# Evolutionary Algorithm Parameter Tuning with Sensitivity Analysis

Frédéric Pinel, Grégoire Danoy, and Pascal Bouvry

FSTC/CSC/ILIAS, University of Luxembourg
6 Rue R. Coudenhove Kalergi, L-1359 Luxembourg
frederic.pinel@uni.lu, gregoire.danoy@uni.lu, pascal.bouvry@uni.lu

**Abstract.** This article introduces a generic sensitivity analysis method to measure the influence and interdependencies of Evolutionary Algorithms parameters. The proposed work focuses on its application to a Parallel Asynchronous Cellular Genetic Algorithm (PA-CGA). Experimental results on two different instances of a scheduling problem have demonstrated that some metaheuristic parameters values have little influence on the solution quality. On the opposite, some local search parameter values have a strong impact on the obtained results for both instances. This study highlights the benefits of the method, which significantly reduces the parameter search space.

**Keywords:** Evolutionary Algorithm, Parameter Tuning, Sensitivity Analysis

## 1 Introduction

Evolutionary Algorithms (EAs) have been used since many years to optimize combinatorial and continuous hard problems. These nature-inspired algorithms function by iteratively applying specific operators in order to modify potential solutions to a problem and converge to an optimal or near-optimal solution. Despite their application success, EAs remain highly dependent on their parameterization but also on the optimization problem class. Moreover, the complexity of recent EAs, such as cellular genetic algorithms (CGAs), implies an increase in the number of parameters to be set. As mentioned by De Jong in [8], the No Free Lunch (NFL) theorem state that no single algorithm will outperform all other algorithms on all classes of problems. This induces several key questions, including: "which parameters are useful to improve the EA performance?". Although a lot of works have been conducted in the field of parameter setting for EAs, most of these focused on independently searching for the best parameter values without considering if these parameters have a direct influence on the EA performance.

The contribution of this paper is the proposal of a generic of sensitivity analysis method to quantitatively study the influence and interdependencies of the parameters of an EA when applied on a specific optimization problem. The objective is to help the algorithm designer in parameter setting by narrowing the

parameter search space prior to optimizing their values. We here focused on a Parallel Asynshronous Cellular Genetic Algorithm (PA-CGA) [22] and analyzed its parameters influence on two different instances of a scheduling problem of independent tasks in a grid.

The paper structure is detailed next. The next section contains a brief survey on parameter setting techniques. Then section 3 presents the sensitivity analysis method. In section 4 a detailed description of the scheduling problem and of the PA-CGA is given. Section 5 and 6 respectively present the experimental setup used and discusses the obtained results. Finally in section 7 the conclusion and perspectives of the work are presented.

## 2 Related Work

Parameter setting can greatly influence the performance of Evolutionary Algorithms and therefore focused the interest of many researchers. Comprehensive surveys have been introduced by De Jong in [8], Eiben [10] and more recently by Kramer in [17].

As mentioned by Maturana et al. in [19], one of the main problems is to assess which parameters can lead to the algorithm transformation, i.e. improvement. Yet, they proposed a classification of parameters, distinguishing *behavioral parameters* (operators probabilities, population size) and *structural parameters* (encoding, choice of operators). A similar classification was proposed by Smit and Eiben in [27] distinguishing between *numerical* and *symbolic* parameters. In this work we focused on behavioral, respectively numerical parameters setting.

The EA parameter space can be explored in offline (before the search) or online (during the search) setting. Eiben in [9] classified these parameter techniques as parameter *tuning*, and *parameter control*. In this work we are interested in parameter setting before the run (i.e. tuning), for which a taxonomy extension has been proposed by Kramer in [17] (see Fig. 1).

Tuning by hand induces user experience for setting the EA parameters beforehand. This solution is largely predominant in the literature in which parameters are usually set based on empirical evaluations as mentioned in [19].

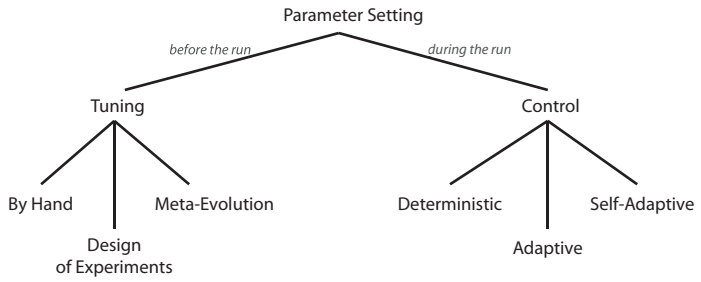


**Fig. 1.** Parameter setting in EA's taxonomy [17].

The second tuning class, design of experiments (DoE), refers to Bartz-Beielstein work on Sequential Parameter Optimization (SPO) [3] which is a heuristic combining classical and modern statistical techniques (e.g. latin hypercube sampling - LHS). The objective is to design the experimental plan prior to doing the experiments. Some works have focused on analyzing the sensitivity of parameter, but limited to the study of the independent influence of parameters values on the fitness. De Castro et al. in [7] studied the sensitivity of 3 parameters (number of antibodies, number of generated clones and amount antibodies to be replaced) of their Clonal Selection Algorithm (CLONEALG). Similarly, Ho et al. in [13] have analyzed the sensitivity of parameters of their Intelligent Genetic Algorithm (IGA), including mutation and crossover probabilities. In [20] Min et al. analyze the sensitivity of the population size and the termination condition (maximum number of generations) of a standard GA on a reverse logistics network problem. Finally, most lately Geem et al. in [11] analyzed the sensitivity of Harmony Search (HS) parameters (harmony size, memory considering rate and pitch adjustment).

The last parameter tuning class, meta-evolution, is also referred to as nested evolution. This is a two-level evolutionary process in which one algorithm optimizes the parameters of the second one. A recent approach has been proposed by Nannen and Eiben in [21], Relevance Estimation and Value Calibration of EA parameters (REVAC). It estimates the expected performance of the EA when parameter values are chosen from a probability density function (PDF) and includes a measure of the parameter relevance (normalized Shannon entropy).

The contribution of this work lies in the DoE class, in which existing approaches provide information on the best parameter values for the specific problem tackled. However these do not answer to two important questions:

1. Do all EAs parameters influence the algorithm performance on a specific problem instance?
2. What are the interdependencies between the parameters? Since, as Eiben already mentioned in [10] parameters are not independent.

The following section describes how sensitivity analysis can answer the drawbacks of the aforementioned approaches.

## 3 Sensitivity Analysis

Sensitivity analysis aims to identify how uncertainty in each of the parameters influence the uncertainty in the output [25] of a model. This technique can answer the following question: which factors cause the most and the least uncertainty in the output (also known as screening). This measures the importance of factors in the model analyzed. It is useful when designing experiments (DoE activity) and setting parameter, because it allows to focus on the most influential factors, possibly setting arbitrary values to the least influential ones. Moreover, this knowledge is also useful at design-time. The designer of a model intuitively develops an idea of its behavior. Sensitivity analysis allows the designer to verify

his hypothesis, and modify the model accordingly. This work therefore proposes to use sensitivity analysis to study the influential parameters of an EA on a specific problem class, i.e. scheduling problem of independent tasks in a grid. The objective is to reduce the EA parameter search space.

### 3.1 Desirable Sensitivity Analysis Properties

There are several ways to conduct a sensitivity analysis. Section 2 listed a few. Before presenting the suggested method, the desired characteristics of a method for sensitivity analysis are presented below. The method should:

- be model independent (it does not place requirements on the type of model to work),
- evaluate the effect of a parameter while varying other parameters (most manual analysis vary one parameter at a time, which hides interactions between parameters),
- cope with the influence of scale and shape in the model (the probability density function and its parameters),
- describe the influence of uncertainty in the parameters in a quantitative mode (the relative importance of each parameter should be quantified),
- capture the interaction between parameters.

### 3.2 Selected Method

These desired properties restrict the possible methods (such as using entropy as a measure of output uncertainty [21]). The chosen method is based on decomposing the variance of the output, as introduced by Saltelli et al. in [25]. The exact implementation used is an extension to the Fourier Amplitude Sensitivity Test proposed by Saltelli et al., called Fast99 [26]. This method allows the computation of first order effects and interactions for each parameter. Parameters interaction occurs when the effect of the parameters on the output is not a sum of their single (first order) effects.

### 3.3 Application of Sensitivity Analysis

The chosen method benefits from the properties presented in Section 3.1, therefore there are no model specific restrictions.

First, the goal of the analysis must be stated and the output of the model defined accordingly. For an evolutionary algorithm, this can be the quality of the solutions, the number of evaluations, the runtime of the implementation, etc. For each parameter of the model analyzed, the range of possible of values is required, along with their distribution in the range. These values come from experts in the application domain, or from the literature. Unless there are many parameters (greater than 30) or if the evaluation takes too much time (due to the number of parameters combinations), the Fast99 method mentioned in Section 3.2 is suitable. Otherwise, the qualitative method of Morris is better

suited (it is a One-At-a-Time method, or OAT). The method then produces a list of parameter combinations, for which the model is evaluated. In the case of an algorithm, the implementation of the algorithm is run with the prepared parameter combinations. The number of combinations is $N_{samples} \times N_{parameters}$ (1000 samples are typical). The method for the sensitivity analysis then collects the evaluation results and presents the linear and non-linear influence of each parameter. The next sections present a worked application.

## 4 Example Application

The presented sensibility analysis is performed on a parallel asynchronous cellular genetic algorithm [22] for scheduling of independent tasks in a grid. The following section provides a description of the problem and its representation, and section 4.2 presents the algorithm and the parameters used.

### 4.1 Problem Description

The problem the EA attempts to solve arises quite frequently in parameter sweep applications, such as Monte-Carlo simulations [6]. In these applications, many tasks with almost no interdependencies are generated and submitted to the grid system. Efficiency means to allocate tasks as fast as possible and to optimize some criterion, such as makespan or flowtime. Makespan is among the most important optimization criteria of a grid system. Indeed, it is a measure of its productivity (throughput). Task scheduling is treated as a single objective optimization problem, in which the makespan is minimized. *Makespan*, the finishing time of latest task, is defined as

$$\min_{S} \max\{F_t : t \in Tasks\} \ , \tag{1}$$

where $F_t$ is the finishing time of task $t$ in schedule $S$.

More precisely, assuming that the computing time needed to perform a task is known (assumption that is usually made in the literature [5, 12, 16]), the problem is represented with the Expected Time to Compute (ETC) model by Braun et al. [5]. The instance definition of the problem is as follows:

- $nb\_tasks$: the *number* of independent (user/application) *tasks* to be scheduled.
- $nb\_machines$: the *number* of heterogeneous *machine* candidates to participate in the planning.
- The *workload* of each task (in millions of instructions).
- The *computing capacity* of each machine (in *mips*).
- $ready_m$: ready time indicating when machine $m$ will have finished the previously assigned tasks.
- The Expected Time to Compute ($ETC$) matrix ($nb\_tasks \times nb\_machines$) in which $ETC[t][m]$ is the expected execution time of task $t$ on machine $m$.

The two benchmark instances used for this study consist of 512 tasks and 16 machines. Both instances represent different classes of ETC matrices. The classification is based on three parameters: task heterogeneity, machine heterogeneity, and consistency [2]. Instances are labelled as $u\_x\_yyzz$ where:

$u$ stands for uniform distribution (used in generating the matrix).
$x$ stands for the type of consistency ($c$ for consistent, $i$ for inconsistent, and $s$ for semi-consistent). An ETC matrix is considered consistent when the following is true: if a machine $m_i$ executes a task $t$ faster than machine $m_j$, then $m_i$ executes all tasks faster than $m_j$. Inconsistency means that a machine is faster for some tasks and slower for some others. An ETC matrix is considered semi-consistent if it contains a consistent sub-matrix.
$yy$ indicates the heterogeneity of the tasks ($hi$ means high, and $lo$ means low).
$zz$ indicates the heterogeneity of the resources ($hi$ means high, and $lo$ means low).

### 4.2 Parallel Asynchronous Cellular GA

The chosen EA is a parallel asynchronous CGA (PA-CGA) [22], based on [23]. Cellular genetic algorithms (cGAs) [1] are a kind of GA with a structured population in which individuals are spread in a two dimensional toroidal mesh and are only allowed to interact with their neighbors. The algorithm iteratively considers as current each individual in the grid, and individuals may only interact with individuals belonging to their neighborhood, so parents are chosen among the neighbors with a given criterion. Crossover and mutation operators are applied to the individuals, with probabilities $p_c$ and $p_m$ respectively. Afterwards, the algorithm computes the fitness value of the new offspring individual (or individuals), and inserts it (or one of them) instead of the current individual in the population following a given replacement policy. This loop is repeated until a termination condition is met.

In the PA-CGA, the population is partitioned into a number of contiguous blocks with a similar number of individuals (Figure 2). Each block contains $pop\_size/\#threads$ individuals, where $\#threads$ represents the number of concurrent threads executed. In order to preserve the exploration characteristics of the CGA, communication between individuals of different blocks is made possible. This neighborhood may include individuals from other population blocks. This allows an individual's genetic information to cross block boundaries.

The different threads evolve their population block independently and do not wait on the other threads to complete their generation (the evolution of all the individuals in their block) before pursuing their evolution. Hence, if a breeding loop takes longer for an individual of a given thread, the individuals evolved by the other threads may go through more generations.

The combination of a concurrent execution model with the neighborhoods crossing block boundaries leads to concurrent access to shared memory. To enable safe concurrent memory access, we synchronize access to individuals with a POSIX [15] read-write lock. This high-level mechanism allows concurrent reads

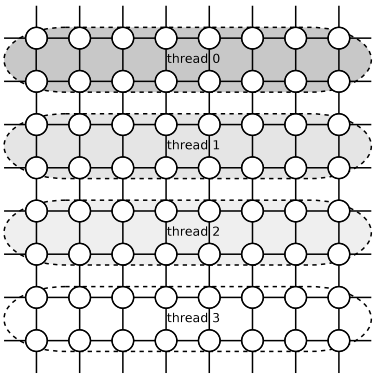


**Fig. 2.** Partition of an 8x8 population over 4 threads.

from different threads, but not concurrent reads with writes, nor concurrent writes. In the two latter cases, the operations are serialized.

The algorithm employes a local search operator, specific to the scheduling problem considered. This operator moves a randomly chosen task from the most loaded machine (a machine's load is the total of the tasks completion times) to a selected candidate machine among the $N$ least loaded ($N$ is a parameter). A candidate machine is selected if its new completion time, with the addition of the task moved, is the smallest of all the candidates. This new completion time must also remain inferior to the makespan. This operation is performed several times (a parameter of the local search).

The following parameters have been used for the PA-CGA. The population is initialized randomly, except for one individual obtained with the *Min-min* heuristic [14]. The selection operator used is binary tournament. The recombination operator used is the one-point (*opx*) crossover and the mutation operator moves one randomly chosen task to a randomly chosen machine. The neighborhood shape used is linear 5 (L5), also called Von Neumann neighborhood, composed of the 4 nearest individuals (measured in Manhattan distance), plus the individual evolved. The replacement strategy is "replace if better", i.e. the newly generated offspring replaces the current individual if it improves the parent fitness value.

## 5 Experimental Setup

The studied parameters of the PA-CGA, called factors in the context of sensitivity analysis, are summarized in Table 1.

For each factor considered in this study, a uniform distribution of the values is considered since we have no a priori indication of the correct values. Population size represents the dimension of the square shaped grid of the cellular GA, which can range between 8X8 to 32X32 individuals. Crossover rate is defined in a range between 0.1 to 1.0. Mutation is defined by its rate, ranging between 0.1

**Table 1.** Uncertainty in the model parameters

| Factor | Distribution | Range of values |
|---|---|---|
| Population size | uniform | 8x8 – 32x32 |
| Crossover rate (P_crossover) | uniform | 0.1 – 1.0 |
| Mutation rate (P_mutation) | uniform | 0.1 – 1.0 |
| Mutation iterations (Iter_mutation) | uniform | 1 – 5 |
| Local search rate (P_search) | uniform | 0.1 – 1.0 |
| Local search iterations (Iter_search) | uniform | 1 – 10 |
| Load for local search (Load_search) | uniform | 0.1 – 0.9 |
| Threads | uniform | 1 – 4 |

and 1.0, and the maximum number of mutations, ranging from 1 to 5. Local search is defined by the same properties (rate between 0.1 and 1.0 and maximum number of iterations between 1 and 10). The value range for the number of least loaded machines to consider 4.2 is 0.1 to 0.9. Finally, as the algorithm can be parallelized, the number of threads also varires between 1 and 4.

The stop condition for each run of the EA is 100 generations. Each set of factors generated for the analysis is used for 4 runs. The result is the average of the makespan over those 4 runs. The sensitivity analysis therefore considers a total of 6400 parameters combinations.

Sensitivity analysis is performed on the algorithm for two different instance files for which we provide their Blazewicz [4] notation:

- $u\_c\_512x16\_hihi$: $Q16|26.48 \leq p_j \leq 2892648.25|C_{max}$;
- $u\_c\_512x16\_lolo$: $Q16|1.44 \leq p_j \leq 975.30|C_{max}$.

The intention is to discover if different problem instances modify the factor prioritization results. The Fast99 implementation of the sensitivity analysis is provided by the R Sensitivity Analysis package [24].

## 6 Results

Figure 3 presents for each factor, their linear and non-linear (or interaction) effects on the output for the problem instance with high tasks and resources heterogeneity: the quality of the solution (the average makespan over 4 independent runs).

The benefits of the sensitivity analyzis are immediately visible. Indeed, the local search parameters and notably the maximum number of iterations, influence the most the output. It is indeed twice more important than the second

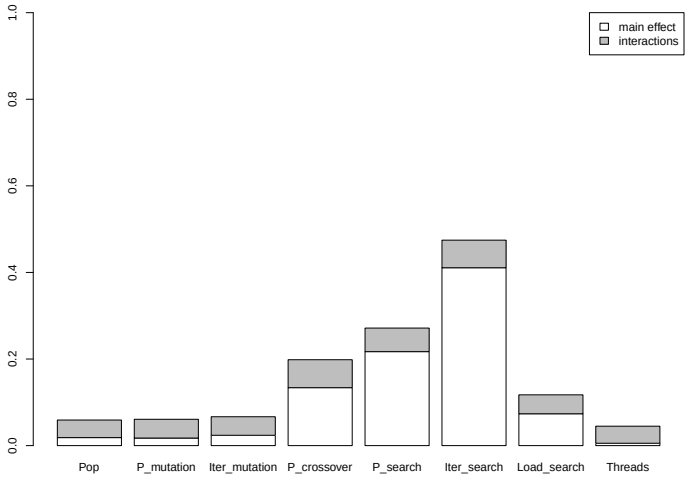


**Fig. 3.** sensitivity analysis, hihi instance

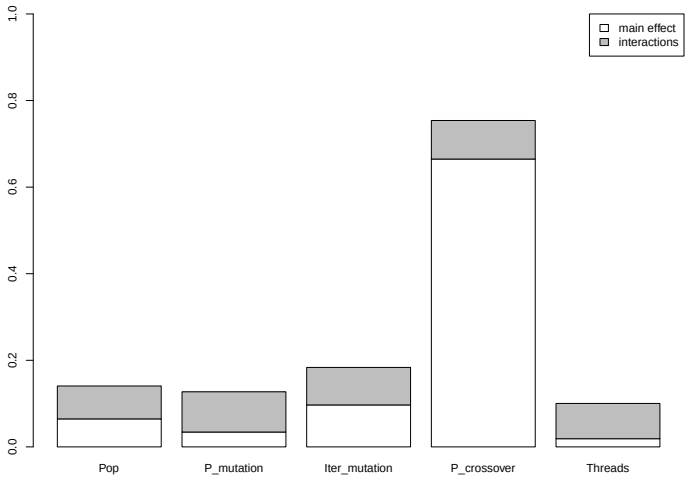


**Fig. 4.** sensitivity analysis, hihi instance with fixed local search parameters

most influential parameter, the local search rate. This result is consistent with related works in the scheduling literature which enlightened the importance of the local search when dealing with hybrid metaheuristics. This also justifies the hand tuning of the parameters performed for [22]. The third most important

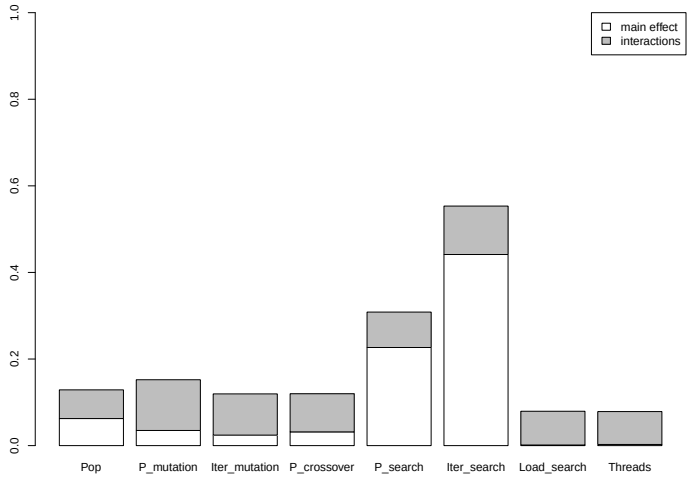


**Fig. 5.** sensitivity analysis, lolo instance

parameter the crossover rate. This is highlighted in Figure 4 which analyzes the effects on the output of the GA parameters, thus using fixed values for the local search. It appears that the crossover rate is at least six times more important than all the other GA parameters.

These results also highlights that some parameters play a limited role, i.e. population size, mutation rate and iterations as well as the number of threads. This is also beneficial because values which have a positive impact on other aspects of the algorithm, such as runtime, can be selected without impacting the quality of the solutions. Indeed, this algorithm was designed to be run for a limited period of time (wall clock), therefore choosing a smaller population size and a higher number of threads will allow the computation of more generations.

Figure 5 shows the same analysis for the instance with low tasks and resources heterogeneity. The two most influential parameters are similar to the hihi instance, local search iterations followed by the of local search rate. One difference can be noticed at the level of the third parameter in terms of importance. This parameter now consists in the "GA population size" while the "crossover rate" was used for the hihi instance. As can be seen in Figure 6, crossover has indeed 40% less influence than population size. Finally the load for local search has almost no influence on the output in the hihi case. Figure 6 shows that there are significant interaction effects, which mean that the remaining parameters combined, influence the output more than individually. The interaction part shows the total interactions (between two, three, etc parameters).

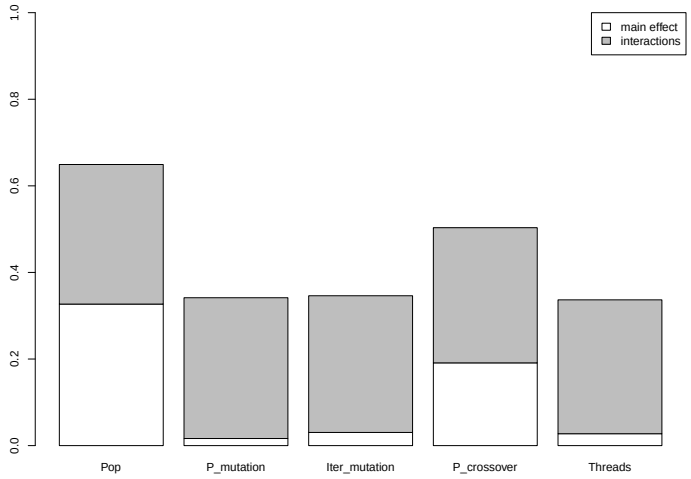


**Fig. 6.** sensitivity analysis, lolo instance with fixed local search parameters

To conclude, these first results are promising because they brought a first exploration of the relationship between the algorithm's influential parameters and the classes of problem instances.

## 7 Conclusion

In this paper, a variance based sensitivity analysis has been proposed to study the influence and interdependencies of the parameters of a Parallel Asynchronous Cellular Genetic Algorithm (PA-CGA). Experimental results on two different instances of a scheduling problem of independent tasks on a grid have shown that, for both problem instances, the two most impacting parameters are the local search ones. As expected, the GA parameters have a limited influence on the solution quality, except the crossover rate and the population size, respectively for the hihi and lolo instance. Current implementations are available [24] to make this analysis a systematic step in any EA experiment.

Future works will include studying the cost of the proposed approach and extending the sensitivity analysis of the PA-CGA parameters on a larger set of scheduling problem instances with different properties. Another targeted objective is to study the parameters sensitivity of the scheduling problem model itself.

## References

1. Alba, E., Dorronsoro, B.: Cellular Genetic Algorithms. Operations Research/Compuer Science Interfaces, Springer-Verlag Heidelberg (2008)

2. Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D., Ali, S.: Representing task and machine heterogeneities for heterogeneous. Journal of Science and Engineering, Special 50 th Anniversary Issue 3, 195–207 (2000)
3. Bartz-Beielstein, T., Lasarczyk, C.W.G., Preuss, M.: Sequential Parameter Optimization. In: 2005 IEEE Congress on Evolutionary Computation. vol. 1, pp. 773–780. IEEE (2005)
4. Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G.: Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24 (1983)
5. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hengsen, D., Freund, R.F.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing 61(6), 810–837 (2001)
6. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments. In: Heterogeneous Computing Workshop. pp. 349–363 (2000)
7. de Castro, L., Von Zuben, F.: Learning and optimization using the clonal selection principle. Evolutionary Computation, IEEE Transactions on 6(3), 239 –251 (Jun 2002)
8. DeJong, K.: Parameter setting in eas: a 30 year perspective. In: Lobo et al. [18], pp. 1–18
9. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evolutionary Computation 3(2), 124–141 (1999)
10. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. In: Lobo et al. [18], pp. 19–46
11. Geem, Z.: Harmony search algorithm for solving sudoku. In: Apolloni, B., Howlett, R., Jain, L. (eds.) Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, vol. 4692, pp. 371–378. Springer Berlin (2007)
12. Ghafoor, A., Yang, J.: Distributed heterogeneous supercomputing management system. IEEE Comput. 26(6), 78–86 (1993)
13. Ho, S.Y., Chen, H.M., Ho, S.J., Chen, T.K.: Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34(2), 1031 – 1044 (2004)
14. Ibarra, O.H., Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM 24(2), 280–289 (1977)
15. IEEE and The Open Group: Posix (ieee std 1003.1-2008, open group base specifications issue 7). http://www.unix.org (2008)
16. Kafil, M., Ahmad, I.: Optimal task assignment in heterogeneous distributed computing systems. IEEE Concurrency 6(3), 42–51 (1998)
17. Kramer, O.: Evolutionary self-adaptation: a survey of operators and strategy parameters. Evolutionary Intelligence 3, 51–65 (2010)
18. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms, Studies in Computational Intelligence, vol. 54. Springer (2007)
19. Maturana, J., Lardeux, F., Saubion, F.: Autonomous operator management for evolutionary algorithms. Journal of Heuristics 16, 881–909 (2010)
20. Min, H., Ko, H.J., Ko, C.S.: A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns. Omega 34(1), 56 – 69 (2006)

21. Nannen, V., Eiben, A.E.: Relevance estimation and value calibration of evolutionary algorithm parameters. In: Proceedings of the 20th international joint conference on Artifical intelligence. pp. 975–980. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007)
22. Pinel, F., Dorronsoro, B., Bouvry, P.: A new parallel asynchronous cellular genetic algorithm for scheduling in grids. In: Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010 (2010)
23. Pinel, F., Dorronsoro, B., Bouvry, P.: A new parallel asynchronous cellular genetic algorithm for de novo genomic sequencing. In: Proceedings of the IEEE International Conference on Soft Computing and Pattern Recognition (SOCPAR09). pp. 178–183 (2009)
24. Pujol, G.: sensitivity: Sensitivity Analysis (2008), r package version 1.4-0
25. Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M.: Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. Wiley (2004)
26. Saltelli, A., Tarantola, S., Chan, K.: A quantitative, model independent method for global sensitivity analysis of model output. Technometrics 41, 39–56 (1999)
27. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the Eleventh conference on Congress on Evolutionary Computation. pp. 399–406. CEC'09, IEEE Press, Piscataway, NJ, USA (2009)