History and Immortality in Evolutionary Computation

Benoit Leblanc[1], Evelyne Lutton[1], Bertrand Braunschweig[2], Hervé Toulhoat[2]

[1] INRIA, projet FRACTALES 78150 Le Chesnay - France
{Benoit.Leblanc, Evelyne.Lutton}@inria.fr
[2] Institut Français du Pétrole 1 et 4, avenue de Bois-Préau
BP 311 - 92852 Rueil-Malmaison Cedex - France
{Bertrand.Braunschweig, Herve.Toulhoat}@ifp.fr

Abstract. When considering noisy fitness functions for some CPU-time consuming applications, a trade-off problem arise: how to reduce the influence of the noise while not increasing too much computation time. In this paper, we propose and experiment some new strategies based on an exploitation of historical information on the algorithm evolution, and a non-generational evolutionary algorithm.

1 Introduction.

Handling noise in Evolution Algorithms has already been studied for the reason that most real-world problems present some noisy behavior, with many possible origins. This difficulty has often been successfully overcome by raising the population size [3] or by making multiple evaluations of the same individual ([5], [9]), using an average as fitness score.

We address here problems where fitness is noisy and in the same time computationally expensive, reducing the applicability of the previous solutions. Considering that each fitness evaluation bears important information that we do not want to lose, an exploitation of the *history* of evaluations is a solution to reduce the misleading noise.

Such a technique has already been experimented by Sano and Kita in [10] for noisy functions, by Corn and al [2] and Zitzler and al [11] for multiobjective optimisation. In Section 2, we propose a similar system of history-based fitness scoring, relying on a *genetic database*. Then in Section 3, it is shown that this genetic database may also be used to produce offspring. A sharing technique complements this scheme, it is described in Section 4. Finally experiments on two multimodal test functions are presented.

2 Historical information.

2.1 Motivation.

Inspired by the principles of Darwinian evolution, evolutionary algorithms (EA) are based on the concept of evolving population. The important size of population guarantees the redundancy of information (genes and their expression) and

its diversity, so the "death" of old individuals is not a problem, but is rather seen as an important evolution mechanism.

Here we deal with the class of problems where the total number of individuals created during the evolution is limited. This constraint arises for example when the fitness evaluation takes a long time. Moreover if the evaluation is subject to noise, the problem of accuracy of information becomes crucial. As stated before, we cannot afford raising too much the population size or the number of evaluations for the same individual.

To reduce the effect of noise, we therefore propose to use similarities between individuals (many instances of a single individual frequently coexists inside a population). Going further in this direction, we may also consider the whole information produced along the evolution: it often happens that an individual is a copy – or a slightly disturbed copy – of a "dead" ancestor. As we will see below, keeping track of all evaluations performed along the evolution provide another way to reduce the noise of the fitness function.

Moreover, if we can use a metric on the search space that makes sense (i.e. on which we can define a regularity property such as: two individuals that are similar with respect to this metric have similar fitness values), the previous idea may be extended. This implies that we assume some regularity properties of the underlying signal. This is a common hypothesis for many "denoising" techniques in signal analysis [6]. Fitness evaluations may be then averaged for individuals that lie in a given neighbourhood (with appropriate weights, related to the fitness regularity assumption). The resulting computation time overload for the EA remains negligible in the case of time consuming fitness.

2.2 An implementation for real-coded genomes.

Sano and Kita [10] proposed to use the history of search to refine the estimated fitness values of an individual, using the fitness evaluations of individuals similar to it. Their approach is based on a stochastic model of the fitness function that allows to use a maximum likelihood technique for the estimation of the underlying fitness function.

Here we make the assumption that the underlying fitness function is regular with respect to the search space metric. Let us first define:

- The search domain:

$$S = \prod_{i=1}^{m} [a_i, b_i], \text{ with } \forall i \in \{1, ..., m\} \ (a_i, b_i) \in \mathbb{R}^2 \text{ and } \mathrm{a_i} < \mathrm{b_i} \qquad (1)$$

- A *max* distance on S:

$$\forall x, y \in S, \quad d_\infty(x, y) = max_{i \in \{1,..,m\}} \left(\frac{|x_i - y_i|}{b_i - a_i} \right) \qquad (2)$$

  The divider $(b_i - a_i)$ ensures that each component of a vector has the same weight in the distance regardless of the extent of its domain.

– An euclidian distance on S:

$$\forall x, y \in S, \quad d_2(x,y) = \sqrt{\sum_{i=1}^{m} \left(\frac{x_i - y_i}{b_i - a_i} \right)^2} \tag{3}$$

– The neighbourhood of a point is defined using the max distance:

$$\forall x \in S,\ \sigma_\infty \in \mathbb{R}_+^*, \quad \mathrm{B}_{\sigma_\infty}(\mathrm{x}) = \ \{\mathrm{y} \in \mathrm{S_g};\ \mathrm{d}_\infty(\mathrm{x},\mathrm{y}) \leq \sigma_\infty\} \tag{4}$$

We now define the regularity of a fitness as the fact that the fitness values of individuals belonging to the neighbourhood of an individual x (being in $B_{\sigma_\infty}(x)$), are also close to the fitness value $f(x)$. Hölder regularity is a well-fitted tool for this purpose:

Definition 1 (Hölder function of exponent h).
Let (X, d_x) and (Y, d_Y) two metric spaces. A function $F : X \to Y$ is called a Hölder function of exponent $h > 0$ and constant k, *if for each $x, y \in X$ such that $d_X(x,y) < 1$, we have:*

$$d_Y\left(F(x), F(y)\right) \leq k \times d_X(x,y)^h \tag{5}$$

Although a Hölder function is always continuous, it needs not be differentiable. Intuitively, a Hölder function with a low value of h looks much more irregular than a Hölder function with a high value of h (in fact this statement only make sense if we consider the highest value of h for which (5) holds). The majority of fitness function on real search space is Hölder.

We now want to keep track of the points of S (y_i) that have been evaluated at least one time. Of course the same point may have been evaluated more than one time, so we have to consider the number of evaluation ($inst(y_i)$) and the average of these evaluations ($\tilde{f}(y_i)$). We can then define the following set:

$$\bar{\Pi}_t = \left\{ \left(y_i, inst(y_i), \tilde{f}(y_i) \right), i \in \{1, ...n_{\bar{\Pi}}\} \right\}$$

The index t denotes the number of fitness evaluations that have been taken into account for the construction of $\bar{\Pi}_t$. It just emphasises that $\bar{\Pi}_t$ can be considered as a *genetic database*, that is continuously updated along the evolution, i.e. when pairs (individual, fitness evaluation) are computed. However, for clarity we will later drop the t subscript.

Using $\bar{\Pi}$ we can now define a weighted fitness function:

$$\forall x \in S, \quad g_{\bar{\Pi}}(x) = \frac{\sum_{y \in \bar{\Pi} \cap B_{\sigma_\infty}(x)} w(x,y) \times inst(y) \times \tilde{f}(y)}{\sum_{y \in \bar{\Pi} \cap B_{\sigma_\infty}(x)} w(x,y) \times inst(y)}$$

Where the weight $w(x,y)$ is defined according to the euclidian distance:

$$w(x,y) = \left(1 - \frac{d_2(x,y)}{\sqrt{m} \times \sigma_\infty} \right)$$

We have $w(x,x) = 1$, and as :

$$max(d_2(x,y),\ y \in B_{\sigma_\infty}(x)) = \sqrt{m} \times \sigma_\infty \text{ and } d_\infty(.,.) \geq d_2(.,.)$$

$w(x,y)$ is always non negative: $\forall x \in S,\ \forall y \in B_{\sigma_\infty}(x),\ w(x,y) \geq 0$

$\bar{\Pi}$ can now be used in the following way. Each time that an individual x has been evaluated, its "raw" (not yet averaged) fitness score is used to update the database. The weighted fitness score can be returned with the computation of $g_{\bar{\Pi}}(x)$. The accuracy of the weighted fitness $g_{\bar{\Pi}}$ depends greatly on the regularity assumption on the fitness function. The parameter σ_∞ is directly related to the regularity of the underlying fitness function (i.e. to k and h), and in the case of an extremely irregular function (i.e. having discontinuities or h near 0), we have to set $\sigma_\infty = 0$.

3 Classical fixed size population versus growing population ?

The idea of using historical information has also been developed for multiobjective optimisation by Corn and al [2] and Zitzler and al [11]. Their approach consists in building an "archive" of non-dominated individuals to maintain diversity, that is updated at each generation.

We propose to build a genetic database, as a simple cumulation of all produced individuals. It can be used directly in a real-coded GA, for example, with the following procedure:

1 : Evaluate each individual of the current population.
2 : Add each individual with its raw fitness score to the database.
3 : Compute weighted fitness scores of all individuals with the help of the database.
4 : Apply your favorites selection schemes, genetic operators, replacement schemes, and loop on step 1 until termination.

Moreover this structure may be used to modify the classical birth and death cycle of an EA. More precisely the individuals to be reproduced can be directly selected in this genetic database. This can be seen as a *growing population* of immortal individuals. To maintain diversity, a simple tournament selection seems then appropriate: choose randomly n_t (if n_t is the size of the tournament) individuals in $\bar{\Pi}$ and keep the one having the best weighted fitness. Any individual of the genetic database may thus have offspring at any time. Thereby the information of the whole evolution is not only used to produce more accurate fitness evaluations but offers a simple way to maintain diversity. We should also emphasize the asynchronous aspect of this algorithm, i.e. we do not have to wait the whole current population to be evaluated in order to perform selection, but at any time we are able to choose from all already evaluated individuals. It is adapted to distributed implementation, for example with a client-server model: a genetic server deserves clients that perform the fitness evaluations. The server can manage the database with the following principles:

- A pool of random offsprings is initially created.
- For any client request, the server supplies an offspring from its pool until this one is empty.
- As soon as a client has finished the evaluation of its current individual, it is returned to the server that adds the information to the database.
- When the offspring pool is empty the server creates new individuals to fill it again. This creation is made by selecting parents from the database (with a tournament for example) and applying genetic operators.
- In order to have a minimum initial diversity, we impose that when the server creates new individuals a minimum number of individuals (call it min_{par}) has to be present in the database before selection can be applied. If this condition is not fullfilled, offsprings are generated randomly until the database is sufficiently large.

4 Sharing

In order to maintain diversity it also seems convenient to use a sharing procedure [4]. We propose the following one, linked to the weighted averaging procedure in a simple way: each time we compute $g_{\bar{\Pi}}(x)$ the following quantity can be computed with few extra computation:

$$\forall x \in S, \quad W_{\bar{\Pi}}(x) = \sum_{y \in \bar{\Pi} \cap B_{\sigma_\infty}(x)} w(x,y) \times inst(y) \tag{6}$$

This can be seen as a neighbour count which is used in the shared fitness function:

$$\forall x \in S, \text{ such as } W_{\bar{\Pi}}(x) \neq 0, \quad h_{\bar{\Pi}}(x) = g_{\bar{\Pi}}(x) \times \left(1 + \frac{1}{W_{\bar{\Pi}}(x)}\right) \tag{7}$$

As for each evaluated point, we have $W_{\bar{\Pi}}(x) \geq 1$, a tournament based on $h_{\bar{\Pi}}(x)$ can be used. The effect of this sharing will be that for an individual without neighbours and evaluated once, we will have $W_{\bar{\Pi}}(x) = 1$ and therefore $h_{\bar{\Pi}}(x) = g_{\bar{\Pi}}(x) \times 2$. On the contrary for an individual having many neighbours we will have $h_{\bar{\Pi}}(x) \simeq g_{\bar{\Pi}}(x)$. As a consequence, isolated individuals will be given a higher probability to be selected than surrounded ones.

5 Experimental procedure.

5.1 Algorithms an genetic operators.

The following EA are compared:

- a **GA** without use of the fitness weighted averaging.
- a GA with fitness weighted averaging (further denoted **GAW**).
- our immortal evolutionary algorithm (**IEA**), with tournament.
- **IEA + sharing**.

Individuals are encoded as real vectors, the search is represented with equation (1). The euclidian distance is used as a metric on this space.

The classical generational GA will use the stochastic universal selection (SUS, see [1]) with a full replacement of population.

The classical **gaussian mutation** will be used for each component with a fixed variance $\sigma_i = 0.1 \times (b_i - a_i)$. We did not experiment here the adaptive σ method, which is commonly used in Evolutionary Strategies.

Regarding **crossover**, we will test 3 configurations:

1 : Classical arithmetic crossover. If we denote (x, y) the parents, (x', y') the offspring, and γ a random uniform number from $[0, 1]$:

$$\begin{cases} x' = \gamma x + (1 - \gamma) y \\ y' = (1 - \gamma) x + \gamma y \end{cases} \tag{8}$$

2 : Arithmetic crossover with mating restriction (that will be further called mating crossover). Only parents that are close enough are allowed to mate. Practically, if m denotes the dimension of the search space, an euclidian distance of $(0.1 \times \sqrt{m})$ will be the threshold.

3 : No crossover.

5.2 Common characteristics of experiments.

- All measurements are averaged on **25 runs**, for a given configuration.
- A limited number of **3200** fitness evaluations is fixed, i.e. for GA, runs are a **100** generations with population of size **32**.
- The unperturbed fitness scores are also kept off-line in order to evaluate the accuracy of the algorithms.
- IEA parameters:
  - tournament size $n_t = 4$
  - number of initial random individual $min_{par} = 32$.
- Each time that measures are computed on the population of a GA (average fitness scores for example), similar measures are taken for the IEA by grouping new individuals in sets of the same size as the population size. Note that this is done only for measurement purpose.
- When crossover is used, the following probabilities are tested:
  - **crossover probability** $p_c = \{0.2, 0.5, 0.8\}$.
  - **mutation probability** $p_m = \{0, 0.01, 0.1\}$.
  - In the case of **mutation alone** we set $p_m = \{0.02, 0.05, 0.1, 0.2\}$.

We must outline that all measures of all run are not reported in this paper, as it would require a large amount of figures. We therefore tried to choose the most significant ones to be discussed here, a complete report of these tests is available in [8].

6 f_1: A multimodal function

6.1 Definition

We consider the following function:

$$\begin{array}{rcl} F_1 : [0,1]^3 : & \rightarrow & \mathbb{R}^+ \\ (x_0, x_1, x_3) & \rightarrow & \left(\sum_{i=0}^{2} t(x_i)\right) \end{array} \tag{9}$$

A gaussian noise is added to obtain the noisy fitness function:

$$f_1 = F_1 \times (1 + N_{(0,0.5)}) \tag{10}$$

With t (see figure 1, left):

$$\begin{array}{rcl} [0,1] : & \rightarrow & [0,1] \\ x & \rightarrow & \begin{cases} (4 * x) & \text{if } x \in [0, 0.25[\\ (2 - 4 * x) & \text{if } x \in [0.25, 0.5[\\ (4 * x - 2) & \text{if } x \in [0.5, 0.75[\\ (4 - 4 * x) & \text{if } x \in [0.75, 1] \end{cases} \end{array} \tag{11}$$

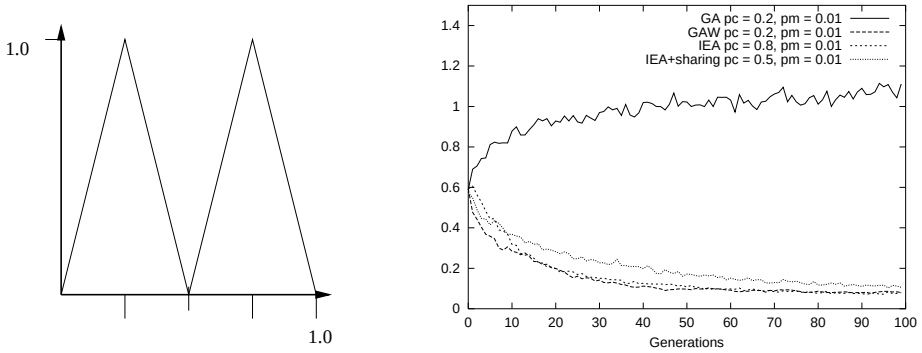

Fig. 1. Left: Function t. Right: average noise on f_1 (on 25 runs).

As t has two optima (at $\frac{1}{4}$ and $\frac{3}{4}$) of the same height, F_1 will have 8 optima of the same height. In order to measure the ability of algorithms to locate many optima, we will count the number of individuals that fall in their vicinity. Precisely, for each optimum $(o_{i,i\in\{0,\ldots,7\}})$, we count the number of individuals that verify $d_\infty(o_i, x) < 0.1$, obtaining the list of optimum neighbouring counts $(no_{i,i\in\{0,\ldots,7\}})$. If we now sort them in decreasing order, it becomes possible to compute averages over different runs.

Finally, as its clear that F_1 is regular ($h = 1$) we set $\sigma_\infty = 0.05$.

6.2 Results on F_1

Figure 1 (right) shows the average noise of fitness evaluations. For the GA without fitness weighted averaging, the quantity plotted is simply $\sum_{x \in population} |f_1(x) - F_1(x)|$, and for the other algorithms $\sum_{x \in population} |g_{\bar{\Pi}}(x) - F_1(x)|$. We remind that in the case of the IEA, the term population denotes only the grouping of the last new individuals for comparison purpose. We clearly see that, in absence of weighted averaging the noise increases as the average fitness increases because of the multiplicative nature of noise. When using the weighted averaging procedure noise decreases significantly.

Figures 2 to 4 show the performance in terms of Average Denoised Fitness (further called **ADF**), corresponding to F_1. The ability to locate optima is measured by the optimum neighbouring counts $(no_{i,i\in\{0,\ldots,7\}})$[1] , for the three configurations (arithmetic crossover, mating crossover and mutation alone).

Note that in the case of arithmetic crossover, the classical GA with a low crossover probability ($p_c = 0.2$) leads to the highest ADF scores, but concentrate rather on a single optimum. The effect of weighted averaging does not change much the results. The IEA obtains lower performances in terms of ADF, but obtains a better diversity of optima when used with sharing.

In the case of mating crossover, the GA and the GAW performs better when p_c is set to 0.8. It must be outlined that the effective crossover ratio is lower, due to the mating restriction rule. But we see on the IEA runs that it provides good results in terms of ADF and in the same time in terms of optima diversity (especially when sharing is used).

The application of mutation alone reveals to be quite efficient at high rate ($p_m = 0.2$) when using a GA, but less interesting for the IEA.

In conclusion for this test function, we can see that a simple GA can efficiently provide a good average fitness, but that the IEA + crossover with mating restriction covers more efficiently the different optima.

7 F_2: An epistatic version of F_1

7.1 Definition

Consider the following function:

$$\begin{array}{rcl} F_2 : [0,1]^3 : & \rightarrow & \mathbb{R}^+ \\ (x_0, x_1, x_3) & \rightarrow & t_2(x_0, x_1) + t_2(x_1, x_2) + t_2(x_2, x_0) \end{array} \tag{12}$$

with t_2 being defined with the help of function t (see section 6):

$$t_2(x, y) = \begin{cases} t(x) & \text{if} \quad (x - 0.5) \times (y - 0.5) > 0 \\ 0.5 \times t(x) & \text{if} \quad (x - 0.5) \times (y - 0.5) < 0 \end{cases} \tag{13}$$

[1] As all graphs do not have exactly the same ordinate scale, we have drawn a line at $(y = 10)$ for visual comparisons.

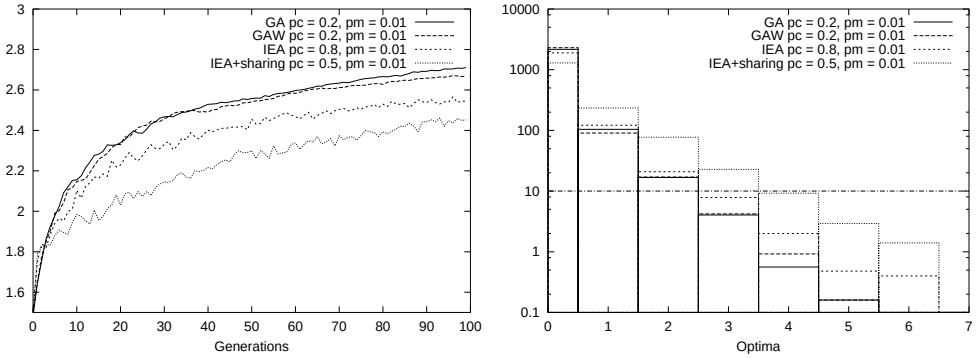

Fig. 2. F_1: Arithmetic crossover. Left: average denoised fitness values. Right: optimum neighbouring counts.

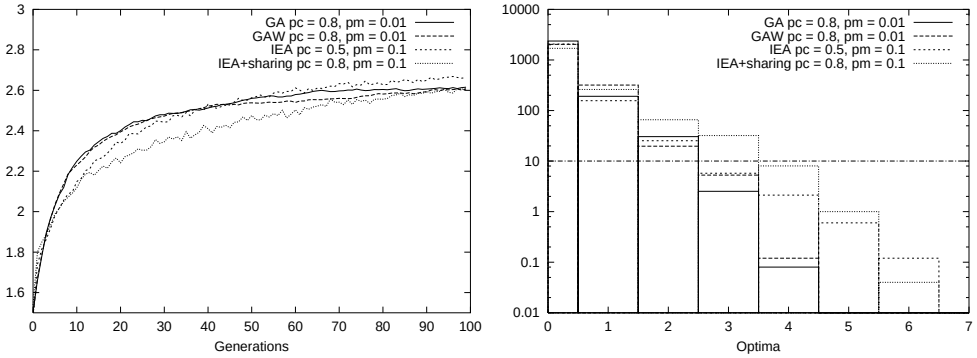

Fig. 3. F_1: Mating crossover. Left: average denoised fitness values. Right: optimum neighbouring counts.

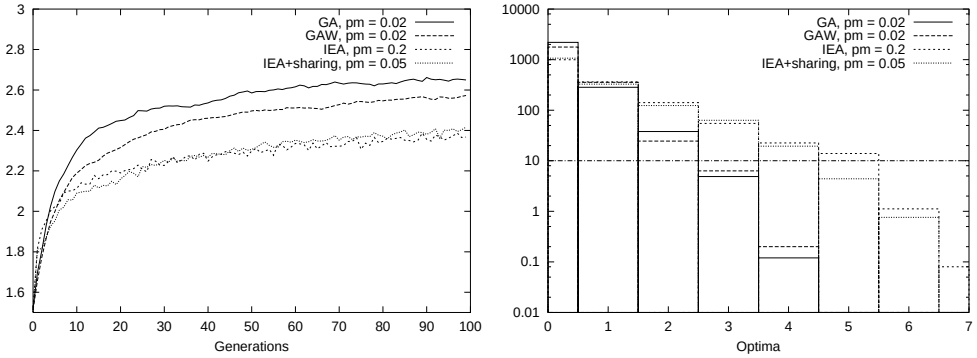

Fig. 4. F_1: mutation alone. Left: average denoised fitness values. Right: optimum neighbouring counts.

The noisy fitness is:

$$f_2 = F_2 \times (1 + N_{(0,0.5)}) \tag{14}$$

As F_2 is also a regular function ($h = 1$), we set $\sigma_\infty = 0.05$.
In comparison to F_1, F_2 also has 8 optima. The difference comes from the epistatic form of t_2. In fact there are two main optima at $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ and $(\frac{3}{4}, \frac{3}{4}, \frac{3}{4})$ with a value of 3, and 6 secondary optima with a value of 2 with all other combinations of $\frac{1}{4}$ and $\frac{3}{4}$. It is clear that the two main optima are in some sense opposite and are separated by secondary misleading optima. The goal of the algorithms will then be to explore at least one main optimum without being too much puzzled by secondary optima, and eventually cover both main optima. The optimum neighbouring counts are therefore slightly modified the following way: $(no_{i,i\in\{0,1\}})$ represents neighbouring counts of the main optima (also sorted in order to compute an average over runs) and $(no_{i,i\in\{2,..,7\}})$ will be neighbouring counts of the secondary optima.

7.2 Results on F_2.

Figures 5 to 7 show once again that a simple GA is able to find a main optimum and exploit it but often fails to find the other one, and is rather puzzled by secondary optima. Weighted fitness brings a first improvement, but IEA seems to performs better, again in combination with mating crossover.

8 Other tests.

Other tests were performed, they are reported in [8]:

- prescribed regularity functions (Weierstrass-Mandelbrot functions).
- a molecular simulation application (see [7] for first results).

9 Conclusion.

We propose in this paper a new use of history in evolutionary computation, adapted to computationally heavy and noisy fitness functions. An increased complexity for the EA allows to reduce the number of CPU-time consuming fitness evaluations.

Moreover we experimentally show that, when the function is sufficiently regular in respect to a metric on the search space, it is possible to use similarity of individuals to reduce noise successfully without additional fitness evaluations. We propose a new algorithm using the whole history of evolution to generate new offspring. Experiments with a limited number of fitness evaluations on real-coded test functions show that, if GA can overcome the effect of noise in finding good regions, our immortal evolutionary algorithm (IEA) maintains a better diversity. Of course, we cannot draw firm conclusions on the basis of two test-functions, but these experiments show in which way we can tune the balance

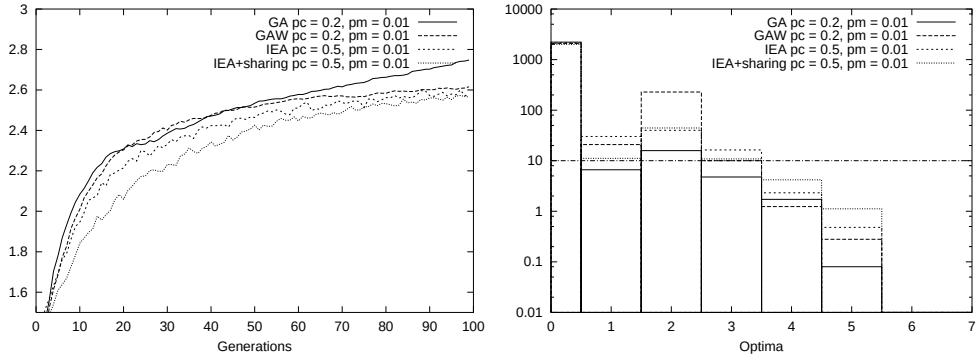

Fig. 5. F_2: Arithmetic crossover. Left: average denoised fitness values. Right: optimum neighbouring counts.

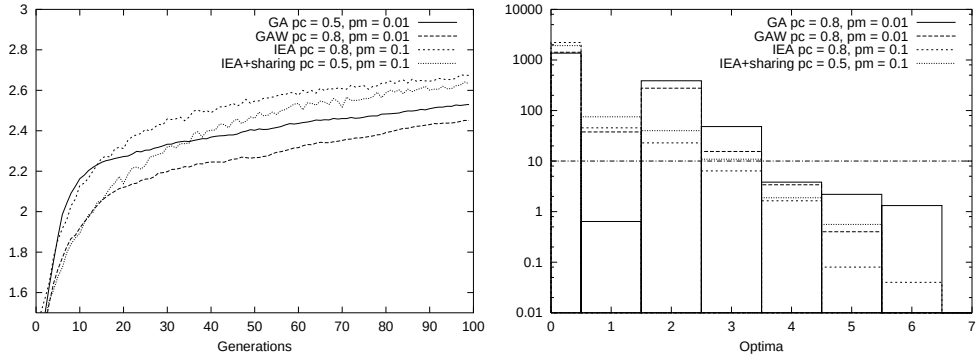

Fig. 6. F_2: Mating crossover. Left: average denoised fitness values. Right: optimum neighbouring counts.

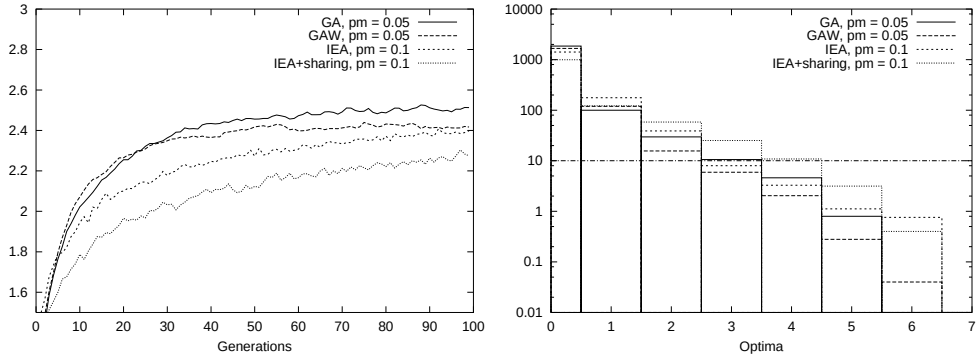

Fig. 7. F_2: mutation alone. Left: average denoised fitness values. Right: optimum neighbouring counts.

between exploration (discovery of many optima) and exploitation (good average fitness of the population), by balancing crossover, mutation and sharing methods. The problem of an automatic adaptation of σ_∞ along the evolution will be considered as future work. For that purpose an on-the-fly estimation of the regularity of the fitness function could be used.

References

1. Baker, J.E.: "Reducing bias and inefficiency in the selection algorithm" in Genetic Algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms, 14-21, **1987**.
2. Korne, D.W., Knowles, J.D., Oates, M.J.: "The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization", in Proceedings of *Parallel Problem Solving from Nature 6*, (pp 571-580), **2000**.
3. Fitzpatrick, J.M., Grefenstette, J.J.: "Genetic Algorithms in noisy environments." in P. Langley, editor, *Machine Learning*, pages 101-120, (Kluwer, Dordrecht, **1988**).
4. Goldberg, D.E., Richardson, J.: "Genetic algorithms with sharing for multimodal function optimization." in J.J. Grefenstette, editor, *Genetic Algorithms and their Applications*, (pp 41-49), Lawrence Erlbaum Associates, Hillsdale, New-Jersey, **1987**.
5. Hammel U., Bäck, T.: "Evolution Strategies on Noisy Functions. How to improve Convergence Properties." in Y. Davidor, R. Männer, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature 3*, pages 159-168, (Springer Verlag, Heidelberg, **1994**).
6. Lévy Véhel, J. and Lutton, E.: "Evolutionary signal enhancement based on Hölder regularity analysis", in Proceedings of EVOIASP2001 Workshop, Como Lake, Italy, Springer Verlag, LNCS 2038, **2001**.
7. Leblanc, B., Lutton, E., Braunschweig, B., Toulhoat, H.: "Improving molecular simulation: a meta optimisation of Monte Carlo parameters", in Proceeding of *CEC2001, Congress on Evolutionary Computation*, **2001**.
8. Leblanc, B., Lutton, E., Braunschweig, B., Toulhoat, H.: "History and never ending life in evolutionary computation: a molecular simulation application", INRIA Research Report, to appear, **2001**.
9. Miller, B.L.: "Noise, sampling, and genetic algorithms", Doctoral dissertation, Illigal Report No. 97001, **1997**.
10. Sano, Y., Kita, H.: "Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search", in Proceedings of *Parallel Problem Solving from Nature 6*, (pp 571-580), **2000**.
11. Zitzler, E. and Thiele L.: "Multiobjective Evolutionary Algorithm: A Comparative Case Study and the Strength Pareto Approach ", in *IEEE Transactions on Evolutionary Computation, 2(4)*, (pp 257-272), **1999**.