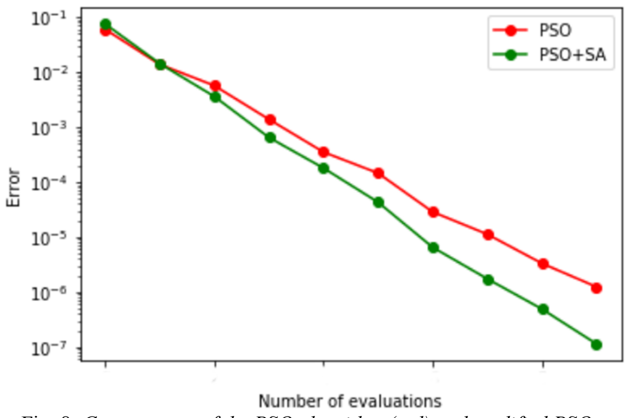


Fig. 8. *Convergence of the PSO algorithm (red) and modified PSO algorithm (green) without linear effect variable*

In this paper, the indicator δ_j for the jth variable has been used with PSO algorithm, as an illustration. But the indicator can be used in the same way with other metaheuristics.

As an example, in the Artificial Bee Colony metaheuristic [4], for a candidate solution x_i, a random offset is made along only one dimension j, towards another chosen candidate $x_k, j \neq k$.
The modified shift equation, is given by (4).

$$y_{ij} = x_{ij} + \delta_j \Phi_{ij}(x_{ij} - x_{kj}) \qquad (4)$$

Differential Evolution [11] shifts candidate solutions' variables in the mutation operation. The original mutation equation is given by (5). This mutation generates a mutant individual using three distinct individuals, x_{r1}, x_{r2}, x_{r3} and a constant mutation factor F.

$$v_j^{t+1} = x_{r1j}^t + F\ (x_{r2j}^t - x_{r3j}^t),\ \forall j \in \{1,\ldots,D\} \qquad (5)$$

In that case, the indicator δ_j replaces the constant factor F, for the jth variable. Each variable has a constant value, according to its effect on the objective function. There are many mutation variants, our indicator can be used in the same way.

V. Conclusion and perspectives

Following our previous work, we explored the possibilities to use information gathered during the optimization process to accelerate the convergence.

Our sensitivity analysis method is used to differentiate between variables with linear and nonlinear effects. We defined an indicator based on this distinction.

This indicator makes it possible to adjust the shift length according to the nature of the variable. Thanks to a very simple example we have illustrated the interest of this approach. Now, we need to test this method on a large number of benchmarks. The key point is to modify the usual objectives functions [7] in order to have linear and nonlinear variables.

As of now, the indicators δ_j are computed once (after 25 iterations). It might be interesting to re-evaluate them during the algorithm. Indeed, after several iterations, the algorithm focuses its researches on a small zone. In such a restricted zone, the behavior of the variables may change.

References

[1] Chelouah, R., Siarry, P. (2000). Tabu search applied to global optimization. *European Journal of Operational Research* 123(2), 256–270.

[2] Iooss, B., Lemaître, P. (2015). Uncertainty management in simulation-optimization of complex systems: Algorithms and applications. In *A Review on Global Sensitivity Analysis Methods*, Springer US, Boston, MA.

[3] Jourdan, A., Franco, J. (2010). Optimal Latin hypercube designs for the Kullback–Leibler criterion. *AStA Advances in Statistical Analysis*, 94(4), 341–351.

[4] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University.

[5] Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N. (2012). A comprehensive survey: Artificial Bee Colony (ABC) algorithm and applications. *Artificial Intelligence Review,* 42(1), 21–57.

[6] Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks,4 , 1942–1948.

[7] Liang, JJ., Qu BY, Suganthan PN, Hernández-Díaz AG (2013). Problem definitions and evaluation criteriafor the CEC 2013 special session on real-parameter optimization. Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore.

[8] Morris, M.D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics,* 33(2), 161–174.

[9] Loubière P., Jourdan A., Siarry P., Chelouah R. (2016). A smart exploration of the search space for the Artificial Bee Colony algorithm. *Applied Soft Computing*, 41, 515-531.

[10] Loubière P., Jourdan A., Siarry P., Chelouah R. (2017). A sensitivity analysis method aimed at enhancing the metaheuristics for continuous optimization, *Artificial Intelligence Review*, 50, 625-647.

[11] Price, K., Storn, R.M., Lampinen, J.A. (2005). *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York.

[12] Rana S, Jasola S, Kumar R (2011) A review on particle swarm optimization algorithms and their applications to data clustering. *Artificial Intelligence Review,* 35(3), 211–222

[13] Saltelli A (2002) Sensitivity analysis for importance assessment. *Risk Anal*ysis, 22(3), 579–590.