How to Mislead an Evolutionary Algorithm using Global Sensitivity Analysis *

Thomas Chabin[1], Alberto Tonda[1], and Evelyne Lutton[1]

UMR 782 GMPA, INRA
1 Av. Lucien Brétignères, 78850 Thiverval-Grignon, FRANCE
thomas.chabin,alberto.tonda,evelyne.lutton@grignon.inra.fr

Abstract. The idea of exploiting Global Sensitivity Analysis (GSA) to make Evolutionary Algorithms more effective seems very attractive: intuitively, a probabilistic analysis can prove useful to a stochastic optimisation technique. GSA, that gathers information about the behaviour of functions receiving some inputs and delivering one or several outputs, is based on computationally-intensive stochastic sampling of a parameter space. Nevertheless, efficiently exploiting information gathered from GSA might not be so straightforward. In this paper, we present three mono- and multi-objective counterexamples to prove how naively combining GSA and EA may mislead an optimisation process.

1 Introduction

Sensitivity analysis 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 [19]. In general, Sensitivity Analysis can be applied to any function f, $\mathbb{R}^n \rightarrow \mathbb{R}^p$. In practice, this technique is widely exploited by the modeling community, to analyze the behaviour 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 $f : X_l, K_n \rightarrow Y_m$, 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 K_n are not known with precision but rather defined by a range value of uncertainty. Many sensitivity analysis tools perform a stochastic sampling of considerable magnitude in the space of parameters, and then exploit statistical techniques to derive information from this large quantity of data.

It is easy to see the potential interest of data collected through sensitivity analysis for an optimisation of the parameters of the model: not only sensitivity analysis provides a fine-grained sampling of a search space, but it also conveys useful information about how each parameter influences each output. This holds

* 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/

true especially for evolutionary optimisation techniques, that are based on a biased stochastic sampling of the search space. Re-using the extensive amount of computation performed for a sensitivity analysis to improve the performance of an evolutionary algorithm (EA) targeting the same search space, sounds not only sensible, but also extremely appealing. Not surprisingly, the literature already shows approaches that exploit the synergy between sensitivity analysis and EAs [7]. However, making use of the information conveyed by sensitivity analysis might not be as straightforward as it seems.

In this paper, we exhibit three case studies, specifically designed to deceive an EA exploiting sensitivity analysis data. 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 sensitivity analysis; 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 sensitivity analysis information might actually be harmful to the optimisation process.

The rest of the paper is organized as follows: Section 2 recalls a few basic concepts of sensitivity analysis, with a particular focus on the analysis of joint variation of parameter interactions, and lists previous works at the interface of sensitivity analysis and EAs. Section 3 discusses one of these combination strategies. Counterexamples and experimental results are illustrated in Section 4, while the implications are discussed in Section 5. Finally, Section 6 concludes the paper.

2 Background

2.1 Sensitivity analysis: Global and Local

Sensitivity analysis 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. Sensitivity analysis 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 [20, 23]. Thus, global sensitivity analysis (GSA) [19] that examines the joint variation of variable interactions, seems to be better suited for complex, nonlinear models.

2.2 Global Sensitivity Analysis

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; and *factor fixing*, highlighting which variables can be fixed to an arbitrary value with few influence on the output. One of the most common approaches has been developed by Sobol [21]. Impacts of each individual decision variable and its interactions with other variables on performance objectives are represented with the following sensitivity indices, taking values in $[0, 1]$.

First-order sensitivity indices are used for the factor priority problem. A first-order index S_i is associated to each parameter K_i, and represents the direct influence of its uncertainty on an output Y:

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

It corresponds to the part of the variance of Y explained directly by the uncertainty in K_i: $V[E(Y|K_i)]$ is the conditional expectation of Y knowing K_i, fixed at each possible value within the uncertainty range of K_i. 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 correspond to interaction effects. For instance, indices of order 3 S_{ijk} are associated to each triplet of parameters K_i, K_j, K_k:

$$S_{ijk} = \frac{V[E(Y|K_i, K_j, K_k)]}{V(Y)}$$

The sum of all n-order indices is always equal to 1. The computation of higher-order indices is expensive, as there are $\binom{n}{k}$ of such indices for k parameters. In practice, they are rarely used. They are not 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 it 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:

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

$K_{\sim i} = K_1, K_2, ..., K_{i-1}, K_{i+1}, ...K_n$ is the set of all parameters except K_i. Therefore, if a parameter has a total-effect index near zero, its uncertainty 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 Optimisation

In order to compute GSA indices, the search space of a group of parameters is sampled, aiming at finding the parameters whose variation influences the output of a function (or a model) the most. It is therefore not surprising that several attempts have been performed to combine Sensitivity analysis with optimisation 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 optimisation* [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 [22], EAs are used to find the worst possible parameter settings for a model, maximising 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 Sensitivity Analysis, providing the user with a general impression of the most influential parameters.

Another research line, presented in two technical reports [17, 16], aims at using the points sampled by a CMA-ES algorithm [11] during the optimisation process as the basis for a sensitivity analysis, 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. This methodology raises several theoretical questions that will need to be thoroughly analyzed before its widespread application.

In [7], the authors present an example where the use of GSA improves the EA efficiency. They use GSA measurements to reduce the problem's dimensionality, first optimising the values of a sub-set of the most sensitive parameters, and then restarting the evolution from the solutions found in this way, finally optimising the remaining values. However, preliminary results presented in [3] hint that this strategy may not always be viable.

3 Adaptive dimensionality reduction based on GSA

The idea of using progressive refinements techniques to perform a search in high dimensional spaces appeared as attractive for a long time. This very simple idea is at the origin, for instance, of the *messy genetic algorithm* scheme proposed by Goldberg et al. 25 years ago [8] : "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 [18, 12]. 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-influential parameters, via a sensitivity analysis of the fitness function with respect to each parameter in the search space. A straightforward

strategy for dimensionality reduction is then to ignore non-influential parameters in a first optimisation stage, like in [7].

4 Experimental analysis

We propose a series of counterexamples for testing the limits of dimensionality reduction based on GSA, in the same spirit as deceptive functions design [10, 9]: global information collected through statistical analysis of some features (building blocks statistics in the case of deceptiveness "à la Goldberg") yields puzzling information to the algorithm. Other interpretations may also stem from theoretical studies regarding the influence of local regularity features [13, 15]: global optima are located in very irregular areas, while attractive local optima are located inside smooth areas. Statistical features are actually not able to capture local irregularities and are thus yielding erroneous information to the algorithm [14].

The strategy that is tested relies on the following statement (factor fixing approach, see Section 2.2): *a low total effect index indicates a non-influential 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.

4.1 Algorithms

Three EAs have been tested: (i) CMA-ES, (ii) an explicit population based EA, implemented with the EASEA package[1] [4] and (iii) NSGA-II, a multi-objective genetic algorithm. The following schemes have been considered for progressive refinement:

- *Approach 1* performs an optimisation of the influential parameters only. Non-influential parameters are fixed to the middle of their interval of uncertainty.
- *Approach 2* is based on [7]. Influential parameters are optimised in a first stage, like in Approach 1, and then the best point is injected in the initial population of a second optimisation, this time using all parameters.

CMA-ES. The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [11] is a popular EA, widely used for many real-world optimisation problems. It is known for its robustness and computational efficiency. For Approach 2, CMA-ES is restarted as follows:

- The mean point is initialised to 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 $0.3 \times (range_{max} - range_{min})$.

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

EA. The second algorithm used in our tests is a classical EA, i.e. an explicit population based EA, programmed in EASEA [4]. 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.

NSGA-II. The Nondominated Sorting Genetic Algorithm [6] is a Multiobjective evolutionary algorithm. This algorithm builds a set of non-dominated solutions that approximates an optimal Pareto front. Thanks to a clever ranking and to the use of a crowding distance, the population stabilises on an efficient sampling of the Pareto front. Approach 2 with NSGA-II uses a similar setting as above, for the EASEA-EA.

4.2 Counterexample I

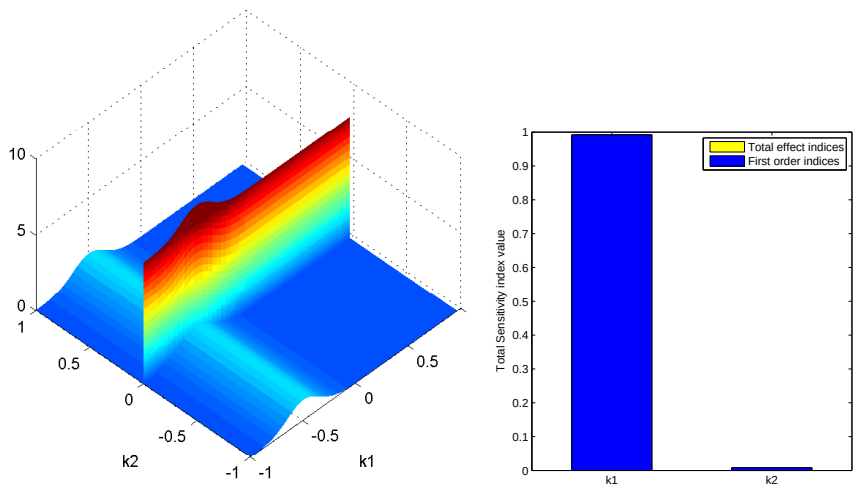

Fig. 1. *Counterexample I.* **(left)** In the fitness landscape, the peak of fit_1 is at $k2 = 0.0005$. The line $k2 = 0$ is at the bottom of the peak. **(right)** Sensitivity analysis shows that $k1$ is much more influential than $k2$.

The first counterexample is 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 projection of the fitness function on the subspace of non-influential parameters then provides an averaged viewpoint on the fitness landscape that conceals high, thin

peaks. We thus propose the following bi-dimensional function (Fig. 1):

$$fit_1(k1, k2) = g(k1, 1.33, -0.5, 0.3) + g(k2, 7.98, 0.0005, 0.05) + h(k1)$$

where g is a Gaussian: $g(k, a, b, c) = a \cdot exp(-\frac{(k-b)^2}{2c^2})$ and $k1, k2 \in \big[-1; 1\big]$

To make optimisation easier with respect to parameter $k1$, a small gradient, $h(k1)$ is added to the fitness:

$$h(k1) = \begin{cases} \frac{1}{1.0005}k1 + \frac{1}{1.0005} & \text{for } k1 \leq 0.0005 \\ -\frac{1}{0.9995}k1 + \frac{1}{0.9995} & \text{elsewhere} \end{cases}$$

A global sensitivity analysis, whose results are presented in Fig. 1 reads that $k1$ is influential whereas $k2$ is not, since the total effect index of $k2$ is far lower that the total effect index of $k1$.

	EASEA-EA	CMA-ES
Population size	$\mu = 200$	10
Offsprings size	$\lambda = 180$	-
Number of generations	35	632
Tournament selection	$Size = 2$	-
BLX-α Crossover	$p = 1.$	-
Log normal self adaptive mutation	$p = 1.$ $\tau = \sqrt{2}$	-
Number of Runs	100	100

Table 1. Settings for the EAs used in Counterexample I

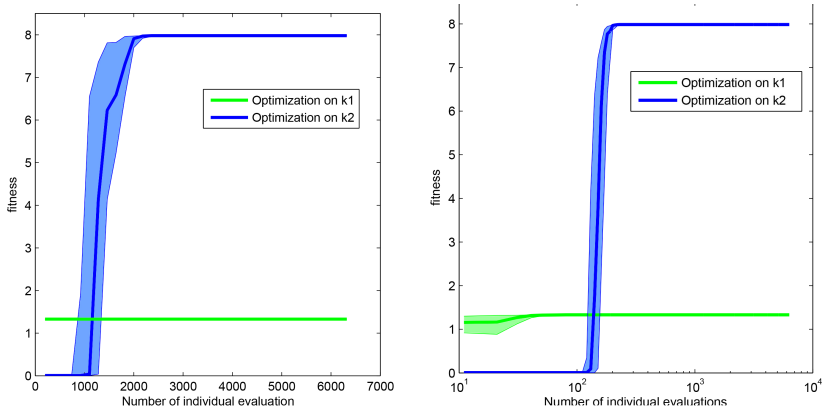

Fig. 2. *Counterexample I.* Comparison of optimisation runs on $k1$ and $k2$, respectively, using the EASEA-EA**(left)** and CMA-ES **(right)**. Statistics on 100 runs are displayed with median in bold and first- and third- quartile in thin lines of the same color.

Approach 1 is tested: optimisation is run on parameter $k1$ only, and the result is compared to an optimisation on parameter $k2$ only. Since $k1$ seems to bear all influence whereas $k2$ appears to be non-influential, it is naively expected that the optimisation on $k1$ will find a better value than the optimisation on $k2$. The algorithms' settings are reported in table 1. Statistics on 100 runs are displayed in Fig. 2 for the EASEA-EA and CMA-ES algorithms. In this case, optimising on the non-influential parameter is unexpectedly a better option than optimsing on the supposedly most influential parameter.

4.3 Counterexample II

A restart strategy (Approach 2 of Section 4.1) may counterbalance the problems presented above. We will see however that a restart strategy using GSA may still be puzzled. This is the purpose of counterexample II (Fig. 3).

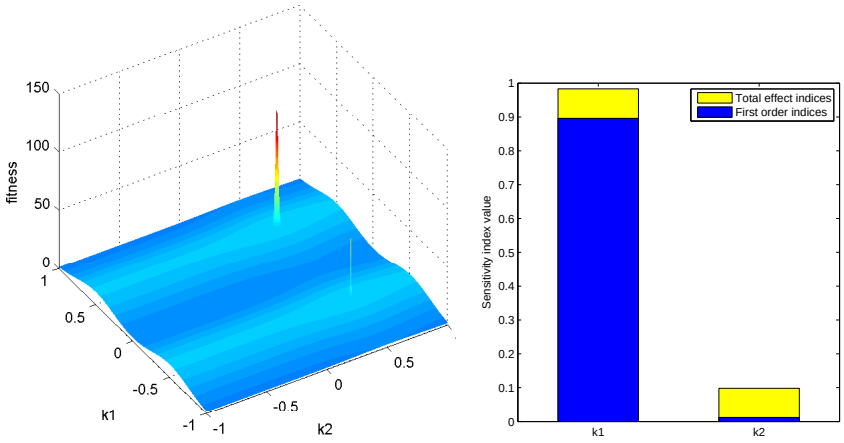

Fig. 3. *Counterexample II.* **(left)** fit_2 has two thin peaks, a very thin one corresponding to a local optimum at $(-0.5, 0.5)$ and a larger one, global optimum, at $(0.5, 0.5)$. **(right)** Sensitivity analysis shows that the total effect index for $k1$ is much higher than for $k2$.

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

$k1, k2 \in \big[-1;1\big]$, g and $g2d$ are Gaussians:

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

fit_2 has a local optimum at $(k1 = -0.5; k2 = 0.5)$, and a global optimum at $(k1 = 0.5; k2 = 0.5)$. A GSA on Counterexample II (See Fig. 3), shows that $k1$ can be considered as an influential parameter and $k2$ as a non-influential one.

A progressive refinement strategy (Approach 2) is compared to a plain optimisation (full search space) using a classical EA, with the settings reported in Table 4.3. Over 100 runs, the full search always finds the global optimum whereas the restart strategy (Approach 2) always get stuck on the local optimum (Fig. 4.3).

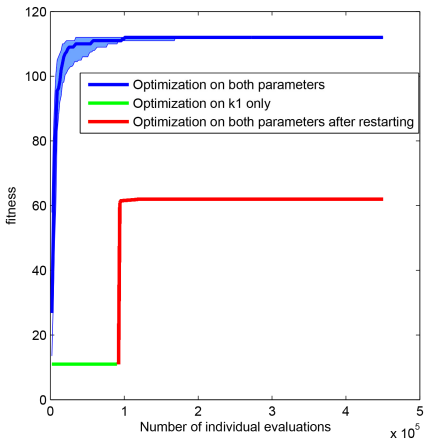

Fig. 4. *Counterexample II.* Statistics of 100 runs on Counterexample II with a classical EA.

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 2. *Counterexample II.* EA parameter setting, full search space and Approach 2.

This behaviour is due to the fact that the function is deceptive: when considering only $k1$ for optimisation, and fixing $k2$ to 0, the function has a maximum of 11.14 for $k1 = -0.5$ and a local maximum of 11.04 for $k1 = 0.5$. Thus, the first-stage optimisation 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 CMA-ES with two 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. The results are not reported here, but in both cases, we noticed that Approach 2 was bringing deceiving information to the algorithm, and delayed or even prevented convergence.

4.4 Counterexample III

The third counterexample is based on a multi-objective problem, to better shed light on the potential limits of the method presented in [7]. A bi-objective min-

imisation problem on a two parameters space has been derived using the fit_2 function. A small offset has been put on parameter $k1$ for the second objective, as follows:

$$fit_{Obj_1}(k1, k2) = -fit_2(k1, k2)$$

$$fit_{Obj_2}(k1, k2) = -fit_2(k1 + 0.05, k2)$$

The theoretical Pareto front is located in the $(k1, k2)$ parameter space, on the segment $\big[(0.45, 0.5); (0.5, 0.5)\big]$. A sub-optimal Pareto front also exists on the segment $\big[(-0.55, 0.5); (0.5, 0.5)\big]$.

As expected, a GSA on Counterexample III provides similar information on the behaviour of the two objective functions as for Counterexample II : $k1$ is influential on both objectives whereas $k2$ is not (See Fig. 5).

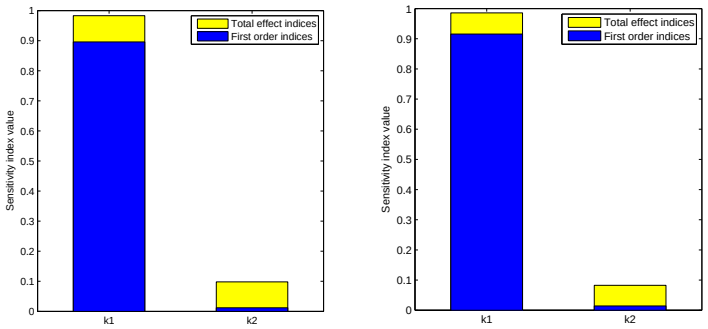

Fig. 5. *Counterexample III.* Sensitivity analysis on objective 1 **(left)** and on objective 2 **(right)**. For both objectives, the total effect index for $k1$ is much higher than for $k2$.

The restart strategy is compared to a classical approach, using the NSGA-II algorithm. The settings for NSGA-II are given in Table 4.4. The restart strategy always ends up near the sub-optimal Pareto front, whereas the classic strategy finds solutions near the optimal Pareto front. A typical result is displayed in Fig. 4.4.

For facilitating comparison, two performance metrics have been computed on 100 runs (see Fig. 7). The hypervolume indicator [24] computes the volume of the dominated portion of the objective space. A high hypervolume value means that the solutions are well spread along the objective space and/or are close to the optimal Pareto front. The convergence indicator [5] computes a distance between the current solution front and a predefined set of good solutions. Here, solutions have been taken on the theoretical Pareto front. A low value corresponds to a good approximation of the Pareto front.

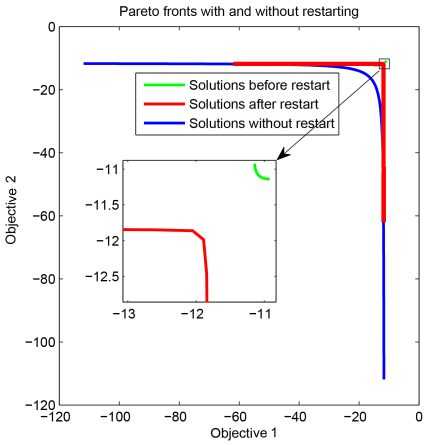

Fig. 6. *Counterexample III.* Typical Pareto front obtained with a classical NSGA-II and with the two steps restart strategy.

Population size	250
Number of generations	full search : 250 Approach 2 : 50 then 200
Number of Runs	100

Table 3. Settings for NSGA-II on Counterexample III.

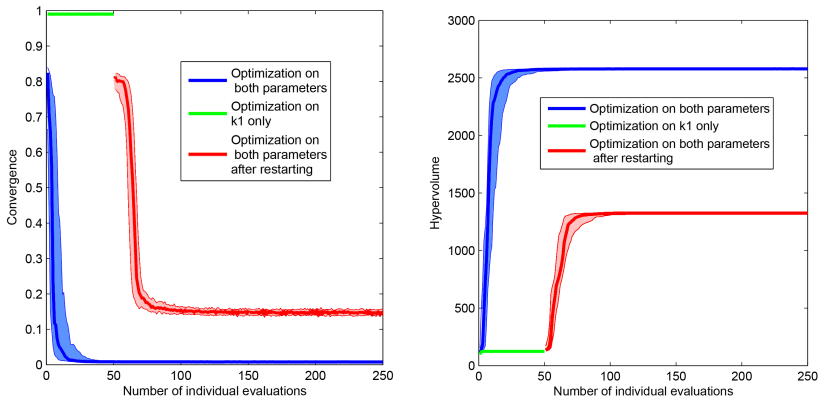

Fig. 7. *Counterexample III.* Convergence metric **(left)** and hypervolume metric **(right)** averaged on 100 runs using NSGA-II and a population size of 250.

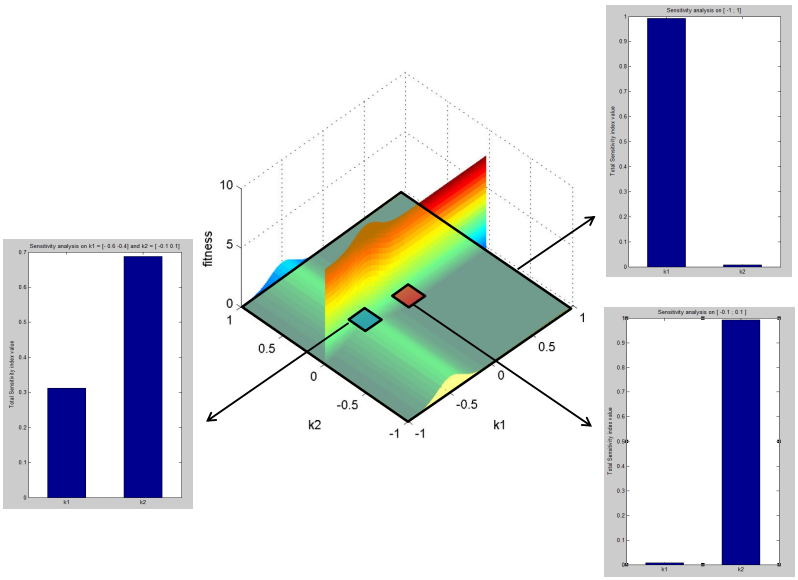

Fig. 8. Various sensitivity analyses on three sub-spaces for Counterexample I: parameters influences vary a lot !

5 Discussion

The counterexamples presented in Section 4 shed light on the fact that sensitivity analysis techniques may deliver misleading information to the optimisation 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 optimisation 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 influential on some subspace and not on another. Fig 8 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 optimisation 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 computational time that may sometimes be better spent by performing an optimisation process. Additionally, the averaged information provided by GSA may hide some interesting irregular areas where global optima could

be found. Finally, adaptive refinement methods, like Approach 2 presented in this paper, or the one proposed in [7], need to identify a non-negligible subset of non-influential parameters, which is not always the case, especially for complex optimisation problems. More progressive strategies may be imagined, but once again with all the risks tied 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, different 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 optimise first.

In this paper, we presented three case studies, specifically designed to provide deceiving information to sensitivity analysis used during an optimisation process. As a result, stochastic optimisation biased by this information has been experimentally proven unable to reach the global optimum. A simple progressive refinement optimisation scheme based on parameter prioritisation such as in [7] may work on some functions, but there is a risk of falling into a local optimum, from which escaping might prove to be hard. Even if parameter prioritisation might work better for multi-objective problems, thanks to a better diversity preservation mechanism necessary for a correct sampling of Pareto fronts, a multi-objective counterexample is still rather easy to design. This was the purpose of counterexample III.

An interesting point for further developments could be to determine in which cases GSA is beneficial. From this study we can conjecture that regularity of the fitness function may play an important role. If *global* sensitivity analysis has been proven to be puzzling to optimisation in some cases, *local* sensitivity analysis however remains interesting.Sobol indices computed locally for instance may be useful for tuning mutations, in the same spirit as what has been developed in [14], but with an associated computational cost to be taken into account.

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. Chabin, T., Tonda, A., Lutton, E.: Is global sensitivity analysis useful to evolutionary computation? In: Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference. pp. 1365–1366. ACM (2015)
4. Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it easea. In: Parallel Problem Solving from Nature PPSN VI. pp. 891–901. Springer (2000)

5. Deb, K., Jain, S.: Running performance metrics for evolutionary multi-objective optimizations. In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02),(Singapore). pp. 13–20. Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02),(Singapore) (2002)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on 6(2), 182–197 (2002)
7. 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)
8. Goldberg, D., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3(5), 493–530 (1989)
9. Goldberg, D.: Genetic algorithms and walsh fuctions: II. Deception and its analysis. Complex Systems 3(2), 153–171 (April 1989)
10. Goldberg, D.: Genetic algorithms and walsh functions: I. A gentle introduction. Complex Systems 3(2), 129–152 (April 1989)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)
12. Kargupta, H.: The gene expression messy genetic algorithm. In: International Conference on Evolutionary Computation. pp. 814–819 (1996)
13. Leblanc, B., Lutton, E.: Bitwise regularity and ga-hardness. In: ICEC 98, May 5-9, Anchorage, Alaska (1998)
14. 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)
15. Lutton, E., Véhel, J.L.: Hölder functions and deception of genetic algorithms. IEEE transactions on Evolutionary computation 2(2), 56–72 (July 1998)
16. Müller, C., Paul, G., Sbalzarini, I.: Sensitivities for free: Cma-es based sensitivity analysis. Tech. rep., ETH Zurich (2011)
17. Paul, G., Müller, C., Sbalzarini, I.: Sensitivity analysis from evolutionary algorithm search paths. Tech. rep., ETH Zurich (2011)
18. Rajeev, S., Krishnamoorthy, C.: Genetic algorithms-based methodologies for design optimization of trusses. Journal of Structural Engineering 123(3), 350–358 (1997)
19. 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)
20. Saltelli, A., Annoni, P.: How to avoid a perfunctory sensitivity analysis. Environmental Modelling & Software 25(12), 1508–1517 (2010)
21. 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)
22. Stonedahl, F., Wilensky, U.: Evolutionary robustness checking in the artificial anasazi model. In: AAAI Fall Symposium: Complex Adaptive Systems (2010)
23. 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)
24. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Parallel problem solving from nature—PPSN V. pp. 292–301. Springer (1998)