Is Global Sensitivity Analysis Useful to Evolutionary Computation? *

Thomas Chabin, Alberto Tonda, Evelyne Lutton
UMR 782 GMPA, INRA
1 Av. Lucien Brétignères
78850 Thiverval-Grignon, FRANCE
{thomas.chabin,alberto.tonda,evelyne.lutton}@grignon.inra.it

ABSTRACT

Global Sensitivity Analysis (GSA) studies how uncertainty in the inputs of a system influences uncertainty in its outputs. GSA is extensively used by experts to gather information about the behavior of models, represented as a function receiving some inputs or parameters and delivering one or several outputs, through computationally-intensive stochastic sampling of a parameters space. Some studies propose to make use of the considerable quantity of data acquired in this way to optimize the model parameters, often resorting to Evolutionary Algorithms (EAs): intuitively, a probabilistic analysis can prove useful to a stochastic optimization technique. Nevertheless, efficiently exploiting information gathered from GSA might not be so straightforward. In this paper, we present two counterexamples to prove how naively combining GSA and EA can bring about negative outcomes. Experimental results substantiate our claim, and suggest that GSA information should be thoroughly examined before deciding how to tackle the optimization of a target model.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

Keywords

Sensitivity analysis, global sensitivity analysis, evolutionary computation, EASEA, cma-es, real-valued optimization

1. INTRODUCTION

Sensitivity analysis (SA) is the study of how the uncertainty in the output of a mathematical function can be apportioned to different sources of uncertainty in its inputs [16]. In general, SA can be applied to any function f such as:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^p \tag{1}$$

In practice, this technique is widely exploited by the modeling community, to analyze the behavior of models with respect to their parameters, and to later plan new experiments to reduce the uncertainty on the most sensitive parameters. Indeed, a model can be defined as a function $f : X_l, P_n \rightarrow Y_m$, whose objective is to simulate a real physical phenomena. Knowing the initial conditions represented by the vector X_l, the model produces the final conditions of the studied phenomena, Y_m. In real-world cases, the parameters of the function P_n are not known with precision but rather defined by an range value of uncertainty. Many SA tools perform a stochastic sampling of considerable magnitude in the parameters' space, and then exploit statistical techniques to obtain information from this large quantity of data.

It is easy to see the potential utility of data collected through SA for an optimization of the model's parameters: not only SA provides a fine-grained sampling of a search space, but it also conveys useful information about how each parameter influences each output. This holds true especially for evolutionary optimization techniques, that are based on a biased stochastic sampling of the search space. Re-utilizing the extensive amount of computation performed for a SA to improve the performance of an evolutionary algorithm (EA) targeting the same search space, sounds not only sensible, but also extremely appealing. Not surprisingly, literature already shows approaches that exploit the synergy between SA and EAs [4]. However, making use of the information conveyed by SA might not be as straightforward as it seems.

In this paper, we propose two case studies, specifically designed to deceive an EA exploiting SA data to optimize their values. Experimental results show that even a state-of-the-art EA is unable to find the optimal parameter configuration for the problems, if biased by the information provided by SA; on the contrary, the same algorithm routinely converges on the global optimum if no aprioristic knowledge is given, thus proving that a naive use of SA information might actually be harmful to the optimization process.

The rest of the paper is organized as follows: section 2 briefly introduces the basic concepts of SA, and shows how other attempts have been made to combine this technique with evolutionary computation. Section 3 discusses one of these combination strategies. Two counterexamples and the related experimental results are illustrated in section 4, while

*This work has been funded by the French National Agency for research (ANR), under the grant ANR-11-EMMA-0017, EASEA-Cloud Emergence project 2011, `http://www.agence-nationale-recherche.fr/`

GECCO'15, July 11-15, 2015, Madrid, Spain.

the implications are discussed in section 5. Finally, section 6 concludes the paper.

2. BACKGROUND

In the following, a few basic concepts of SA are recalled, with a particular focus on the analysis of joint variation of parameter interactions; subsequently, previous works at the interface between SA and EAs are listed.

2.1 Sensitivity Analysis: Global and Local

SA is a technique used to understand how variation in the output of a function can be apportioned qualitatively or quantitatively to different uncertain input sources. SA techniques can be broadly classified as *local* or *global*. Local sensitivity analysis (LSA) is the simpler approach, where only one function variable is perturbed at a time, while the remaining are fixed to a nominal value. Different studies have shown that limiting the analysis to local sensitivities might deliver unreliable results [17] [20]. Thus, global sensitivity analysis (GSA) [16] that examines the joint variation of variable interactions, seems to be better suited for complex, nonlinear models.

2.2 Global Sensitivity Analysis

One of the most common approaches to GSA has been developed by Sobol [18], to provide the impact of each individual decision variable and its interactions with other variables on performance objectives, using sensitivity indices. GSA is mainly used for two goals: *factor prioritizing*, deciding which variable uncertainty to work on, in order to reduce the uncertainty of the output the most; and *factor fixing*, highlighting which variables can be fixed to an arbitrary value without influencing much the output.

Each of the following types of indices takes values between 0 and 1, and represents a proportion of influence.

First-order sensitivity indices are used for the factor priority problem. A first-order index is associated to each parameter, representing the direct influence of the uncertainty of that parameter onto the variance of an output. A first-order index is defined by the following formula:

$$S_i = \frac{V[E(Y|X_i)]}{V(Y)}$$

where X_i is the input i on which the index is calculated, Y is the output, and S_i is the first-order index on i. The idea is to calculate the variance of the conditional expectation of Y knowing X_i, fixed at every possible value within the uncertainty range of X_i, divided by the variance of Y. Fixing to its true value the variable associated to the highest first-order index, would lead to the greatest reduction in the variance of the output.

Higher-orders sensitivity indices can also be computed. For example, indices of order 3 can be attributed to every triplet of parameters: such indices represent how much a triplet is directly responsible for the variance in the output, and are computed with the following formula:

$$S_{ijk} = \frac{V[E(Y|X_i, X_j, X_k)]}{V(Y)}$$

where X_i, X_j, X_k are the parameters on which index S_{ijk} is calculated, while Y is the output. The sum of every n-order index is always equal to 1. It must be noted that the computation of higher-order indices is expensive, as there are $\binom{n}{k}$ of such indices for k parameters, and they are seldom used in practice. As such, they will not be considered in this paper.

Total-effect sensitivity indices are used for the factor-fixing problem. A total-effect index is attributed to each parameter, and is interpreted as the sum of all n-order indices involving the considered parameter. A total effect index S_{Ti} represents how much the uncertainty of a parameter, combined with every other uncertainty, is responsible for the output variance. It is defined by the following formula :

$$S_{Ti} = 1 - \frac{V[E(Y|X_{\sim i})]}{V(Y)}$$

where $X_{\sim i} = X_1, X_2, ..., X_{i-1}, X_{i+1}, ...X_n$, the set of all parameters except X_i, Y the output. Therefore, if a parameter has a total effects index near zero, the uncertainty on this parameter has nearly no influence on the output variance. For this reason, this parameter can be fixed to an arbitrary value inside his interval of uncertainty without affecting much the variance of the output.

2.3 Sensitivity Analysis and Optimization

On one hand, SA produces a considerable amount of information, that could be interpreted as a sampling of a search space of parameters. On the other hand, since SA is aiming at finding the parameters whose variation influences the output of a function (or a model) the most, the entire process can be seen as an optimization problem. It is therefore not surprising that several attempts have been performed to combine SA with optimization tools, especially those featuring a stochastic sampling of the search space.

A considerable number of research lines exploit LSA to perform what is termed *robust optimization* [2], a set of techniques which seek a certain amount of robustness against uncertainty, seen as variability in the value of the parameters of the problem or its solution. Some work, like [1] also propose a multi-objective strategy to assess the identifiability and LSA of the parameters of a system.

In [19], EAs are used to find the worst possible parameter settings for a model, maximizing the distance between experimental data and model predictions. The results are then exploited to evaluate the influence of each parameter on the outputs. While surely interesting, this approach lacks the statistical support of global SA, providing the user with a general impression of the most influential parameters.

Another research line, presented in two technical reports [14] [13], aims at using the points sampled by a CMA-ES algorithm[8] during the optimization process as the basis for a SA, through a *de-biasing* of the sampling. In practice, weights are used on the sampling points, on the basis of the covariance matrix' determinant at each generation, to express their bias with respect to a completely random process. While surely interesting, this methodology raises several theoretical questions that will need to be thoroughly analyzed before its widespread application.

In [4], the authors use GSA measurements to reduce the problem's dimensionality, first optimizing the values of a sub-set of the most sensitive parameters, and then restarting the evolution from the solutions found in this way, finally optimizing the remaining values.

3. ADAPTIVE DIMENSIONALITY REDUCTION BASED ON GSA

The idea of using progressive refinements techniques to perform a search in large dimension spaces appeared as attractive since a long time. This very simple idea is at the origin, for instance, of the *messy genetic algorithm* scheme proposed by Goldberg 25 years ago [5] : "Nature did not start with strings of length two million (an estimate of the number of genes in Homo sapiens) and try to make man. Instead, simple life forms gave way to more complex life forms, with the building blocks learned at earlier times used and reused to good effect along the way." Messy GAs rely on a variable length bit-string representation of the search space made of a list of couples (locus, allele value) specifying the value of a bit at a given place of the genome. In this way some genes may be over-specified (several possible values) while other may be under-specified (no affected value). Fitness calculation is then performed after an additionnal stage relying on various rules for inferring uncomplete string values. This scheme has been extended in various ways including continuous search spaces [15, 9]. It implements a self-adaptive progressive refinement, where the selection of primary, "heavy" parameters, is let to evolution.

Adaptive schemes (in the sense of "non-self-adaptive") may also be considered in this context, the critical point being an *a priori* knowledge of an importance prioritization of the parameters. Sensitivity analysis may then represent an attractive solution to deal with parameters importance ordering. The idea is to identify non-influent parameters, via a SA of the fitness function with respect to each variable of the search space. A straightforward strategy for dimensionality reduction is then to ignore non-influent parameters in a first optimization stage, like in [4].

4. EXPERIMENTAL ANALYSIS

We have built two counterexamples for testing the limits of dimensionality reduction based on GSA. They have been designed in the same spirit as deceptive functions [7, 6]: global informations that may be collected through a statistical analysis of some features (building blocks statistics in the case of deceptiveness "a la Goldberg") yields puzzling information to the algorithm. Other interpretations may also stem from theoretical studies regarding the influence of local regularity features [10, 12]: global optima are located in very irregular areas, while attractive local optima are located inside smooth areas. Statistical features are not able to capture local irregularities and are thus yielding erroneous information to the algorithm [11].

4.1 Isolating a relevant subset of parameters

The tested optimization strategy rely on the following statement (factor fixing approach, see section 2.2): a low total effect index indicates a non-influent parameter that can be arbitrarily fixed with only few impact on the fitness function.

To decide which parameters are non-influential, a threshold is arbitrarily fixed (a low value in the range $[0, 1]$): parameters that have a total sensitivity index below this threshold are considered non-influential.

4.2 Algorithms

Two strategies have been considered :

- *Approach 1* performs an optimization of the influential parameters only. Non-influential parameters are fixed to the middle of their interval of uncertainty.
- *Approach 2* is based on [4]. Non-influential parameters are optimized in a first stage, like in Approach 1, whose best point is injected in the initial population of a second optimization using all parameters.

Two Evolution Strategies have been tested, the CMA-ES and an explicit population based EA, programmed thanks to the EASEA package[1] [3].

4.2.1 CMA-ES

The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [8] is a popular EA, widely used for many real-world optimization problems. It is known for his robustness and computational efficiency. For Approach 2, CMA-ES is restarted as follows:

- The mean point is initialised to the the best set of influential parameters found during the first stage, while the values of non-influential parameters are set to the middle of their interval of uncertainty.
- The standard deviation for each influential parameter is kept to the value obtained at the last generation of the first stage, and the standard deviation for non-influential parameters is set to $0.3 \times (range_{max} - range_{min})$.

4.2.2 EA

The second algorithm used in our test is a classical EA programmed in EASEA [3]. For Approach 2, the initial population of the second stage is seeded with the content of the last generation of the first stage. The non-influential parameters who were fixed at the middle of their interval of uncertainty (or search space) are attributed a random value in their range of uncertainty.

4.3 Counterexample I

For the first counterexample, the idea is to build a function for which a non-influential parameter remains important for the precise location of a global optimum. This can be achieved with functions having simultaneously waves along some axes (corresponding to influential "shapes") and thin peaks along other axes. The result is that the projection of the fitness function on the subspace of non-influential parameters provides an averaged viewpoint on the fitness landscape that erases high, thin peaks. This behaviour is achieved by the following bi-dimensional function (Figure 1):

$$fit_1(k1, k2) = g(k1, 1.33, -1, 0.18) + g(k2, 7.98, 1, 0.0005) + h(k1)$$

where g is a Gaussian:

$$g(k, a, b, c) = a \times exp(-\frac{(k-b)^2}{2c^2})$$

and $k1, k2 \in \big[-1; 1\big]$.

To facilitate optimization along parameter $k1$, a small gradient, $h(k1)$ is added to the fitness: from $k1 = -1$ to $k1 = 0.0005$, $h(k1)$ goes from 0 to 0.1, and from $k1 = 0.0005$ to $k1 = 1$, $h(k1)$ goes from 0.1 to 0.

[1] `http://easea.unistra.fr`

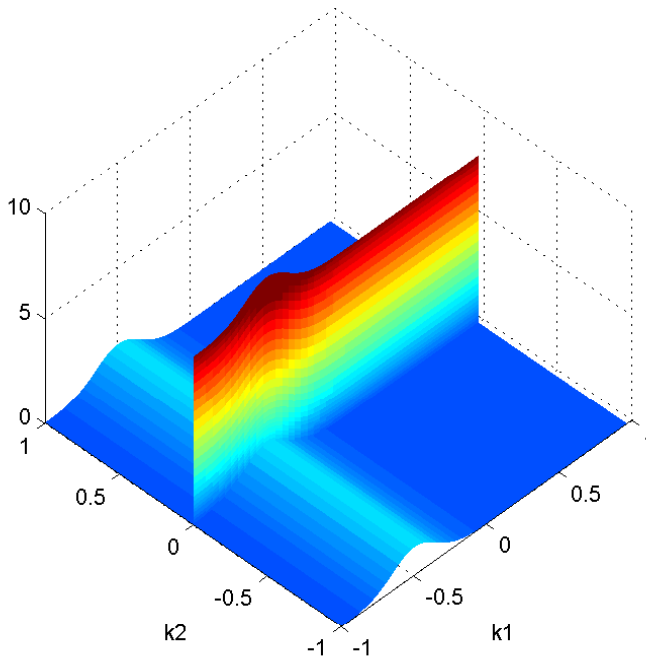

Figure 1: Counterexample I. The peak is positioned in correspondence of $k2 = 0.0005$. Line $k2 = 0$ is actually at the bottom of the peak.

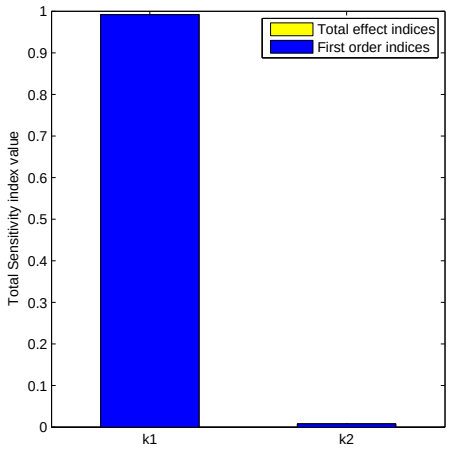

Figure 2: Sensitivity analysis for Counterexample I. Parameter $k1$ is shown to be much more influential than parameter $k2$.

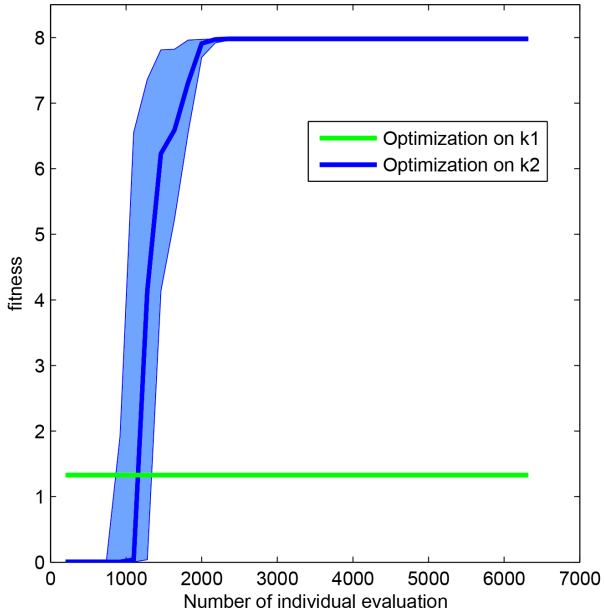

Figure 3: Comparison of optimization runs on $k1$ and $k2$, respectively, for Counterexample I using a classical EA. The statistics report an average on 100 runs, the median is displayed in bold and first- and third- quartile with thinner lines of the same color.

A global sensitivity analysis, whose results are presented in Fig. 2 reads that $k1$ is influent whereas $k2$ is not, since the total effect index of $k2$ is far lower that the total effect index of $k1$.

In a first test, the function is optimized on parameter $k1$ alone, and the result is compared to an optimization on parameter $k2$ only. Since $k1$ seems to bear all influence whereas $k2$ appears to be non-influential, it is naively expected that the optimization on $k1$ will find a better value than the optimization on $k2$. Using the EA with parameters reported in 1, we obtain the results reported in Figure 3 (statistics on 100 runs). In this case, optimizing on the non-influential parameter is unexpectedly a better option than optimizing on the supposedly most influential parameter.

Population size	$\mu = 200$
Offsprings size	$\lambda = 180$
Number of generations	35
Tournament selection	$Size = 2$
BLX-α Crossover	$p = 1.$
Log normal self adaptive mutation	$p = 1.$ $\tau = \sqrt{2}$
Number of Runs	100

Table 1: EA parameter setting for Counterexample I

The same conclusions are reached using the CMA-ES with the settings reported in Table 2. Statistics for 100 runs are displayed in Figure 4.

4.4 Counterexample II

The behaviour shown in Counterexample I may be counterbalanced by a restart strategy, such as Approach 2 de-

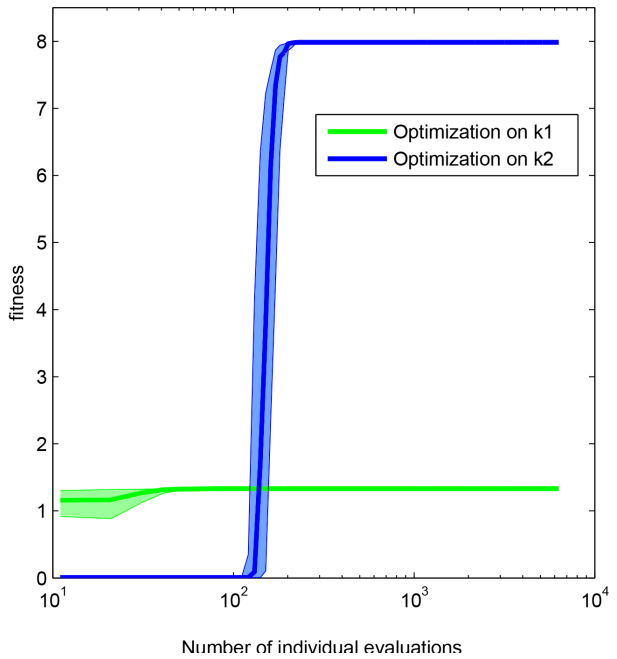

Figure 4: Comparison of optimizations on $k1$ and $k2$, respectively, for Counterexample I using CMA-ES.

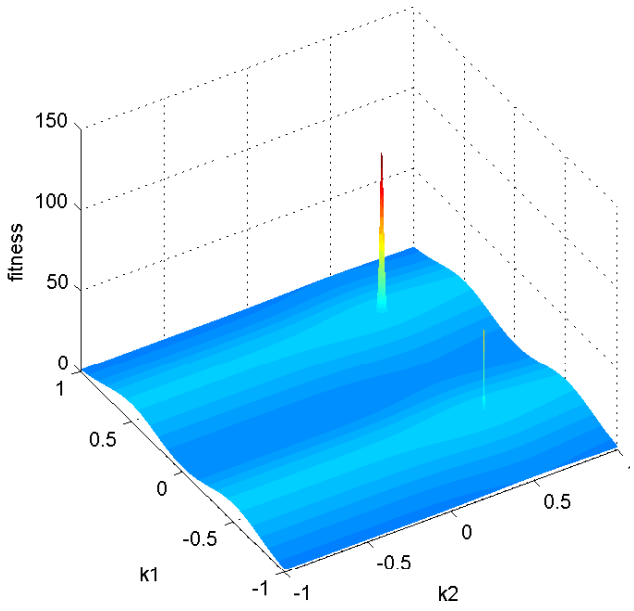

Figure 5: Counterexample II. There are two thin peaks, a very thin one corresponding to a local optimum is located at $(-0.5, 0.5)$ and a larger one, global optimum at $(0.5, 0.5)$.

Population size	10
Number of generations	632
Number of Runs	100

Table 2: Parameter settings for the CMA-ES in Counterexample I.

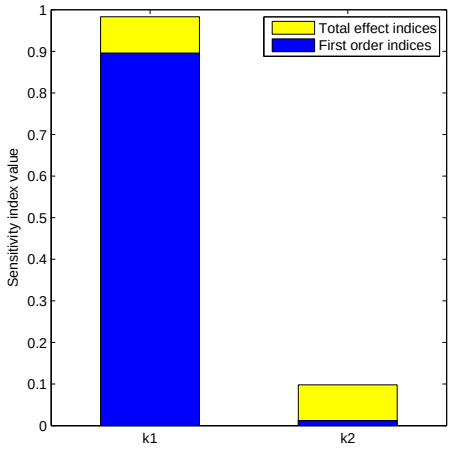

Figure 6: Sensitivity analysis of Counterexample II. It is evident how the total effect index for $k1$ is much higher than the same index computed for $k2$.

scribed in section 4.2. With Counterexample II, we will see that the optimization process using GSA information may still be deceived, even using restarts. The function to optimize in Counterexample II is defined as follows:

$$fit_2(k1,k2) = g(k1,10.9,0.5,0.25) + g(k1,11,-0.5,0.25) + g(k2,1,0.5,0.25) + g2d(k1,k2,100,0.5,0.01,0.5,0.01) + g2d(k1,k2,50,-0.5,0.0025,0.5,0.0025)$$

where

- g is a Gaussian:

$$g(k,a,b,c) = a \times exp(-\frac{(k-b)^2}{2c^2})$$

- $g2d$ is a 2D Gaussian:

$$g2d(k1,k2,a,b,c,d,e) = a \times exp(-(\frac{(k1-b)^2}{2\times c^2} + \frac{(k2-d)^2}{2\times e^2}))$$

- $k1, k2 \in [-1;1]$

This fitness function is displayed in Fig. 5, where 2 optima are visible: a local optimum, located at $(k1 = -0.5; k2 = 0.5)$, and a global optimum, located at $(k1 = 0.5; k2 = 0.5)$. A GSA on Counterexample II (See Figure 6), shows that $k1$ can be considered as an influential parameter and $k2$ as a non-influential one.

A progressive refinement (Approach 2) is compared to a plain optimization (full search space) using a classical EA, with the parameter settings reported in Table 3. Over

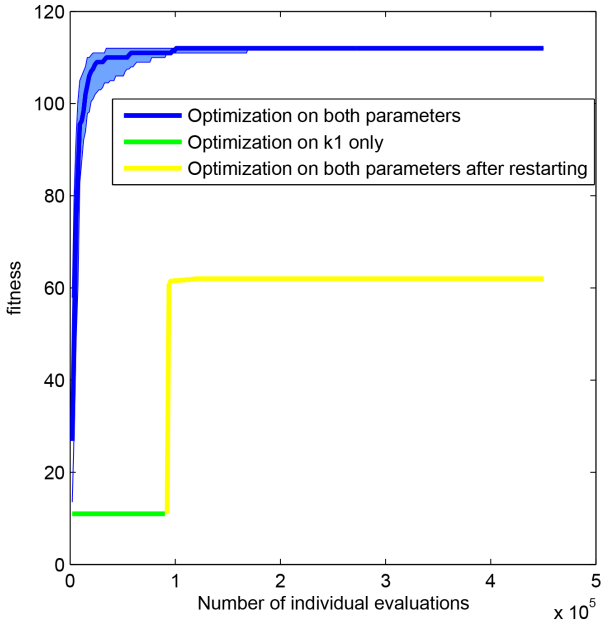

Figure 7: Statistics of 100 runs on Counterexample II with a classical EA.

100 runs, the full search always finds the global optimum whereas the restart strategy Approach 2 always get stuck on the local optimum (Figure 7).

Population size	$\mu = 2000$
Offsprings size	$\lambda = 1800$
Number of generations	full search : 250 Approach 2 : 50 then 200
Tournament selection	$Size = 2$
BLX-α Crossover	$p = 1.$
Log normal self adaptive mutation	$p = 1.$ $\tau = \sqrt{2}$
Number of Runs	100

Table 3: EA parameter setting for the optimization, full search space and Approach 2, for Counterexample II.

The first-stage optimization concentrates the population around the line $k1 = -0.5$, which prevents the second stage from finding the global peak positioned at $k1 = 0.5$.

The same set of experiments has been performed using the CMA-ES with two parameter settings: a first one letting the CMA-ES self-tune its population size, the second one using a larger population size with the idea of artificially maintaining diversity. See Table 4 for the details.

Figures 8 and 9 display the statistics over 100 runs for the CMA-ES. It is noticeable that, using a population of 250 individuals, the strategy is trapped on the local optimum even when using a search on all parameters. When the CMA-ES self-tunes its population size, and thus runs using a very small number of individuals, this effect is still very frequent for the full search. In both cases, it is again obvious that Approach 2 provides deceiving information to the algorithm, and delays or even prevents it from converging.

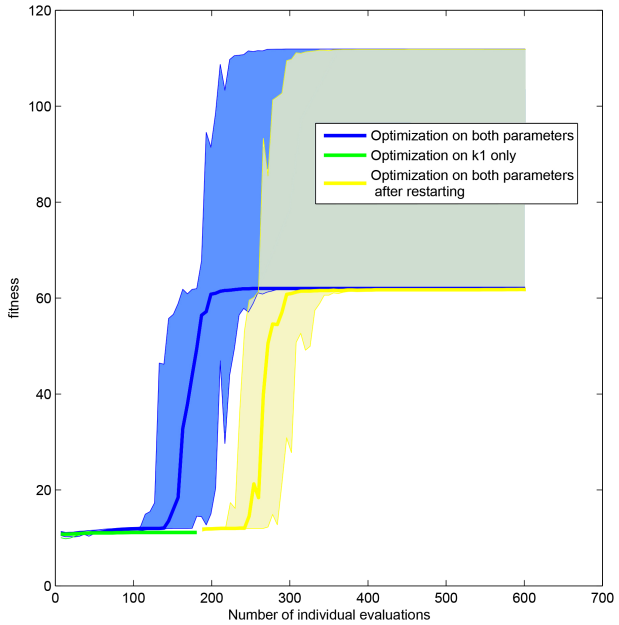

Figure 8: Statistics of 100 runs on Counterexample II with CMA-ES and a population size of 6. The grey area corresponds to the superposition of the blue and the yellow areas. This illustrates the fact that CMA-ES is able to sometimes find the global optimum using Approach 2 also. In the most frequent scenario however, both algorithms fall into the local optimum.

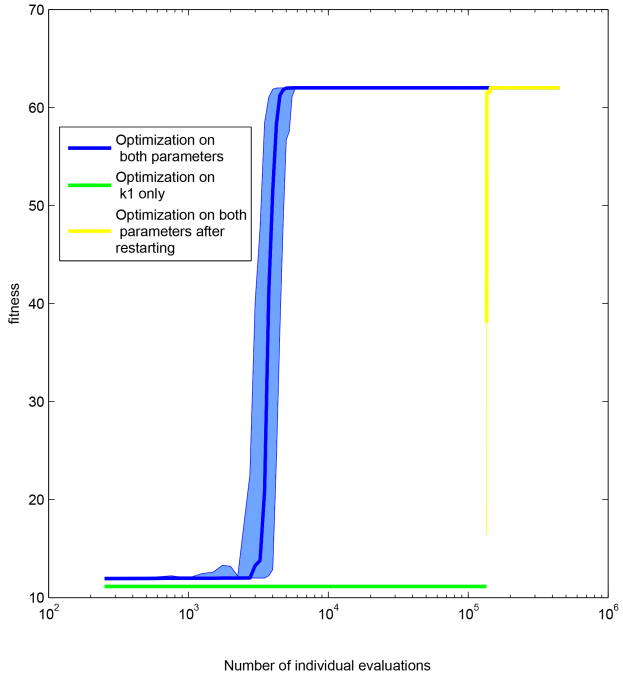

Figure 9: Statistics of 100 runs on Counterexample II with CMA-ES and a population size of 250

Population size	6	250
Number of generations	75033	1800
Number of Runs	100	100

Table 4: Parameter setting for the two runs of CMA-ES on Counterexample II. Number of generations are tuned to allocate the same number of computations for each experiment.

5. DISCUSSION

The two previous counterexamples shed light on the fact that sensitivity analysis may deliver misleading information to the optimization process. A possible explanation is that GSA is based on a statistical analysis over a given parameter range. In this way it provides an averaged viewpoint on each parameter, and it is clear that averaging may hide many fine details that are important for optimization purposes. Another problem is due to the fact that the results of a GSA may drastically vary with the choice of the parameter range. It often happens that a parameter is influent on some subspace and not on another. Fig 10 illustrates this effect for Counterexample I: when $k1, k2 \in \left[-1; 1\right]$, $k1$ is the parameter that has almost all the influence, whereas $k2$ is almost non-influential. But on other areas, results can be the opposite: for instance if $k1, k2 \in \left[-0.1; 0.1\right]$, $k1$ is regarded as non-influential, while $k2$ becomes predominant.

The question of an efficient use of GSA inside an optimization procedure is raised: GSA is, in itself, extremely time consuming, and this cost has not been taken into account in the previous experiments. It seems obvious that GSA, based on a stochastic sampling of the full search space or of an area of it, consumes a computation time that may sometimes better be spent to perform the search itself. Additionally, using the averaged information provided by GSA may hidden some interesting irregular areas where optima may be found. Finally, adaptive refinement methods like approach 2 proposed in this paper, or the one proposed in [4], need to identify a non-negligible subset of non-influential parameters, which is not always the case, especially for complex optimization problems. More progressive strategies may be imagined, but once again with the risk related to an assessment of the relative importance of parameters averaged over a given area.

6. CONCLUSIONS

GSA is a technique able to deliver information on how the uncertainty in the inputs of a system might influence uncertainty in its outputs. Since this data is acquired through a stochastic sampling of the search space, several research lines exploited the intuitive synergy between GSA and EAs, using the information to reduce the dimensionality of the search space, or to choose the variables on which to optimize first.

In this paper, we presented two case studies, specifically designed to provide deceiving information to sensitivity analysis used during an optimization process. As a result, stochastic optimization biased by this information has been experimentally proven unable to reach the global optimum on both problems. A simple progressive refinement optimization scheme based on parameter prioritization such as in [4] may work on some functions, but there is a risk of falling into a local optimum, from which escaping might prove to be hard. Parameter prioritization might work better for multi-objective problems, thanks to a better diversity preservation mechanism necessary for a correct sampling of Pareto Fronts.

LSA remains interesting, and for instance local Sobol indices may be useful for tuning mutations, in the same spirit as what has been developed in [11], but with an associated computational cost that should always be taken into account.

7. REFERENCES

[1] Barichard, V., Hao, J.K.: Resolution d'un probleme d'analyse de sensibilite par un algorithme d'optimisation multiobjectif. In: 5eme conference francophone de Modelisation et SIMulation (MOSIM 2004), Nantes. pp. 59–66 (2004)

[2] Beyer, H.G., Sendhoff, B.: Robust optimization–a comprehensive survey. Computer methods in applied mechanics and engineering 196(33), 3190–3218 (2007)

[3] Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it easea. In: Parallel Problem Solving from Nature PPSN VI. pp. 891–901. Springer (2000)

[4] Fu, G., Kapelan, Z., Reed, P.: Reducing the complexity of multiobjective water distribution system optimization through global sensitivity analysis. Journal of Water Resources Planning and Management 138(3), 196–207 (2011)

[5] Goldberg, D., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3(5), 493–530 (1989)

[6] Goldberg, D.: Genetic algorithms and walsh fuctions: II. Deception and its analysis. Complex Systems 3(2), 153–171 (April 1989)

[7] Goldberg, D.: Genetic algorithms and walsh functions: I. A gentle introduction. Complex Systems 3(2), 129–152 (April 1989)

[8] Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)

[9] Kargupta, H.: The gene expression messy genetic algorithm. In: International Conference on Evolutionary Computation. pp. 814–819 (1996)

[10] Leblanc, B., Lutton, E.: Bitwise regularity and ga-hardness. In: ICEC 98, May 5-9, Anchorage, Alaska (1998)

[11] Lutton, E., Lévy Véhel, J.: Pointwise regularity of fitness landscapes and the performance of a simple es. In: CEC'06. Vancouver, Canada (July, 16-21 2006)

[12] Lutton, E., Véhel, J.L.: Hölder functions and deception of genetic algorithms. IEEE transactions on Evolutionary computation 2(2), 56–72 (July 1998)

[13] Müller, C., Paul, G., Sbalzarini, I.: Sensitivities for free: Cma-es based sensitivity analysis. Tech. rep., ETH Zurich (2011)

[14] Paul, G., Müller, C., Sbalzarini, I.: Sensitivity analysis from evolutionary algorithm search paths. Tech. rep., ETH Zurich (2011)

[15] Rajeev, S., Krishnamoorthy, C.: Genetic algorithms-based methodologies for design optimization of trusses. Journal of Structural Engineering 123(3), 350–358 (1997)

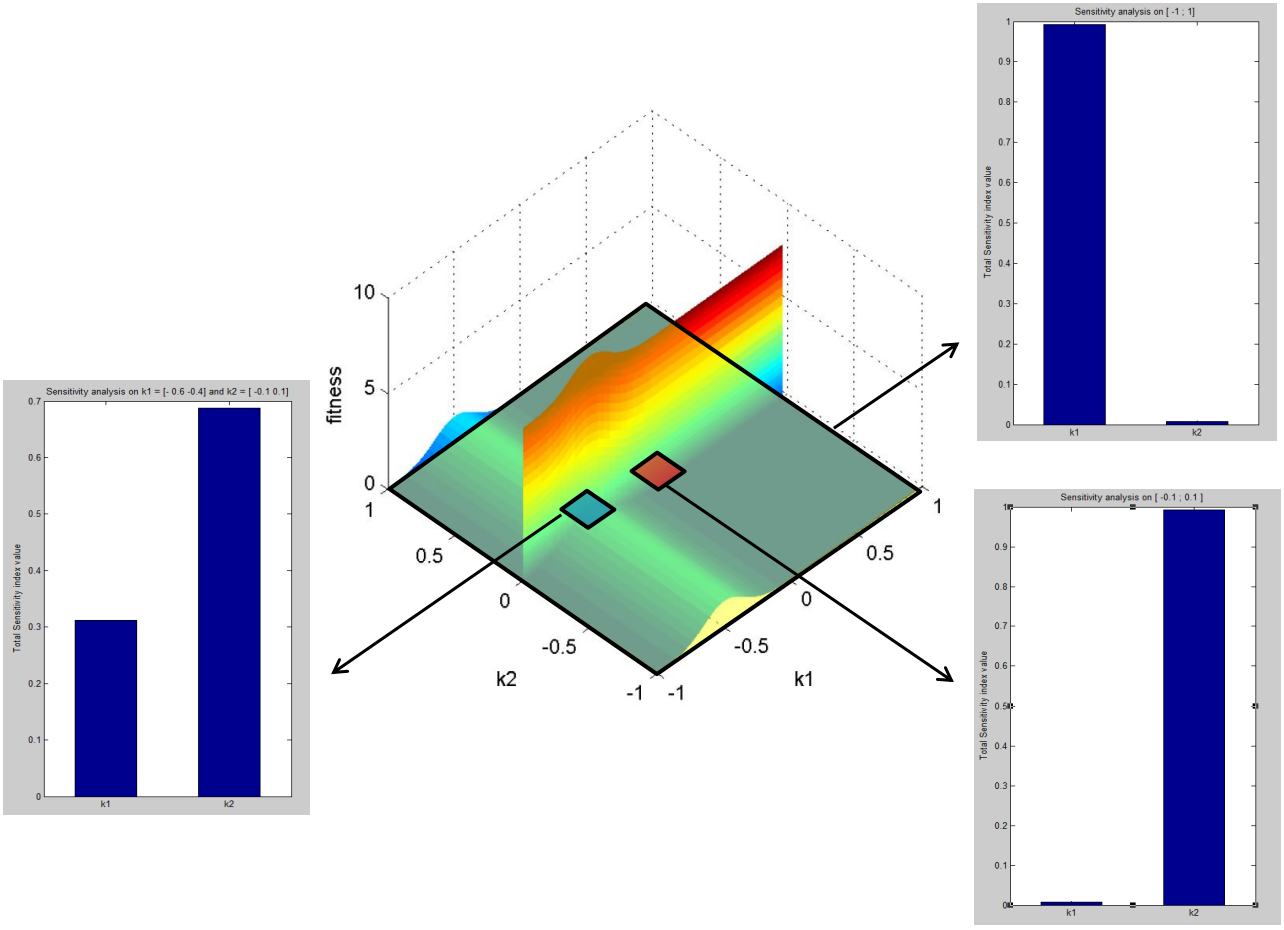

Figure 10: Various sensitivity analyses on three sub-spaces for Counterexample I: parameters influences vary a lot !

[16] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity analysis, The Primer. John Wiley & Sons (2008)

[17] Saltelli, A., Annoni, P.: How to avoid a perfunctory sensitivity analysis. Environmental Modelling & Software 25(12), 1508–1517 (2010)

[18] Sobol, I.M.: Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. Mathematics and computers in simulation 55(1-3), 271–280 (2001)

[19] Stonedahl, F., Wilensky, U.: Evolutionary robustness checking in the artificial anasazi model. In: AAAI Fall Symposium: Complex Adaptive Systems (2010)

[20] Tang, Y., Reed, P., Wagener, T., Van Werkhoven, K., et al.: Comparing sensitivity analysis methods to advance lumped watershed model identification and evaluation. Hydrology and Earth System Sciences Discussions 11(2), 793–817 (2007)