# Evolving From Genetic Algorithms to Flexible Evolution Agents

**Gabriel Winter**
Institute of Intelligent Systems & Numerical Applications in Engineering (IUSIANI).
Evolutionary Computation Division (CEANI).
Las Palmas de G. C. University
Canary Islands, Spain
gabw@step.es

**Blas Galván**
IUSIANI-CEANI
bgalvan@step.es

**Silvia Alonso**
IUSIANI-CEANI
oraami@yahoo.es

**Begoña González**
IUSIANI-CEANI
bgonzalez@iusiani.ulpgc.es

## Abstract

In this paper an alternative design for the internal structure of Evolutionary Algorithms is presented and a first implementation of such structure is tested. In order to improve the internal structure architecture of Evolutionary Algorithms as well as to learn which the adequate values of parameters and/or structure of operators are, a review showing the successive efforts made in this line is included. In this way, the evolution from the simplest GA to the present advanced structures is illustrated using figures which summarize their characteristics. Based on the analysis of such figures, a new and more flexible internal structure design is introduced maintaining the benefits of its predecessors but permitting a major degree of self-adaptation, because now the parameters, the operators and the structure can adapt themselves at each optimization step. This new structure is named a Flexible Evolution Agent (FEA). Results of a first software implementation are presented and compared using several well-known test functions.

## 1 INTRODUCTION

From the beginning in the Genetic Algorithms (GAs) research it has been clear the necessity to choose the best parameters and operators that would be used during every optimization process. The first attempts in this line were aimed at the search of global parameters that could be used in all problems, as the accomplished by De Jong (1975) or more recently, by Schaffer *et al* (1989). However, almost immediately it was demonstrated that far away that existing parameters valid for all cases, every particular problem needs to have its parameters adjusted to its own requirements. So, GAs have had to solve this misadventure introducing the possibility of adapting these parameters (or even the operators) needed in the optimization process.

Many authors have deepened in this line of research, and nowadays, establishing the best way of adjusting the parameters and/or operators and correcting the unfeasible solutions constitute the main present trends. Nevertheless, some researchers also distinguish between parameters *adaptation* and *self-adaptation*, as Eiben *et al* (1999) widely explain in their exhaustive state-of–the-art. As a sample of parameters adaptation, Davis (1989) proposed a schema that adjusted the operators' rates depending on their success in creating good individuals. By contrast, self-adaptation commonly refers to any of the Evolutionary Strategies (ESs) approaches, mainly developed by authors as Bäck and Schwefel (1991). ESs present an alternative for self-adapting the parameters during the optimization process, in such a way that no pre-established settings are needed. Nonetheless, in the specialized literature generally the terms adaptation and self-adaptation are equally considered.

Regarding adaptation in general, most of the works have focused in parameters adaptation, and only a few, (as the raised by Igel *et al*, 2001) have been centered in adapting operators. Normally, adapting operators involve setting the parameters either. Hence, only parameters adaptation is usually revised. The majority of works have consisted in adapting simply one numerical parameter, as might be mutation rate (Matsui *et al*, 2001) or fitness adaptation (Wright and Agapie, 2001) whereas few studies have been done regarding other important aspects of the implementation as could be the way of representing the search space (Schraudolph and Belew, 1992, with their Dynamical Parameter Encoding).

One of the most used approaches when self-adapting parameters is the model given by other big branch in the GA research: the use of Meta-GAs (Grefenstette, 1986). These algorithms were first proposed by Weinberg (1970) and consisted in two GA levels: the first level is a kind of 'training stage' where a search of the best suited

parameters or operators was carried out. Then, the second level is a simple GA where the adjusted values enter as preset parameters. Anyway, the algorithm's structure is fixed as the preceding ones; the difference is that the parameter enters in an initial GA to be self-adapted to the specific problem. The selection of the best adapted solutions can be made evolving different populations independently, using rules or any feasible mechanism.

One step further is the research made considering adaptation of both parameters and operators at the same time (Hinterding *et al*, 1996), combining for instance mutation rates and population sizing adjustment, or even more, making evolve the genetic operators in an independent population (Edmonds, 1998). Other authors (Darwen, 2000) have remarked the inevitability of implementing a parameterless algorithm, despite of the high computational costs that a mechanism to adapt ALL the parameters should have.

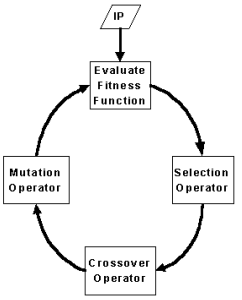
An alternative way has been chosen by other researchers: to make the algorithm itself self-adaptive (Krasnogor *et al*, 2001) .This approach and similar ones are the actual tendencies in the GAs and ESs fields. As the use of some kind of memory in these algorithms has proven to be really fruitful, to implement mechanisms that can learn and memorize the previous experience gained in the optimization process is a benchmark problem. Among them, the use of several procedures such as classifier systems (Lanzi, 2001) or agents (Bot *et al*, 2001) has demonstrated to be very successful.

Following these rules of thumb, a more flexible mechanism, able not only of self adapting the parameters and the operators involved in the search, but also the own structure of the algorithm, the Flexible Evolution Agent (FEA), is introduced in this work. First of all, several schemes showing the essential evolution from the other approaches to the presented implementation are depicted. Secondly, the FEA structure is developed, as well as an analysis of the reasons of its specific construction. Then, a section of the results attained using one first FEA implementation over several mathematical test functions is reviewed. Finally, some conclusions and the future lines of work are described.

## 2 FROM A SIMPLE GA TO A FEA

In order to show why a flexible algorithm such as the FEA is so necessary, several schemas showing the evolution of the different approaches from Genetic Algorithms to a flexible method are described. As several and more complicated schemes are represented next, the flux diagram used is not an ordinary one due to the necessity of including more complicated mechanisms afterwards hardly explained of any other way. A simple algorithmic implementation accompanies each figure, depicting basically every approach.

As developed in the previous section, the current GAs do not allow the variation of their own structure of operators during the optimization process. For instance, a simple GA uses repeated trials of the Selection-Crossover-Mutation sequence of fixed operators. Thus, they have a fixed structure as can be seen in Figure 1.

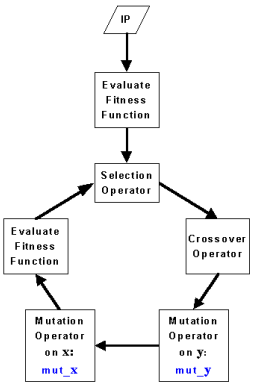



Let $P^t(\mathbf{x})$ be the population where **x** denotes the individuals. Then the GA algorithm is:

- Establish parameters of the GA: population size, maximum number of generations, selection operator, crossover operator, crossover rate, mutation operator, mutation rate;
- t = 0;
- Generate $P^t(\mathbf{x})$;
- **While not** termination criterion **do**
  - Evaluate $P^t(\mathbf{x})$;
  - Set $P^{t'}(\mathbf{x}) \leftarrow$ selection $[P^t(\mathbf{x})]$;
  - Set $P^t_{new}(\mathbf{x}) \leftarrow$ crossover $[P^{t'}(\mathbf{x})]$;
  - Set $P^{t+1}(\mathbf{x}) \leftarrow$ mutation $[P^t_{new}(\mathbf{x})]$;
  - Set $P^t(\mathbf{x}) \leftarrow P^{t+1}(\mathbf{x})$;
- **Od**
- Print solutions;

Figure 1: A simple GA: Structure and implementation.

In the same line than the referred Meta-GAs, but including the learning stage inside the algorithm, the next step is to consider a GA that self-adapts its mutation rates as the described in Figure 2 (following the Matsui (2001) example). The diagram is a bit more complex, but still does not show how the algorithm really uses the memorized information. Now, the solutions include information not only concerning the individual, but also about the mutation rates associated to the individuals.



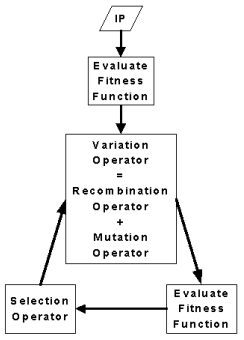

Let $P^t(\mathbf{W})$ be the population where $\mathbf{W} = (\mathbf{x}, \mathbf{y})$ denotes the individuals being **x** the variables vector and **y** the mutation rates vector. Then the algorithm will be:

- Establish parameters of the GA: population size, maximum number of generations, selection operator, crossover operator, crossover rate, mutation operator;
- t = 0;
- Generate $P^t(\mathbf{W})$;
- Evaluate $P^t(\mathbf{W})$;
- **While not** termination criterion **do**
  - Set $P^{t'}(\mathbf{W}) \leftarrow$ selection $[P^t(\mathbf{W})]$;
  - Set $P^t_{new}(\mathbf{W}) \leftarrow$ crossover $[P^{t'}(\mathbf{W})]$;
  - Set $P^{t+1}(\mathbf{W}) \leftarrow (\text{mut\_x} \circ \text{mut\_y})[P^t_{new}(\mathbf{W})]$;
  - Set $P^t(\mathbf{W}) \leftarrow P^{t+1}(\mathbf{W})$ ;
  - Evaluate $P^t(\mathbf{W})$;
- **Od**
- Print solutions;

Figure 2: A GA that self-adapts its mutation rates: Structure and implementation.

A qualitative jump is made when an Evolutionary Strategy is taken into account. A simple implementation such as the classic (μ,?)-strategy (Bäck *et al*, 1991) is described in Figure 3. Again, the learning mechanisms are

included in the implementation, but following the ESs model where is in the variation step when the covariances matrices are considered. The solutions in this case are enlarged with information about the standard deviations and the rotation angles, but the structure of the algorithm is still essentially fixed.




Let $P(\mathbf{W})$ be the population where $\mathbf{W} = (\mathbf{x}, \sigma, \alpha)$ denotes the individuals being $\mathbf{x}$ the variables, $\sigma$ the standard deviations and $\alpha$ rotation angles. Then the EE algorithm is:

- Establish parameters of the EE: population size, maximum number of generations, selection operator, recombination operator, mutation operator;
- t = 0;
- Generate $P(\mathbf{W})$;
- Evaluate $P(\mathbf{W})$;
- **While not** termination criterion **do**
  - Set $P'(\mathbf{W}) \leftarrow \text{variation}[P^t(\mathbf{W})]$;
  - Evaluate $P'(\mathbf{W})$;
  - Set $P^{t+1}(\mathbf{W}) \leftarrow \text{selection}[P'(\mathbf{W})]$;
  - Set $P^t(\mathbf{W}) \leftarrow P^{t+1}(\mathbf{W})$;
- **Od**
- Print solutions;

Figure 3: A simple ES: Structure and implementation.

Analyzing all these algorithms, they only are flexible in the way that they increasingly store more information and use it in several ways, but they still depend on the problem to be solved. The scheme included in Figure 4 shows the intended objective in the optimization methods tried to fulfill with the FEA, the new method that is proposed in the next section

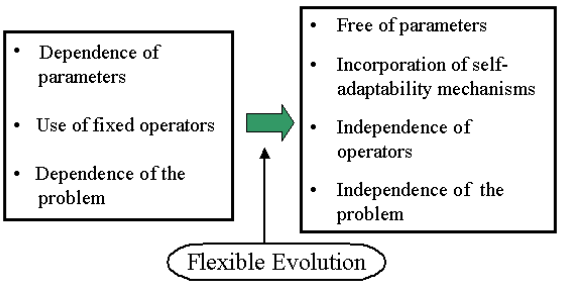



Figure 4: Desirable Evolution of the Global Optimization Methods

## 3 DESCRIPTION OF THE PROPOSED FEA STRUCTURE

The FEA alternative pretends to store up all the possible advantages from the last trends in the Evolutionary Programming field as well as introducing new options from other research branches. The initial premise is that, in principle, any mechanism that has proven to be fruitful in all related areas can help to develop a new and of growing difficulty implementation, as robust and efficient as it could possibly be. The intention is to generate a more flexible algorithm that can get the biggest benefit of the use of stored information, as this is one of the keys of all the already mentioned successful approaches, and in addition, to incorporate new procedures to facilitate the internal decisions making along the optimization process. This yields to that the FEA posses an Enlarged Genetic Code (EGC), in such a way that a solution contains information not only about the variables of the individuals, but also about the different parts that the FEA is divided in as it is explained later. Moreover, in order to obtain a higher flexibility in the implementation, FEA has a Dynamic Structure of Operators (DSO) allowing self-adaptation of the operators sequence to the optimization conditions along the process. Finally, with the aim of controlling effectively all the procedures involved in the search, some Artificial Intelligence based methods can be used as members of the Decision Engine.

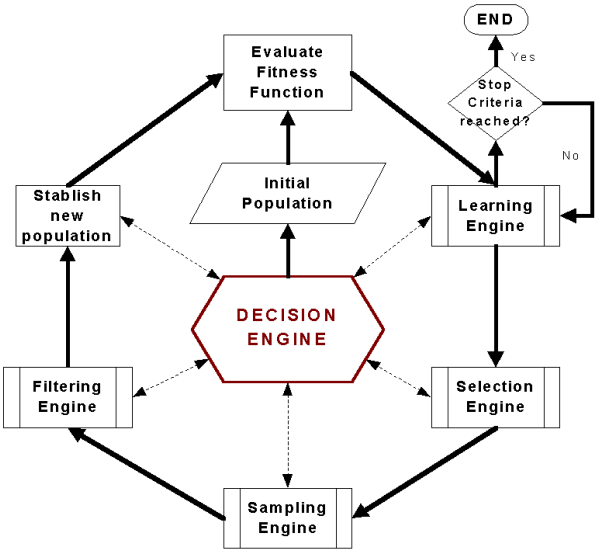



Figure 5: Scheme of a Flexible Evolution Agent

In order to guarantee the agent's flexibility, the algorithm has been implemented so that the different tasks that it executes can be coordinated. As a consequence, the FEA has been subdivided into several functions, called '*engines*'. These subroutines are designed to group the diverse actions that are to be executed during the optimization depending on their objectives. In this way, all the learning tasks are clustered in a 'Learning engine', and something similar happens with all the selection

schemes, sampling strategies or decision mechanisms. The most important engine is the situated in the center of Figure 5: the Decision Engine, whose mission is to take all the necessary decisions in order to initiate, develop and finish the optimization process. Its main functions are (for every stage of the process) to decide about: the population size, each individual's structure (its genotypic information), the learning mechanisms to be considered, the stop criteria to be used, the selection method to be applied, the sampling method/methods to be utilized (and their adequate parameters), the filtering processes (if necessary) and the criteria to generate new populations (including their sizes). It is important that the Decision Engine will be able of deciding if its own behavior is correct or not, and in this case that it could vary their criteria about how it manages the optimization in order to get a more successful process. The internal structure of this engine may be able of taking all the precedent decisions along the process, using advanced methods such as rule based Expert Systems, Fuzzy Logic, Probabilistic Methods or any other coming from the Artificial Intelligence area. The Decision Engine, however, must receive general instructions from the user to establish guidelines or an ordered set containing the strategies and tactics that it has to take into account to rule over the optimization process, as well as the data and constraints of the problem to be solved.

Regarding the rest of engines, a short explanation is included. The Learning Engine stores everything that could be useful afterwards, such as information about the variables, statistics and successful stories of the rest of engines. The Selection Engine chooses which solutions are to be sampled whereas the Sampling Engine selects the sampling method that is used and carries out the variation processes over the variables as explained in the next section. The Filtering Engine is in charge of debugging the possible errors that it may detect in the solutions that reach that point. Then, the EGC introduced before might tackle with individuals with the following general structure $P(\mathbf{W}) = P(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}, \mathbf{h}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \ldots)$, where $P$ denotes the population of $\mathbf{W}$ individuals, $\mathbf{x}$ the variables and the rest are information relative to the engines ($\mathbf{y}$ about the learning, $\mathbf{z}$ about selection, $\mathbf{w}$ about the sampling, $\mathbf{h}$ about the filtering, $\mathbf{f}$ and $\mathbf{g}$ about the decisions) or other data as $\mathbf{d}$ for the new population and $\mathbf{e}$ for the objective function evaluation. As can be seen, this chromosome is much more complicated than the current utilized ones. The intention is to use this information to learn about the process and even to establish rules that could be fruitful and may be included in the Decision Engine.

## 4 A FIRST FEA IMPLEMENTATION

As the proposed general structure is a lot more complicated algorithmically speaking than the existent ones, we have opted for exploring the possibilities it offers developing a first version whose implementation incorporates only some of the elements of the general FEA scheme. In this implementation, the Decision Engine consists in a simple Probabilistic Control Mechanism (PCM) based on the use of probabilistic frontiers for all decisions to be taken. The general PCM design is [1]:

- Set *decvar* = *U*(0,1)
- If (*decvar* ≤ *alpha*) Then
  Adopt a conservative decision
- Else
  Adopt a non-conservative decision

Depending on which engine the PCM is deciding on, the probabilistic frontier *alpha* and the decisions to be taken can be different. In general a conservative decision means to adopt a successful one taken in the near past and non-conservative means to choose any other available decision for the engine. After several tests the value 0.6 has been elected for *alpha.*

The selected Learning Engine extracts a subset from the actual population containing the best individuals and stores information about the sampling methods used during the optimization, so the EGC contains two vectors as depicted afterwards. The Selection Engine establishes which solutions are to be sampled by the Sampling Engine by means of Tournament Selection (2:1). The Sampling Engine is the first function that is completely designed and explained in the next subsection. Any Filtering Engine has been incorporated yet.

Let be $P^t(\mathbf{W})$ the population[2] and $P_m^t(\mathbf{W})$ the $m$ best individuals of $P^t(\mathbf{W})$. Then the FEA algorithm is:

Establish population size & maximum number of generations;
$t = 0$;
Generate $P^t(\mathbf{W})$;
**While not** termination criterion **do**
  Evaluate $P^t(\mathbf{W})$;
  Extract best $\left[P^t(\mathbf{W})\right]$;
  Extract $P_m^t(\mathbf{W}) \leftarrow \left[P^t(\mathbf{W})\right]$;
  Set $P'^t(\mathbf{W}) \leftarrow$ selection $\left[P_m^t(\mathbf{W})\right]$;
  Subject to the PCM from the Decision Engine: Generate $P_{new}^t(\mathbf{W}) \leftarrow s$, ($\forall s$: $s$ = sample ($xi$, $yi$) or $s$ = sample $\mathbf{W}$, being $\mathbf{W} \in P'^t(\mathbf{W})$);
  Set $P^{t+1}(\mathbf{W}) \leftarrow \left\{P_m^t(\mathbf{W}) \cup P_{new}^t(\mathbf{W})\right\}$
  Set $P^t(\mathbf{W}) \leftarrow P^{t+1}(\mathbf{W})$
**Od**

[1] $U(0,1)$ = function that generates a random uniform number in (0,1).
[2] Let be $\mathbf{W} = (\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_i, \ldots, x_{nvar})$ and $\mathbf{y} = (y_1, y_2, \ldots, y_i, \ldots, y_{nvar})$. $\mathbf{W}$ consists in $nvar$ variables, $\mathbf{x}$, and the sampling methods used to sample each variable, $\mathbf{y}$.

```
Print solutions & stored information;
```

Under this scheme the algorithm can use any of the available sampling methods included in the Sampling Engine, being a little conservative but given it the possibility to explore using all the methods. Only two decisions can be taken by the PCM: to decide between to sample one or all variables and to decide between to use the preceding sample method or any other. The mechanism to correct wrong decisions is implicit because the Selection Engine may extract the best individuals from the population, any of them probably containing the best decisions in their genotype. The constraints are handled obliging to all the variables to be included in their feasible values. An earliest proposal of a FEA implementation can be found in Winter *et al* (2001).

### 4.1 THE SAMPLING ENGINE

Among all the functions constituting the FEA, the main component already developed is the 'Sampling Engine' which includes all the sampling methods that are to be used, both the Crossover ones (such as the Arithmetic crossover, the Geometric, the Heuristic, etc.) and the Mutation mechanisms (as the Uniform mutation operator, the non-Uniform one, etc), the majority included in the article by Michalewicz *et al* (1996) and in the INGENET network report by Bäck *et al* (1998). Furthermore, some new variations have been developed with the aim of exploring and/or exploiting the search space. The actual engine contains 39 sampling methods, many of which are detailed now. Nevertheless, and as a matter of lack of space, all the variations of methods are not mentioned. A more extended version of the components of the sampling engine can be found in Winter *et al* (2002). Let be:

*sign* = random variable in {-1,1}.

*nvar* = total number of variables.

$\mathbf{x}^{old1} = \left(x_1^{old1}, x_2^{old1}, \ldots, x_{nvar}^{old1}\right)$ = best solution found.

$\mathbf{x}^{old2} = \left(x_1^{old2}, x_2^{old2}, \ldots, x_{n\,var}^{old2}\right)$ = one of the best solutions found.

$\mathbf{x}^{old} = \mathbf{x}^{old1}$ or $\mathbf{x}^{old2}$.

$\mathbf{x}^{new} = \left(x_1^{new}, x_2^{new}, \ldots, x_{nvar}^{new}\right)$ = new solution.

As the sampling methods act, depending on the probabilistic frontier *alpha*, only over any of the variables or over all the variables of $\mathbf{x}^{old1}$ or $\mathbf{x}^{old2}$, $x_i^{old1}$ and $x_i^{old2}$ may be used when referring to a specific component of those vectors. Then, the new values are designated.

(a) *Sampling methods with different Nature*[3]:

(i) Assign a value: $x_i^{new} = x_i^{old}$.

(ii) Reduce or enlarge a value (dividing a value by another one):

$$x_i^{new} = \left(x_i^{old1} / x_i^{old2}\right) U(0,1).$$

(iii) Reduce or enlarge a value (using the addition):

$$x_i^{new} = x_i^{old} + sign\; x_i^{old}\, U(0,1).$$

(iv) Reduce or enlarge a value (using the addition plus the division):

$$x_i^{new} = \left(x_i^{old1} / x_i^{old2}\right) + sign\; x_i^{old}\, U(0,1).$$

(v) Reduce or enlarge a value (including a constant parameter $j$):

$$x_i^{new} = x_i^{old}\, sign + j\;\; sign x_i^{old}\, U(0,1).$$

(vi) Heuristic crossover:

$$x_i^{new} = x_i^{old2} + sign\left(x_i^{old1} - x_i^{old2}\right) U(0,1).$$

(vii) Geometric crossover (simplest version):

$$x_i^{new} = \sqrt{x_i^{old1} x_i^{old2}}.$$

(viii) Arithmetic crossover (simplest version):

If *auxi = U(0,1)* then

$$x_i^{new} = auxi\; x_i^{old1} + sign\;(1 - auxi) x_i^{old2}.$$

(ix) Guaranteed Average Crossover:

$$x_i^{new} = 0.5\, x_i^{old1} + o.5\, x_i^{old2}.$$

(x) Gaussian Mutation: $x_i^{new} = x_i^{old} + N(0, s)$.

(xi) Variant form of Gaussian Mutation (II):

If $q \in \{1, 2, \ldots, nvar\}$, $t_0 = 1/\sqrt{q}$ then

$$x_i^{new} = x_i^{old} + x_i^{old} e^{t_0 N(0,1)} N(0, s).$$

(xii) Variant form of Gaussian Mutation (III):

If $q \in \{1, 2, \ldots, nvar\}$, $t = 1/\sqrt{2q}$, $t' = 1/\sqrt{2\sqrt{q}}$ then $x_i^{new} = x_i^{old} + x_i^{old} e^{(t N(0,1) + t' N(0,1))} N(0, s)$.

(b) *Sampling methods with different Range*:

The rest of variations for completing the 39 sampling methods included in the engine have been obtained applying the three mechanisms described below for varying the range to some of the rest of the methods. For simplicity, only the general procedures are described and not the real variations[4].

(i) Multiplying for a random number (a certain number of times).

(ii) Dividing by a parameter.

(iii) Dividing by a certain value varying in an interval.

In order to illustrate the use of the sampling engine, Figure 6 shows the typical behavior that can be observed from the sampling methods in a single execution of the algorithm for Keane's function, one of the application examples used and whose expression is detailed in

[3] Only 12 out of the 18 methods really implemented. See (Winter *et al*, 2002) for further details.

[4] See (Winter *et al*, 2002) for further details.

Section 5. In the figure, two different graphs have been incorporated, both showing how many times is used every of the sampling methods introduced in the sampling engine along the optimization process in a single run. The upper diagram bar contains a representation of the times that every method is used ONLY to obtain the best solutions -one per generation- whereas the lower diagrams present the times that every method is used to get all the solutions-for every variable of each individual of all the generations-, that is to say, the total number of times that every method is used. The pattern of the lower graph differs slightly from one execution to another, as long as the number of generations considered is alike. This means that the behavior of the sampling methods for a specific function is very similar, and this is an important key to be able of building adequate sampling engines.

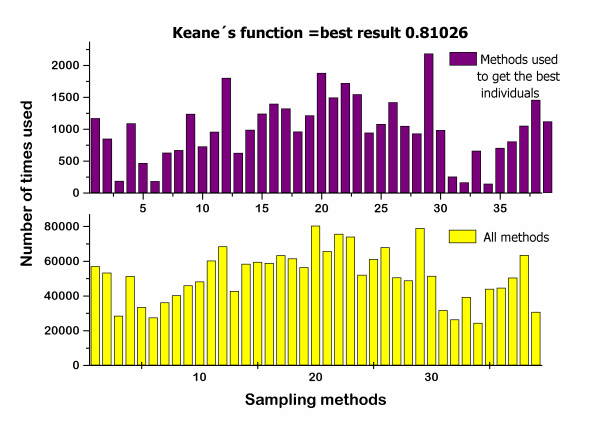



Figure 6: Keane's function: use of the different sampling methods

This is also the reason why that the use of mean graphics instead of the actual one containing just one execution is helpless for this objective. In contrast, the pattern of the graphic representing only the best methods used is invaluable at the moment of deciding which among them are needless or essential for the optimization process. Its pattern changes in the different executions of the algorithm, although its general shape is more or less constant. Hence, the way of deciding if a method is good or not when building a Sampling Engine can be to see if its bar in both graphics coincides being the higher or the lower. For instance, having a look at Figure 6 we can see that the methods 3, 6, 32 and 34 are clearly the worst ones, as they are the less used while the methods 12, 20 and 29 are the best ones. The competitive and cooperative behavior among sampling operators is shown. A survey reviewing what happens among them and how to build a Sampling Engine is being completed at present.

# 5 TEST RESULTS

We study the Flexible Evolution Agent robustness utilizing some well known mathematical test functions of proven complexity, frequently used for test purposes and whose analytical expressions are included below. The majority of them are multimodal functions presenting severe difficulties hardly overcame either by classical optimization methods or even by evolutionary methods. The number of variables $n$ is specified in every case of the tables of results.

**Keane's function:**

$$f(\mathbf{x}) = \left| \frac{\left[ \sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i) \right]}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

Constraints: $\prod_{i=1}^{n} x_i \geq 0.75$ , $\sum_{i=1}^{n} x_i \leq 0.75n$ and $0 \leq x_i \leq 10$ .

**Rastrigin's function:**

$$f(\mathbf{x}) = nA + \sum_{i=1}^{n} \left( x_i^2 - 2\cos(2\pi x_i) \right)$$

being $A = 10$ and $-5.12 \leq x_i \leq 5.12$ .

**Griewank's function:**

$$f(\mathbf{x}) = \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left( \frac{x_i}{\sqrt{i}} \right) + 1$$

where $d = 4000$ and $-600.0 \leq x_i \leq 600.0$ .

**Rosenbrock's function**:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left[ 100\left( x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$$

where $-5.12 \leq x_i \leq 5.12$ .

With the aim of illustrating the results that the method is able to get, two surveys have been done. The first one shows how efficient the FEA is by means of obtaining objective function values using as criterion for stopping the optimization process to reach at least (if it is possible) a fixed constant that has been called 'control value'. This value is different for each function and corresponds to some of the results found out by other researchers. The number of variables and individuals corresponds either to those utilized by these reference investigations. Tables 1 and 2 give the average results obtained for every test function in 30 runs.

From some of the attained values, it is not clear to know if establishing a stop criterion such as the defined above is enough for getting the best possible results using the FEA algorithm. Hence, it has been necessary to predetermine another control value in order to see how far it is able to reach. The new values have been fixed arbitrarily in the sense that as for any function are different and unknown

some tests have been done with the aim of deciding which to choose (see Table 2).

Table 1: Results for 30 runs using a predefined stop criterion[5] .

| **Functions** | | **Keane** | **Rastrigin** | **Griewank** | **Rosenbrock** |
|---|---|---|---|---|---|
| | Ref. | Bäck (1999) | Bäck (1999) | Herrera (2002) | Herrera (2002) |
| **Reference studies** | *n* | 20 | 20 | 25 | 25 |
| | **CV** | **0.803** | **$4.2135\,10^{-11}$** | **$6.77\,10^{-13}$** | **17.2** |
| | I | 100 | 100 | 61 | 61 |
| | G | >200 | 244 | - | - |
| | E | >20000 | 24400 | 100000 | 100000 |
| **Best values** | **OF** | **0.80329** | **$4.2633\,10^{-13}$** | **$1.1178\,10^{-16}$** | **$3.3390\,10^{-2}$** |
| | G | 1344 | 17 | 29 | 1 |
| | E | 114240 | 1700 | 1769 | 61 |
| **Mean values** | **OF** | **0.8031** | **$9.9984\,10^{-12}$** | **$1.4188\,10^{-14}$** | **3.32692** |
| | SD | $8\,10^{-5}$ | $3.7915\,10^{-11}$ | $1.4058\,10^{-14}$ | 4.07644 |
| | G | 3140.33 | 15.97 | 22.74 | 1.2 |
| | E | 266928.33 | 1596.67 | 1387.26 | 73.2 |
| **Worst values** | **OF** | **0.803001** | **$3.7915\,10^{-11}$** | **$4.4983\,10^{-14}$** | **16.75339** |

Table 2: Results obtained for 30 runs, with control values assigned arbitrarily[5].

| ***n* = 25** | | **Keane** | **Rastrigin** | **Griewank** | **Rosenbrock** |
|---|---|---|---|---|---|
| **Settings** | CV | > 0.80 | $\leq 10^{-13}$ | $\leq 10^{-16}$ | $\leq 10^{-9}$ |
| | I | 100 | 50 | 50 | 50 |
| **Best values** | **OF** | **0.81026** | **$2.8422\,10^{-14}$** | **$5.421\,10^{-20}$** | **$1.3713\,10^{-9}$** |
| | G | 1557 | 25 | 22 | 779 |
| | E | 132345 | 1250 | 1100 | 38950 |
| **Mean values** | **OF** | **0.8043** | **$3.1264\,10^{-13}$** | **$3.3175\,10^{-16}$** | **$1.0199\,10^{-5}$** |
| | SD | $3.14\,10^{-3}$ | $2.591\,10^{-13}$ | $3.2057\,10^{-16}$ | $2.0488\,10^{-5}$ |
| | G | 1506.23 | 21.33 | 25.07 | 855.3 |
| | E | 128029.83 | 1066.67 | 1253.33 | 42765 |
| **Worst values** | **OF** | **0.80002** | **$9.0949\,10^{-13}$** | **$8.9609\,10^{-16}$** | **$9.6154\,10^{-5}$** |

[5]Tables notation: *n* = number of variables, CV = control value, I = number of individuals, G = generations, E = evaluations, OF = objective function value and SD = standard deviation.

# 6 CONCLUSIONS

Nowadays, what has been pointed out as a benchmark in the evolutionary methods area is the algorithms capability of self-adapting their parameters and operators along the optimization process. Seen the features of the actual optimization procedures, a new implementation called Flexible Evolution Agent (FEA) has been presented in this work with the intention not only of taking advantage of some of the actual characteristics but also developing new mechanisms. The flexibility of the first approach made is due to its principal features: the Dynamic Structure of Operators it owns, the Extended Individuals Code and the Probabilistic Control Mechanisms for the decisions-making. The Engines in which the FEA is divided allow a more dynamic execution of the agent: in particular, the PCM (the Decision Engine) tries to reach a trade-off between the exploration and the exploitation of the search space, what is equivalent to achieve a competitive and cooperative balance among the sampling operators. The results obtained with the first FEA implementation have been very positive optimizing the majority of the used test functions, fact that permits to be optimistic about the future advances when the general FEA potentiality will be completely developed.

# 7 FUTURE WORK

Henceforth, our efforts may be centered in the introduction of several additional mechanisms (that can coexist or not) in the Decision Engine such as Fuzzy Logic or Expert Rule Based Systems in order to improve it. The next task could be completing a survey studying how the Sampling Engine truly works and how to build it. Lastly, a Filtering Engine to decide about the goodness of the attained solutions is to be included.

## References


Th. Bäck, F. Hoffmeister and H. P. Schwefel (1991). A survey of Evolution Strategies. In Lashon B. Belew and Richard K. Booker (eds.), *Proceedings of the 4th International Conference on Genetic Algorithms*, 2-9. San Diego, CA. Morgan Kaufmann.

Th. Bäck and B. Naujoks (June 1998). Innovative Methodologies in Evolution Strategies. Report D2.2 from INGENET Networks. http://ingenet.ulpgc.es/

Th. Bäck, W. Haase, B. Naujoks, L. Onesti and A.Turchet (1999). Evolutionary algorithms for academic and industrial cases. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki and J. Périaux (eds.), *Evolutionary Algorithms in Engineering and Computer Science*, 383-398. John Wiley & Sons, td, England.

M. C. J. Bot, N. Urquhart and K. Chisholm (2001). Agent Motion Planning with GAs Enhanced by Memory Models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2001, 227-234. Morgan Kaufmann Publishers, San Francisco, California.

P.J. Darwen (2000). Black Magic: Interdependence Prevents Principled Parameter Setting, Self-adapting costs too much computation. In *Applied Complexity: From Neural Nets to Managed Landscapes*, 227-237. Institute for Food and Crop Research, Christchurch, New Zealand.

L. Davis (1989). Adapting Operator Probabilities In Genetic Algorithms. In J.D. Schaffer (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, 61-69. Morgan Kaufmann Publishers, San Mateo, CA.

K. A. De Jong (1975). *An analysis of the behaviour of a class of genetic adaptive systems*. Doctoral Dissertation, University of Michigan, Ann Arbor, (University Microfilms Nº 76-9381), USA.

B. Edmonds (1998). *Meta-Genetic Programming: Co-evolving the Operators of Variation*. CPM Report Nº: 98-32. Centre for Policy Modelling, Manchester Metropolitan University.

A. E. Eiben, R. Hinterding and Z. Michalewicz (July 1999). Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, **III**(2): 124-141.

J.J. Grefenstette (1986). Optimization of control parameters for genetic algorithms. IEEE Trans. Systems, Man, and Cybernetics, SMC-**16**(1): 122-128.

F. Herrera, M. Lozano, E. Pérez and A. M. Sánchez (2002). Una propuesta de generación de múltiples descendientes y Selección de los dos mejores para Algoritmos Genéticos con codificación Real con el Operador de Cruce BLX-a. In E. Alba, F. Fernández, J. A. Gómez, F. Herrera, J. I. Hidalgo, J. Lanchares, J. J. Merelo and J. M. Sánchez (eds.), *Actas del Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados* (AEB'02), 15-22. Mérida, Spain.

R. Hinterding, Z. Michalewicz and T.C. Peachey (1996). Self-Adaptive Genetic Algorithm for Numeric Functions. In *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature*, Berlin.

C. Igel and M. Kreutz (2001). Operator Adaptation in Evolutionary Computation and its Application to Structure Optimization of Neural Network. In *Technical Report* IRINI 2001-3. Institut für Neuroinformatik, Rurh-Universität Bochum, 44780, Bochum, Germany.

N. Krasnogor and J. Smith (2001). Emergence of Profitable Search Strategies based on a Simple Inheritance Mechanism. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO), 432-439. Morgan Kaufmann Publishers, San Francisco, California.

P.L. Lanzi (2001). Mining Interesting Knowledge from Data with the XCS Classifier System. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2001, 958-965. Morgan Kaufmann Publishers, San Francisco, California.

S. Matsui and K-I Tokoro (2001). Improving the Performance of a GA for Minimum span Frequency Assignment Problem with and Adaptive Mutation Rate and a New Initialization Method. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2001, 1359-1366. Morgan Kaufmann Publishers, San Francisco, California.

Z. Michalewicz, G. Nazhiyath, M. Michalewicz, (1996). A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems. In L. J. Fogel, P. J. Angeline and Th. Bäck, (eds.), *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 305-312. The MIT Press, Cambridge, MA.

J.D. Schaffer, R. Caruana, L. Eshelman and R. Das, (1989). A Study Of Control Parameters Affecting Online Performance Of genetic Algorithms For Function Optimization, In *Proceedings of the 3rd International Conference on Genetic Algorithms*, 51-60, edited by J.D. Schaffer, Morgan Kaufmann Publishers, San Mateo, CA.

N. N. Schraudolph and R. K. Belew (1992). Dynamic Parameter Encoding for Genetic Algorithms. *Machine Learning*, **IX**(1): 9-21.

R. Weinberg (1970). Computer simulation of a living cell. Unpublished Doctoral Dissertation. **31**(9), 5312B. University of Michigan, Ann Arbor ,MI. (University Microfilms Nº 71-4766),USA.

G. Winter, B. Galván, P. D. Cuesta and S. Alonso (2001). Flexible Evolution. *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*. Athens, Greece.

G. Winter, B. Galván, S. Alonso and B. González (2002). Una propuesta de evolución flexible en el diseño de algoritmos evolutivos. In E. Alba, F. Fernández, J. A. Gómez, F. Herrera, J. I. Hidalgo, J. Lanchares, J. J. Merelo and J. M. Sánchez (eds.), *Actas del Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados* (AEB'02), 246-252. Mérida, Spain.

A. H. Wright and A. Agapie (2001). Cyclic and Chaotic Behavior in Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2001, 718-724. Morgan Kaufmann Publishers, San Francisco, California.