# Simulated annealing, weighted simulated annealing and genetic algorithm at work

François Bergeret[1] and Philippe Besse[2]

[1]Motorola Semiconducteurs S.A., avenue Eisenhower, 31023 Toulouse Cedex, France
[2]Laboratoire de Statistique et Probabilités, U.M.R. CNRS 5583, Université Paul Sabatier, 31062 Toulouse cedex France

**Summary**

Two well known stochastic optimization algorithms, simulated annealing and genetic algorithm are compared when using a sample to minimize an objective function which is the expectation of a random variable. Since they lead to minimum depending on the sample, a weighted version of simulated annealing is proposed in order to reduce this kind of over-fit bias. The algorithms are implemented on an optimization problem related to quality control. A design of experiment is used to get the best trade-off between optimization and execution time. Simulated annealing appears to be more efficient than the genetic algorithm. With regard to the bias problem, the randomly weighted version of simulated annealing allows to achieve a solution less dependent on the sample and thus less biased.

**Keywords:** stochastic optimization, quality control, design of experiment.

## 1 Introduction

Stochastic optimization aims at finding the global minimum of an objective function. Simulated annealing (Kirkpatrick and al., 1983) and genetic algorithm (Holland, 1975) are two stochastic algorithms. The execution time of such algorithms is high in order to approach the global minimum. They are widely used in many applications (Davis, 1987) but few comparisons exist. We are interested in finding an optimal test sequence in quality control. It is a combinatorial optimization problem whose objective function can be modeled as the expectation of a random variable. The expectation depends on a parameter $\gamma$ which belongs to a finite set $\Phi$; $X$ is a random variable that modelizes the production. The expectation is taken over $X$ and is unknown because the distribution of $X$ is unknown. The problem is

$$\min_{\gamma \in \Phi} E[f(X, \gamma)].$$

Since the function $H(\gamma) = E[f(X, \gamma)]$ is usually not "convex" in $\gamma$, global optimization algorithms are used in order not to get stuck in a local minimum. To estimate the expectation, a sample is used: $(X_1, ..., X_n)$ are independent and identically distributed. This sample remains fixed for all the study. The natural idea is to estimate $H(\gamma)$ by the sample average:

$$H_n(\gamma) = \frac{1}{n} \sum_{i=1}^{n} f(X_i, \gamma).$$

In this paper, we aim at:

- adjusting the parameters of the algorithms,
- comparing the algorithms,
- reducing over-fit bias that may occur because the same sample is used to optimize and to estimate.

The adjustment of the parameters is done by design of experiment. The comparison of the algorithms shows that simulated annealing is more efficient than the genetic algorithm on the test optimization problem. A randomly weighted version of simulated annealing is then proposed to reduce over-fit bias: the objective function is estimated on a weighted sample. Numerical experiments show that this version reduces the bias.

We introduce notations and main results about simulated annealing in section 2. The weighted version of simulated annealing is presented in section 3. The implementation of the algorithms is detailed in section 4 and practical comparisons (simulated annealing versus genetic algorithm and simulated annealing versus weighted simulated annealing) are presented in section 5. Main results are discussed in section 6.

## 2 Simulated Annealing

Simulated Annealing (see Aarst and Korst, 1989) uses a direct analogy with the physical annealing process of solids which consists in two steps:

- Increase quickly the temperature to a maximum value at which the solid melts. The particles are randomly arranged.
- Decrease slowly the temperature until the particles arrange themselves in the ground state of the solid. The energy is then minimal.

In optimization, a solution is equivalent to a state of the physical system and the cost function we want to minimize is equivalent to the energy of a state. At the beginning of the algorithm, all the solutions are accepted. As the algorithm evolves, the probability of accepting deteriorations decreases with the temperature. At the end of the algorithm, only improvements are allowed and the system is considered to be frozen.

## 2.1 The algorithm

The following notations are necessary to define the algorithm:

- $\gamma_0$ is the initial (random) solution,
- $\gamma$ is the current solution,
- $\gamma'$ is a neighbour of $\gamma$, that is a solution which is close to $\gamma$ in some sense,
- $k$ is the iteration number of the changes of temperature,
- $c_k$ is the control parameter and plays the role of the temperature,
- a *proposed transition* is a neighbour visited,
- an *accepted transition* is the transformation of the current solution into a new one,
- $L_k$ is the number of proposed transitions for $k$ fixed,
- $L_k^a$ is the number of accepted transitions for $k$ fixed.

The algorithm has two nested loops:

- An outer loop on $k$ to control the temperature $c_k$. The algorithm exits this loop when a stopping criterium is satisfied.
- An inner loop is used to visit new solutions for $k$ fixed. The algorithm exits this loop if one of the two following conditions is satisfied:
    1. the number of accepted transitions $L_k^a$ reaches a certain value $L_{max}^a$,
    2. the number of proposed transitions $L_k$ reaches a certain value $L_{max}$;

$L_{max}^a$ and $L_{max}$ are two parameters of the algorithm. They are fixed. At each step of the inner loop, the algorithm generates a neighbour solution $\gamma'$ of the current solution $\gamma$ and compares the values of the objective function for the two solutions. Improvements of the objective function are always accepted in order to reach the minimum. Deteriorations in cost are accepted with the acceptance probability

$$A_{\gamma\gamma'}(c_k) = exp(\frac{H(\gamma) - H(\gamma')}{c_k}).$$

Let us remark that this probability depends on the difference $H(\gamma) - H(\gamma')$: too strong deteriorations are not allowed in order not to loose the previous information. It also depends on the control parameter $c_k$ which is very important: at the beginning $c_k$ is large and the domain space is visited randomly, then $c_k$ is slowly decreased to visit lower energy regions and to escape from local minima. As $c_k$ goes to 0, only improvements are allowed and the algorithm behaves like local search algorithms. In a pioneering work, S. Geman and D. Geman (1984) prove that suitable inverse cooling schedules ensure the convergence of the chain to a global minimum. More precise studies are the papers of Hajek (1985, 1988) and Tsitsiklis (1989). They give simple sufficient and necessary conditions on the cooling schedule $c_k$ for the algorithm to converge in probability to the global minima. Sharper estimates can be found in Catoni (1991), Chiang and Chow (1988) and various exensions can be found in Trouvé (1993), Hwang and Sheu (1992), Bélisle (1992). From a practical point of view, the essential problem is to choose the cooling schedule so that the convergence occurs as fast as possible. More over Catoni (1991) studies optimal cooling schedules given a finite-time execution. A finite-time implementation of the algorithm is now presented in the next section.

## 2.2 The cooling schedule

Some parameters of the algorithm were defined in the previous section. The evolution of these parameters is controlled by a cooling schedule. A cooling schedule specifies:

- an initial value of the control parameter $c_0$,
- a decremental function for decreasing the value of $c_k$,
- a stop criterion for the objective function,
- a finite number of transitions for each value of $c_k$.

Following this definition, several cooling schedules have been proposed in the literature. The most famous is the original schedule from Kirkpatrick and al. (1983):

- $c_0$ is initialized such that the acceptance ratio $r = L_0^a/L_0$ is close to one. In practice, before the algorithm starts, different values of $c$ are tested, in increasing order; at each attempt, the corresponding value of the acceptance ratio $r$ increases because $c$ increases. When the observed $r$ is close to one (e.g. $r > 0.99$), the corresponding value of $c$ is taken as the starting point $c_0$.

- The decremental function is

$$c_k = \alpha \times c_{k-1},$$

  $\alpha$ usually lies between 0.8 and 0.95.

- The algorithm is terminated if the value of the cost function remains unchanged for a number of consecutive iterations.

- As it has been told in 2.1, the value of $c_k$ is decreased if at least a number $L^a_{max}$ of transitions are accepted for $k$ fixed. However this requires $L_k \rightarrow \infty$ when $c_k \rightarrow 0$ because, when $c_k$ is close to 0, almost only improvements are accepted. This is why, as there are few improvements because the current solution is nearly optimal, $L_k$ is also bounded by some constant $L_{max}$.

With this cooling schedule the temperature decreases exponentially with $k$ , that is (assuming $ln\alpha \simeq \alpha - 1$):

$$c_k = c_0 exp[(\alpha - 1)k].$$

Another cooling schedule specifies a logarithmic decrement of the temperature:

$$c_k = d/log(k).$$

From a practical point of view, the essential problem is to choose the cooling scedule so that the convergence occurs as fast as possible. In a recent paper Catoni (1992) study optimal cooling schedules given a finite time execution. He shows that the exponential schedule is better as long as the execution time is finite.

## 2.3 Convergence

To provide conditions for the convergence of simulated annealing towards the set $S^*$ of global minima of $H$, the following specifications are needed.

$S_\gamma$ is the set of neighbours of $\gamma$. It is assumed that:

$$\begin{gathered} \gamma \notin S_\gamma, \\ \gamma' \in S_\gamma \Leftrightarrow \gamma \in S_{\gamma'}. \end{gathered} \tag{1}$$

Let us denotes

$$\chi_{S_\gamma}(\gamma') = \begin{cases} 1 & \text{if } \gamma' \in S_\gamma, \\ 0 & \text{otherwise.} \end{cases}$$

$G_{\gamma\gamma'}(c)$ is the generation probability. It is the probability of generating $\gamma'$ from a neibourghood of $\gamma$, $S_\gamma$. It is given by:

$$G_{\gamma\gamma'}(c) = G_{\gamma\gamma'} = \frac{\chi_{S_\gamma}(\gamma')}{\Theta}, \tag{2}$$

where $\Theta = |S_\gamma|, \ \forall \gamma \in \Phi$.

$A_{\gamma\gamma'}(c)$ is the acceptance probability. It is the probability of accepting $\gamma'$ generated from $S_\gamma$. It is given by:

$$A_{\gamma\gamma'} = \begin{cases} 1 & \text{if } H(\gamma') < H(\gamma), \\ exp(\frac{H(\gamma)-H(\gamma')}{c}) & \text{if } H(\gamma') \geq H(\gamma). \end{cases} \tag{3}$$

$P_{\gamma\gamma'}(c)$ denotes the transition probability, that is the probability of replacing the current solution $\gamma$ by $\gamma'$. As a result of (2) and (3), it is given by:

$$P_{\gamma\gamma'}(c) = \begin{cases} G_{\gamma\gamma'}A_{\gamma\gamma'}(c) & \text{if } \gamma \neq \gamma', \\ 1 - \sum_{l \in S \ l \neq \gamma} P_{\gamma l}(c) & \text{if } \gamma = \gamma'. \end{cases} \tag{4}$$

The algorithm is modeled as a Markov chain $(Y_l)_l$; $Y_l$ is the $l^{th}$ trial and $P(c)$ is the transition matrix associated with the algorithm. The stationary distribution is then defined by:

$$q_\gamma(c) = \lim_{l \to \infty} P(Y(l) = \gamma / Y(0) = \gamma'), \ \ \forall \gamma'.$$

The convergence proof works as follows (see for example Geman and Geman 1984, Gidas 1985, Hajek 1988):

1. The temperature $c$ is firstly assumed to be constant. The Markov chain $(Y_l)_l$ is then homogeneous. Under specifications (2), (3), (4) and if the following condition holds:

   $$\forall \gamma, \gamma' \in \Phi, \ \exists p \geq 1, \ \exists l_0, ..., l_p \in \Phi, \ l_0 = \gamma, \ l_p = \gamma' : \\ G_{l_k l_{k+1}} > 0, \ \ 0 \leq k \leq p-1, \tag{5}$$

   the Markov chain is irreducible and aperiodic. It converges towards an unique stationary distribution.

   If the equation (1) holds, then the stationary distribution satisfies the detailed balance equation:

   $$q_\gamma(c) P_{\gamma\gamma'}(c) = q_{\gamma'}(c) P_{\gamma'\gamma}(c), \ \ \forall \gamma, \gamma' \in \Phi.$$

   In that case, it can be shown that the distribution:

   $$q_\gamma(c) = \frac{exp(-H(\gamma)/c)}{N_0(c)}, \ \ \forall \gamma \in \Phi,$$

with

$$N_0(c) = \sum_{l \in \Phi} exp[-\frac{H(l)}{c}],$$

is the unique stationary distribution. Note that a weaker condition than condition (1) exists in Hwang and Sheu (1992). With this condition, it is still possible to precise the stationary distribution. In any case, the structure of the generation probability is the important point of the proof.

2. The temperature $c$ decreases. The simulated annealing algorithm is described by combining the homogeneous Markov chains of finite length into one single inhomogeneous Markov chain. If the cooling is done sufficiently slowly, the inhomogeneous Markov chain converges, when $c \to 0$, towards the uniform distribution on the set of global optima:

$$q^*_\gamma = \frac{1}{|S^*|} \chi_{S^*}(\gamma).$$

Strong ergodicity of the Markov chain is used to show the convergence. Isaacson and Madsen (1976) provide conditions for the strong ergodicity of a Markov chain. The Markov chain has to be weakly ergodic and the stationary distribution for $c$ fixed has to exist and has to satisfy some properties. For a detailed account and discussion of the origins of the various approaches commonly used, see for instance Catoni (1991). Another study (Miclo, 1996), based on the use of the relative entropy-distance and log-sobolev inequalities, leads to a simpler convergence proof, and extends other which have been published.

### Application to an expectation

Consider a simulated annealing algorithm with the objective function:

$$H_n(\gamma) = \frac{1}{n} \sum_{i=1}^{n} f(X_i, \gamma).$$

The temperature $c$ is fixed. The specifications (1), (2), (3), (4) are used. The neigbourhood structure is defined to satisfy the condition (5). Then, the stationary distribution is:

$$q_\gamma(c) = \frac{1}{N_0(c)} exp - [\frac{\frac{1}{n}\sum_{i=1}^{n} f(X_i, \gamma)}{c}],$$

with

$$N_0(c) = \sum_{l \in \Phi} exp - [\frac{\frac{1}{n}\sum_{i=1}^{n} f(X_i, l)}{c}].$$

The temperature $c$ decreases. If the cooling is done sufficiently slowly, simulated annealing converges towards the set of global optima of $H_n$.

# 3 Weighted simulated annealing

As stated in the introduction, a weighted version is proposed in this section to reduce over-fit bias since we are interested in finding a global minimum of $H$. The weighted algorithm is detailed and its convergence is studied.

## 3.1 The weighted algorithm

The aim of this adaptation is to avoid a minimum which would depend too much on the sample. This is done by introducing, at each step of the simulated annealing, a small perturbation. Each new evaluated solution is in a neighbourhood of the initial sample. More precisely, *at each step of the algorithm*, the cost function $H(\gamma)$ is estimated by:

$$H^b(\gamma, \omega) = \frac{1}{n} \sum_{i=1}^{n} \omega_i f(X_i, \gamma),$$

where the $\omega_i$'s are random weights such that $\sum_{i=1}^{n} \omega_i = n$. The weights are usually taken as the average of $B$ multinomial random vectors formed from $n$ draws on $n$ equally likely cells. The algorithm was presented yet in section 2. The difference is that each solution is a couple $(\gamma, \omega)$. When a new solution is generated, new weights are generated. The acceptance probability and the transition probability for a neighbour $(\gamma', \omega')$ of $(\gamma, \omega)$ are those of the standard algorithm. The function $H^b(\gamma, \omega)$ is then a weighted estimation of $H(\gamma)$. There is a similarity with the implementation of the bootstrap (Efron, 1979) by Monte Carlo methods because of the multinomial random vectors. The differences are:

- Each time the objective function is estimated in the algorithm, new weights are generated.
- The optimization is performed once. With a Monte Carlo method approximating the bootstrap, it would have been performed $B$ times leading each time to a different solution of the optimization problem.

The random generation of new weights at each step prevents from optimizing only on the initial sample. This may lead to a less biased estimation of the objective function at the end of the algorithm. Moreover, the additional computational cost of the algorithm is not important. It is mainly the generation of $B$ vectors of random weights for each evaluation of the objective function.

## 3.2 Convergence

In this section, we prove that the weighted version of the simulated annealing converges towards the set of global optima of the objective function, with regards to an enlarged set of solutions. The enlarged set of solutions is:

$$S = \Phi \times \Omega,$$

$\Omega$ is the set of vectors of weights $\omega = (\omega_i)_{1\leq i\leq n}$ with $\sum_{i=1}^n \omega_i = n$. The number of solutions is still finite because the weights are multinomial. Let us precise the new generation probability:
The probability of generating $(\gamma', \omega')$ from $(\gamma, \omega)$ is:

$$G_{(\gamma,\omega),(\gamma',\omega')} = \frac{\chi_{S_{(\gamma,\omega)}}(\gamma', \omega')}{\Theta} Q(\omega'), \tag{6}$$

where $\Theta = |S_\gamma|, \ \forall\gamma \in \Phi$ and $Q(\omega')$ is the probability of obtaining the vector of weights $\omega'$. By construction, $Q(\omega') > 0, \ \forall\omega' \in \Omega$. The acceptance and transitions probabilities are those of specifications (3) and (4) with:

$$H^b(\gamma, \omega) = \frac{1}{n}\sum_{i=1}^n \omega_i f(X_i, \gamma).$$

The results for the weighted simulated annealing are:

1. The temperature $c$ is constant. If the condition (5) is satisfied for the standard version, it is satisfied for the weighted version: if $G_{l_k l_{k+1}} > 0$ then $G_{(l_k,\omega)(l_{k+1},\omega')} > 0, \ \forall\omega, \omega' \in \Omega$ because all the weights are likely to be generated at each step. Thus, the Markov chain is irreducible and aperiodic and converges towards an unique stationary distribution.

   The condition:

   $$G_{(\gamma,\omega),(\gamma',\omega')} = G_{(\gamma',\omega'),(\gamma,\omega)}, \quad \forall(\gamma, \omega), (\gamma', \omega') \in S,$$

   does not hold with the generation probability (6) because all vectors $\omega$ are not equally likely. For this reason, it is not easy to precise the stationary distribution. However, conditions of Hwang and Sheu (1992) can be used to prove the convergence. They introduce the generalized simulated annealing and propose the Hajek's condition to prove the convergence. It can be shown that the Hajek's condition is satisfied if:

   $$\forall(\gamma, \omega), (\gamma', \omega') \in S, \quad P((\gamma, \omega), (\gamma', \omega')) > 0 \Leftrightarrow P((\gamma', \omega'), (\gamma, \omega)) > 0.$$

   This condition is satisfied with the generation probability (6).

2. The temperature $c$ decreases. If the cooling is done sufficiently slowly, the inhomogeneous Markov chain converges, when $c \to 0$, towards some distribution on the set of global optima of $\Phi \times \Omega$.

The homogeneous Markov chain converges towards a stationary distribution because the optimization is performed on $\Phi \times \Omega$. However, the bias reduction comes from the following point: each possible vector of weights is in a neighbourhood of the current vector of weights. So the optimization is not really performed in $\omega$ since previous solutions in $\omega$ are forgotten. The perturbation only allows not to strongly depend on the initial sample in order to reduce the bias.

# 4 Application in Quality Control

## 4.1 The problem

After manufacturing, electronic devices are tested in order to make sure that the products meet the specifications. The control consists in a number of electrical measurements. Assume that the cost of each measurement (also called test) is known. Assume also that the cost of an undetected bad part is known. Savings are possible by reducing the number of tests but the proportion of bad parts undetected will increase. Thus there is a trade-off between the cost of the test and the cost of undetected bad parts in order to minimize the total manufacturing cost. The selection of the tests that optimize this objective function is a combinatorial optimization problem solved by the simulated annealing and genetic algorithm. Let us introduce some definitions to precise the problem:

- $q$ is the original number of tests.
- A test combination $\gamma$ is a vector of length $q$. $\gamma_i = 1$ means that the test $i$ is in the combination, $\gamma_i = 0$ means that $i$ is not in the combination.
- The domain space for $\gamma$ is:
$$\Phi = \{0, 1\}^q.$$
- $\gamma^p$ is the present test sequence, that is $\gamma_i^p = 1$, $\forall i$.
- $TC(\gamma)$ is the test cost per lot generated by $\gamma$.
- $UC(X, \gamma))$ is a random variable. It represents the cost of the undetected bad parts for the lot $X$.
- $f(X, \gamma)$ represents net savings per lot:
$$f(X, \gamma) = TC(\gamma^p) - TC(\gamma) - UC(X, \gamma).$$
- The objective function is then:
$$H(\gamma) = E[f(X, \gamma)].$$

The function $H(\gamma)$ is to be maximized. According to the definitions of the sections 2 and 3, $H(\gamma)$ is estimated by $H_n(\gamma)$ or $H^b(\gamma, \omega)$ on a sample of lots taken from the production. The number of test combinations is $2^q$ and $q$ ranges from 7 to 700 depending on the product. An exhaustive search is impossible for most products because the computational cost grows exponentially with $q$. For $q < 100$, a step by step technique can be used (Bergeret and Chandon, 1995), but for $q \geq 100$ the results depend strongly on the initial solution. Stochastic algorithms are then used in order to approach the global minimum of the objective function. In the next section, a genetic algorithm is presented in order to compare it with the simulated annealing on this real-world problem.

### 4.2 Genetic algorithms

Genetic algorithms are widely used in many applications (Davis, 1991). They are based on the evolution process: the stronger individuals survive as the generations go on. A solution of the optimization problem is equivalent to an individual. The individuals are grouped in a population, they are crossed, muted and eventually selected according to their fitness value. This cycle is repeated for each generation of the algorithm and stops when no improvement is possible.

The crossover and mutation operators allow the exploration of new regions in the domain space. The crossover works on two individuals of the population to generate two new individuals. It occurs with a certain probability. Mutation makes a small change on a single individual. It occurs with a small probability. As the generations go on, the individuals in the population are better and better because the selection mechanism keeps the best individuals for the next generation. To our knownledge, Cerf (1994) gives the first convergence results for the genetic algorithm to converge, as time goes on, to the global minima of the fitness function. More precisely, Cerf proposed a model for genetic algorithms and formulated the conditions underwhich such a model may be immersed into a generalized simulated annealing framework. Then, he gives several conditions on the rates of decrease of the corresponding cooling schedule to ensure all the particles visit the set of global minima in finite time or as time goes on, when the number of particles is greater than a critical value.

### 4.3 Algorithms optimization by design of experiment

Some parameters of the algorithms have to be tuned for a finite-time implementation of the algorithms. Typical values for these parameters can be found in applications (Davis, 1987) but there is no general theory to fix them. In addition, the best values for these parameters may depend on the problem. For these reasons, we decide to run 2 designs of experiment, one for each type of algorithm (simulated annealing and genetic algorithm). They have two main advantages:

- Changes in the parameters values are organized. Thanks to the orthogonality of the design, the analysis of variance (ANOVA) allows to estimate each effect independently of the other effects.

- Several parameters are changed together. For some designs it is possible to analyze the interactions between the parameters.

An exponential cooling schedule is chosen for the simulated annealing algorithm. Such schedules are easy to code and we know (Catoni, 1992) that they are much faster than the logarithmic cooling schedule. It has three parameters: the decrement coefficient $\alpha$, the minimum number of transitions

| Factor | Lower level | Higher level |
|---|---|---|
| $\alpha$ | 0.8 | 0.95 |
| $L_a$ | 60 | 120 |
| $L_{max}$ | 200 | 300 |

Table 1: Levels of the factors for the simulated annealing algorithm.

| Factor | Lower level | Higher level |
|---|---|---|
| $m$ | 100 | 500 |
| $p_c$ | 0.5 | 0.9 |
| $p_m$ | 0.001 | 0.005 |
| $n_g$ | 25 | 50 |

Table 2: Levels of the factors for the genetic algorithm.

for $k$ fixed, $L^a_{max}$ and the maximum number of proposed transitions $L_{max}$. We also use a standard genetic algorithm with four parameters: the population size $m$, the number of generations $n_g$, the probabilities of crossover and mutation, $p_c$ and $p_m$. These parameters will be the factors of the ANOVA. The factors are assigned two levels, low and high. Generally, these levels are the extreme values found for these parameters in other applications. We find in table 1 and table 2 the levels selected for simulated annealing and genetic algorithm. There are two response variables: the value of the objective function at the minimum and the execution time. The experiment was done with 94 tests on a device which represents a wide range of devices. Further experiments were done on other devices in order to confirm the results. In addition, the objective function for the experiment is $H_n(\gamma)$. A full factorial is used in both cases in order to analyze all the interactions; $2^3$ experiments are necessary for the simulated annealing and $2^4$ for the genetic algorithm. We give the results of the analysis of variance (NS means that the factor is not significant at the 5 % level, *sign.* means that the factor is significant). The results are given in tables 3 and 4. For the simulated annealing algorithm, it is interesting to see that there is no significant factor on the minimum. As the three factors and an interaction are significant on the execution time, it is possible to fix the values at their lower level. It will reduce the execution time without effecting the quality of the result.

For the genetic algorithm, the significant factors are ranked according to the F-ratio:

1. The most important factor is the size of the population. The minimum is 87 on average for $m = 100$ and 100 on average for $m = 500$. This factor is also significant on the execution time, $m = 500$ leads to a

| Factor | Minimum | Execution time |
|---|---|---|
| $\alpha$ | NS | *sign.* |
| $L_a$ | NS | *sign.* |
| $L_{max}$ | NS | *sign.* |
| Interaction $\alpha.L_a$ | NS | *sign.* |
| Other interactions | NS | NS |

Table 3: Results of the ANOVA for simulated annealing.

| Factor | Minimum | Execution time |
|---|---|---|
| $m$ | *sign.* | *sign.* |
| $p_c$ | *sign.* | NS |
| $p_m$ | NS | NS |
| $n_g$ | NS | *sign.* |
| Interaction $m.n_g$ | *sign.* | *sign.* |
| Other interactions | NS | NS |

Table 4: Results of the ANOVA for the genetic algorithm.

longer time.

2. $p_c$ is also significant on the minimum but not on the execution time. $p_c = 0.9$ gives better results.

3. The interaction $m.n_g$ is significant on both response variables. To improve the minimum, one may fix *the two parameters* at their higher value. However this will increase strongly the execution time.

More experiments have to be carried in order to test other values for the parameters. Nevertheless, the parameters can be fixed in order to improve the minimum. If a parameter is not significant on the minimum we fix it to reduce the execution time. The parameters being optimized, we can compare the algorithms.

# 5 Comparison of the algorithms

Two problems have to be solved: the selection of the best algorithm for optimization purposes, and the estimation of the objective function in order to reduce the bias that may occur when the same sample is used to optimize and to estimate. The comparison of the algorithms is first done on $H_n(\gamma)$ which is the natural estimation of $H(\gamma)$. When the best algorithm is selected, the simple version and the weighted version are compared.

| $q$ | Genetic algorithm | Simulated annealing |
|---|---|---|
| 19 | 94 (4.5) | 100 (0.2) |
| 69 | 96 (2.6) | 100 (1.7) |
| 94 | 99 (2.6) | 100 (0.4) |
| 452 | 93 (3.2) | 100 (0.9) |

Table 5: Quality of the solution

| $q$ | Genetic algorithm | Simulated annealing |
|---|---|---|
| 19 | 6 | 2 |
| 69 | 598 | 40 |
| 94 | 683 | 44 |
| 452 | 1730 | 191 |

Table 6: Execution time (minutes)

### 5.1 Simulated annealing versus genetic algorithm

We find a comparison of the genetic algorithm and simulated annealing in Park and Carter (1995). For the max-clique problem, they conclude that simulated annealing is better than the genetic algorithm, both for the quality of the solution and for the time complexity. In order to compare the algorithms on a real world problem, we test them on the objective function presented in 4. Four devices are used. They represent the range of the complexity of the optimization problem, that is the number of tests. The results are the average on three replications, the standard deviation is given in parentheses. Moreover, the result are normalized to 100 for the minimum. The results for the execution time are given in minutes. They represent the execution time on an Apollo 400 workstation.

In all the cases of the table 5, simulated annealing outperforms the genetic algorithm. The difference for the quality of the solution is 7 % for the complex device. In addition, the results for simulated annealing are less variable, which is very important in a real-world problem because one cannot afford several runs of the algorithm. Execution times are much longer for the genetic algorithm, see table 6. We can argue that the parameters are set at values that increase the execution time. However this is not sufficient to reach the minimum of the simulated annealing. To see more precisely the effect of the population size on the quality of the solution, $m$ is increased in the genetic algorithm, see figure 1. As the result of the ANOVA showed an interaction between $m$ and $n_g$, the number of generations is always set at its higher level, $n_g = 50$.

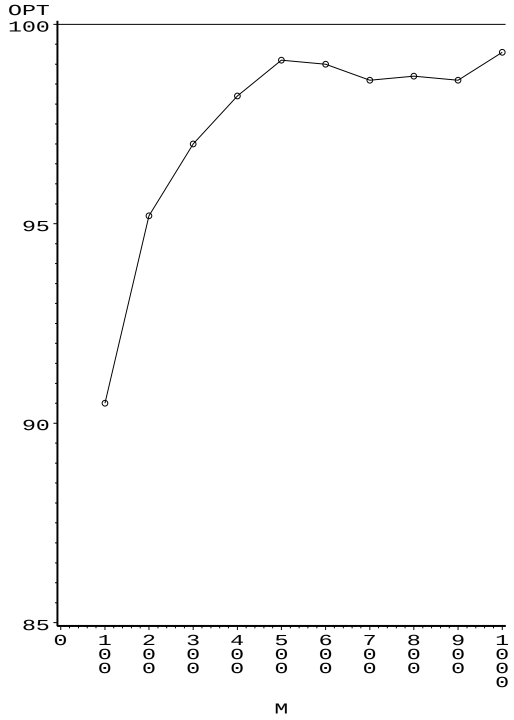


Figure 1: *The effect of the population size on the genetic algorithm.*

The quality of the solution (OPT) increases when $m$ is increased from 100 to 500. When $m$ is greater than 500, there is no significant improvement. The best result is for $m = 1000$ but is does not reach the result of the simulated annealing (100); 52 hours are necessary to obtain this solution when the simulated annealing takes less than one hour. We do not test higher values for $m$ and $n_g$ because of their computational cost. However, we can conclude that on this problem, simulated annealing is more efficient than the genetic algorithm.

## 5.2 Simulated annealing versus weighted simulated annealing

The conclusions of the previous section show that simulated annealing is more efficient than the genetic algorithm. So we concentrate on a weighted version of simulated annealing. Note that a weighted version of the genetic algorithm is also possible by estimating $H(\gamma)$ by $H^b(\gamma, \omega)$ for each individual of the population. To estimate the bias, we use a test sample (ts). Suppose that $\gamma^*$ is built with $n$ lots, the learning sample; $n'$ additional lots, independent of the learning sample, are then used to have an unbiased estimation of the

| $n$ | Standard algorithm | Weighted version |
|---|---|---|
| 50 | 1.43 | 0.95 |
| 100 | 0.62 | 0.49 |
| 150 | 0.99 | 1.02 |

Table 7: Bias estimation

objective function:

$$H^{ts}(\gamma^*) = \frac{1}{n'} \sum_{i=1}^{n'} f(X_i, \gamma^*).$$

$H^{ts}(\gamma^*)$ is unbiased for $H(\gamma^*)$ because the empirical average is an unbiased estimation of the expectation and because the test sample is independent of the learning sample. The bias for $H_n(\gamma^*)$ is then estimated by

$$\widehat{B}_n = H_n(\gamma^*) - H^{ts}(\gamma^*).$$

Empirical results on the devices of the section 5.1 show that the bias increases with the number of tests. When there are few tests, yields (proportion of good devices) are higher and less variable: with regards to the bias, the results for the weighted algorithm are very close of the results of the standard algorithm. When there are a lot of tests, yields are more variable and the number of undetected bad parts is far too optimistic when estimated on the learning sample. So, next results are for a very complex device which has 619 tests. The table 7 compares the results for different learning sample size. They are obtained with the standard simulated annealing and the weighted version. The weights are the average of 30 multinomial random vectors from $n$ draws on $n$ equally likely cells. The test sample size is always $n' = 100$. It consists of the same lots. Surprisingly, the bias is not a decreasing function of the learning sample size. Some of the tests were only selected when $n = 100$ and these tests rejected devices in the test sample. For $n = 50$ or $n = 150$, the bias was then increased.

We can observe that the bias estimation is reduced by 34% with the weighted version when the learning sample size is small ($n = 50$). The difference decreases when $n = 100$ with a 21 % reduction. When $n = 150$ the bias estimation is slightly higher for the weighted version. It means that the sample size is large enough and the weights does not improve the estimations.

The number of tests in the optimal test sequence is higher for the weighted algorithm. For the device with 619 tests, $H^{ts}(\gamma^*)$, which is an unbiased estimation of expected net savings, is slightly lower for the weighted version of simulated annealing. However, for the implementation of the optimal test sequence, engineers may be interested in reducing the number of undetected bad devices and the weighted algorithm may be useful.

# 6 Discussion

The adjustment of the parameters of the algorithms by design of experiment has been very useful. Faster simulated annealing does not have a negative effect on the optimization. On the opposite, the genetic algorithm has to run for a long time in order to improve its efficiency.

Simulated annealing is more efficient than the genetic algorithm on the test optimization problem. This result may depend on the form of the objective function. It seems that the objective function for the test problem does not have too many local optima:

- comparisons with a step by step technique were made on a complex device (619 tests). The results are better for simulated annealing but the difference is not very important.
- the results of the design of experiment allow to decrease quickly the temperature for the simulated annealing without affecting the quality of the optimization.

The genetic algorithm may be more efficient when the objective function is more chaotic because it allows the exploration of new solutions very different of the previous ones. Other comparisons have to be done on other problems in order to confirm the superiority of simulated annealing.

Experiments show that the weighted simulated annealing reduces the bias. The idea is that the stationary distribution for the weighted version is better in a certain sense. The behavior of this distribution is now studied. More precisely, the weighted algorithm may be more efficient for the optimization of an expectation.

## References

Aarst E. and Korst J. (1989), *Simulated Annealing and Boltzmann machines*, Wiley and sons.

Bélisle C. (1992), Convergence theorems for a class of simulated annealing algorithms on $R^d$, *J. Applied Probability*, **29**, 885-895.

Bergeret F. and Chandon Y. (1995), Recherche d'une séquence de test optimale en contrôle de fabrication, *Revue de Statistique Appliquée*, **XLIII, 3**, 21-33.

Catoni O. (1991), Sharp large deviations estimates for simulated annealing algorithms, *Annales de l'Institut Henri Poincaré*, **27**, 3, 291-383.

Catoni O. (1992), Rates of convergence for sequential annealing: a large deviation approach, in: R. Azencott Ed., *Simulated Annealing: parallelization techniques*, Wiley and sons, 25-35.

Cerf R. (1994), *Une théorie asymptotique des algorithmes génétiques*, Thèse de Doctorat, Université Montpellier 2.

Chiang T.S. and Chow Y. (1988), On the convergence rate of annealing processes, *Siam J. Control Optimization*, **26**, No 6.

Davis L. (1987), *Genetic Algorithms and Simulated Annealing*, Pitman, London.

Davis L. (1991), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

Efron B. (1979), Bootstrap methods : another look at the jackknife, *Ann. Stat.*, **7**, 1-26.

Geman S. and Geman D. (1984), Stochastic relaxation, Gibbs distribution, and the Bayesian restauration of images. *Institute of Electrical and Electronics Engineers. Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721-741.

Gidas B. (1985), Non stationary Markov chains and convergence of the annealing algorithm, *Journal of Statistical Physics*, **39**, 73-131.

Hajek B. (1985), A tutorial survey of theory and applications of simulated annealing, *IEEE Conference on Decision and Control.*

Hajek B. (1988), Cooling schedule for optimal annealing, *Mathematics of Operations Research*, **13**, 311-329.

Holland J. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Hwang C.R. and Sheu S.J. (1992), Singular perturbed Markov chains and exact behaviors of simulated annealing processes, *Journal of Theoretical Probability*, **5**, 223-249.

Isaacson D. and Madsen R. (1976), *Markov Chains*, Wiley.

Kirkpatrick S., Gellat C. and Vecchi M. (1983), Optimization by simulated annealing, *Science*, **220**, 671-679.

Miclo L. (1996), Remarques sur l'hypercontractivité et l'évolution de l'entropie pour les chaînes de Markov finies, to appear in J. Azéma P.A. Meyer, and M. Yor editors, *Séminaire de probabilité*, Lecture notes in Mathematics, Springer Verlag.

Park K. and Carter B. (1995), *On the effectiveness of genetic search in combinatorial optimization*, in Proc. 10th ACM Symposium on Applied Computing, Genetic Algorithms and Optimization Track.

Trouvé A. (1993) *Parallélisation massive du recuit simulé*, Thèse de doctorat, Université Paris XI.

Tsitsiklis J.N. (1989), Markov chains with rare transitions and simulated annealing, *Math. Op. Res*, **14**, 1.