Toward Civilized Evolution: Developing Inhibitions

Michèle Sebag
LMS, CNRS 317
Ecole Polytechnique
91128 Palaiseau France

Marc Schoenauer
CMAP, CNRS 756
Ecole Polytechnique
91128 Palaiseau France

Caroline Ravisé
LRI, CNRS 410
Université d'Orsay
91405 Orsay France

Abstract

Most evolutionary algorithms concerned with a memory of evolution aim at memorizing and reusing the recipes of past successes (e.g. fruitful operators or fruitful mutation directions).
The scheme proposed here follows the opposite track, and memorizes the past failures of evolution (unfit offspring) through a virtual individual termed the *virtual loser*. The underlying metaphor is that offspring should attempt to get further away from the virtual loser than their parents. This is done by a new evolution operator, termed *flee-mutation*, as an alternative to standard recombination and mutation. Special attention is paid to adjusting the flee-mutation step size. Experiments on large sized problems validate this approach, and unexpectedly show that a constant flee-mutation step size over a population is desirable.

1 Introduction

In the framework of Evolutionary Computation (Goldberg 1989; Schwefel 1981; Fogel 1995), the population can be viewed as an implicit memory of the past evolution: the population is indeed representative of the successes (most fit individuals) encountered so far.

This paper rather focuses on explicit collective memory (EC-memory), meant as any kind of information which reflects the past history of evolution and is stored apart from the individuals themselves. This general definition notably encompasses all global parameters that are adaptively adjusted (Davis 1989; Eiben & Ruttkay 1996; Hansen, Ostermeier, & Gawelczyk 1995; Baluja & Caruana 1995; Dorigo & Giambardella 1996) as well as rules or beliefs learned over several generations (Reynolds & Matelik 1993; Reynolds 1994; Sebag & Schoenauer 1994; Ravisé & Sebag 1996).

EC-memory can be analyzed with regards to both its use and its contents. As a matter of fact, EC-memory can be used to drive most components of evolution: the selection of individuals (Eiben & Ruttkay 1996); the choice of evolution operators (Davis 1989); the way recombination and mutation operate (e.g. localization of crossing points and/or flipped bits, continuous mutation directions) (Sebag & Schoenauer 1994; Ravisé & Sebag 1996; Hansen, Ostermeier, & Gawelczyk 1995); and it can also control the generation of new individuals, providing an alternative to standard recombination and mutation (Baluja & Caruana 1995; Dorigo & Giambardella 1996).
As to its contents, EC-memory can either reflect uniquely the past successes of evolution, i.e. who are the fittest individuals and how they were found (Davis 1989; Baluja & Caruana 1995; Dorigo & Giambardella 1996); or, it can reflect both successes and errors (e.g. unfit individuals, disruptive operators) (Eiben & Ruttkay 1996; Reynolds 1994; Sebag & Schoenauer 1994).

This paper continues a previous work devoted to "civilized evolution", and based on coupling evolution and machine learning (Sebag & Schoenauer 1994; Ravisé & Sebag 1996): the knowledge learned from the past errors of evolution is used to prevent the further generations from repeating the past errors. In the present work, the EC-memory of errors is no longer expressed within rules, but as a template of past unfit offspring, termed *the virtual loser*. The virtual loser is used to directly evolve individuals via a new operator, termed *Flee-mutation*. The underlying metaphor is that offspring should attempt to be more different from the virtual loser, than their parents.

Flee-mutation combines mutation and recombination in the following respects. Like mutation, it proceeds by flipping some bits in the parent; one difference with regards to standard GA mutation, is that much more bits are flipped. The advantages of large-rate mutation have been emphasized for real-valued search spaces (see (Mühlenbein 1992) among others); and these have been confirmed for some binary problems too (Jones 1995).

On the other hand, flee-mutation resembles recombination, in that the probability of modifying a given bit depends on other individuals. This contrasts with both standard mutation and self adaptive mutation, where the modifications are respectively done at random or depend only on the individual at hand. The difference between recombination and flee-mutation is that the modifications are biased from the past populations (instead of the current population only).

This paper is organized as follows. Section 2 describes some works related to the explicit collective memory of evolution, without pretending to exhaustivity. Section 3 details the flee-mutation scheme; the critical points concern both the mutation step size (how many bits should flipped) and the choice of these bits. Section 4 presents and discusses an experimental validation of this scheme on large sized binary problems (900 bits) taken from (Baluja 1995). We conclude with some perspectives for further research.

2 State of the art

In a majority of works related to EC-memory, memorization proceeds by counting desirable events and/or rewarding fruitful options; it produces numerical informations (Davis 1989; Eiben & Ruttkay 1996; Dorigo & Giambardella 1996; Baluja & Caruana 1995; Hansen, Ostermeier, & Gawelczyk 1995). In other works, memorization is based on machine learning (Michalski 1983; Mitchell 1982), and proceeds by inducing logical rules or beliefs from examples drawn from evolution (Reynolds 1994; Ravisé & Sebag 1996). These two types of memory are respectively referred to as numerical and symbolic.

2.1 Numerical memory

Among the global parameters of evolution are the operator rates. Davis proposed an adaptive mechanism to control these parameters and choose between several types of recombination and mutation operators (Davis 1989). The idea is to gradually increase the rate of those operators which actually lead to fit individuals. In this frame, the EC-memory consists of the operator rates and it controls the choice of operators. The memorization process proceeds by relaxation[1].

Another example of collective memory in the context of SAT problems, is given by the SAW-GA (Eiben & Ruttkay 1996). In this approach, the violation of constraints is accounted for by penalization; the penalization for violating a constraint decreases when the constraint is often satisfied in the population. An individual which violates a widely satisfied constraint while being the only one to satisfy another constraint will thus more likely be selected. The underlying idea is that such an individual brings new valuable genetic material, whereas its failures can be repaired by recombination, as the corresponding genetic material is widely available in the population.
The EC-memory here consists of the penalization weights, and controls the selection step. The weights are also updated by relaxation.

As third example, PBIL memorizes the fittest individuals of the past generations into a virtual individual (in $[0,1]^N$ if N denotes the dimension of the search space) (Baluja & Caruana 1995; Baluja 1995). This virtual individual, which can be thought of as *virtual leader*, is interpreted as a vector of probabilities and used to construct the next population from scratch. For instance, if the virtual leader takes value .9 for the first bit, the first bit will be set to 1 for 90% of the individuals in the population. The EC-memory here consists of the virtual leader, and it allows to sidestep recombination and mutation. Again, the memory is updated by relaxation.

Still another example of evolution guided by numerical EC-memory is provided by artificial ants and pheronome trails (see (Dorigo & Giambardella 1996) among others). The pheronome trails reflect the interesting events of the recent history of ants (for biologic ants, the paths actually leading to some food; for artificial ants, for instance the best tours in the frame of TSP). These trails act as attractors: the more pheronome on a path, the more ants follow this path; and if this path still leads to something interesting, the more ants will add pheronome on this path.
Like in PBIL, this EC-memory can be viewed as a virtual individual[2]. It is used to generate the individuals (or their actions) and it is built by relaxation.

A last example is provided by (Hansen, Ostermeier, & Gawelczyk 1995). The history of one individual (real-valued vector) is constructed by relaxation from its successive climbing steps (difference between the parent and the offspring). The EC-memory consists of the history of the fittest individuals, eventually rearranged as to form a basis of the search space. The mutation of one individual is expressed as a linear combination in this basis, whose coefficients undergo self-adaptation. One advantage of this scheme is to allow self-adaptive mutation to be independent of the given coordinate system.

[1]Relaxation is commonly used (e.g. in neural nets) to ensure smooth updates and prevent numerical oscillations; it reads

$$p := (1-\alpha)p + \alpha\Delta p$$

where α ($\in [0,1]$) is the relaxation factor and governs the speed of changes, and Δp is the instant amount to be added or removed from p.

[2]For instance, in the TSP with N cities, an individual is a permutation over $\{1 \dots N\}$ while the EC-memory is a matrix ($N \times N$), whose coefficients belong to $[0,1]$.

2.2 Symbolic memory

Other works that explicitly refer to the memory of evolution are based on machine learning from examples (Michalski 1983; Mitchell 1982). In cultural algorithms (Reynolds & Matelik 1993; Reynolds 1994), one gradually learns from the individuals in the current population, and thus constructs beliefs about the relevance of schemas. EC-memory here consists of a lattice of beliefs, used to drive the generation of offspring.

In civilized evolution (Sebag & Schoenauer 1994; Ravisé & Sebag 1996), one learns from the operators (crossover and mutation masks) and thus constructs rules characterizing disruptive operators; an example of such a rule would be: *if there exists a crossing point between the 2nd and the 3rd bit, the crossover is disruptive*[3]; or *If bit 1 is mutated, the mutation is disruptive.* These rules are used to control the operators and decrease the odds of disruptive evolution. EC-memory here consists of logical rules.

In both cases, EC-memory encodes some knowledge: this knowledge (about fit/unfit individuals in cultural algorithms, and fruitful/disruptive evolution operators in civilized evolution) is used to guide the recombination and mutation of individuals. The Darwinian evolution paradigm thereby shifts toward "advanced evolution": some knowledge (culture, moral principles,...) is acquired along evolution, and the capitalization of knowledge expectedly speeds up evolution.

2.3 Lesson

This induction-based approach brought some gain with respect to the number of function evaluations needed to reach the optimum. Unfortunately, the cost of induction offsets or even exceeds this gain. This may be partly due to the fact that most test functions are inexpensive; in this context, control must be either exceedingly efficient, or of negligible cost to be competitive.

Still, civilized evolution gives some insights into how evolution actually works. For instance, it enables to compare the control of crossover and the control of mutation, since the same mechanism can be used to control either one or the other. And, despite the fact that the mutation rate was one or several orders of magnitude lower than that of crossover in these experiments, the control of mutation appeared much more efficient than that of crossover. In retrospect, it becomes clear that the disruptiveness of crossover decreases as evolution proceeds and the population gets homogeneous. But nothing, except control, can ever act against the disruptiveness of mutation: this can explain why the control of mutation, determining which bits should *not* be mutated, makes such a difference (Ravisé & Sebag 1996).

[3]Rules related to crossover are valid during a more or less short period of time: typically, after a relevant schema involving bits 2 and 3 has emerged, and before this schema has crowded the population.

In summary, what would be needed is a preventive memory, telling what *not* to do, easy to acquire and use. This leads to investigate numerical, rather than symbolic, EC-memory.

3 Inhibitions and Flee-Mutation

This section first motivates the choice of memorizing errors only. It describes how this memory is constructed and how it guides the choice of the bits to mutate. A last issue concerns the number of bits to mutate.

3.1 When are inhibitions reliable ?

Let us assume that the fitness landscape is fixed and that the fitness of an individual does not depend on the other individuals in the population. These assumptions are most often satisfied in applications of evolutionary computation. We further assume that the selection scheme is elitist, such as in the $(\mu + \lambda)$ evolution strategy scheme (Schwefel 1981).

Under these assumptions, if some offspring is not fit enough to survive selection at step t, it has no chance to survive selection in later steps. In other words, the generation of this offspring is a pure waste of time — an error of evolution — and *will remain so in the future* of evolution. Inhibiting the generation of such individuals thus makes sense.

The strategy based on the memorization and repetition of past successes (e.g. (Dorigo & Giambardella 1996; Baluja & Caruana 1995)) is basically oriented toward exploitation; it can be deceptive as it breaks the balance between exploration and exploitation (Goldberg 1989). In opposition, the strategy based on the memorization and avoidance of past errors restricts the scope of both exploitation and exploration: it just forbids to consider some unfit individuals, regardless of whether these individuals are close to current fittest individuals or not.

The strategy based on inhibitions (memorizing the errors in order not to repeat them) is closely related to the Tabu search (Glover 1977), where the last trials are stored in order not to be considered again. However, this is not directly applicable in the context of evolutionary search: the list of past individuals either covers a negligible fraction of the search space, or is intractable.

3.2 Utility of Inhibitions

The goal is to construct a tractable description of the past errors, and see how this can support evolution. Indeed, errors give some hints as to which bits should be mutated in relevant individuals. Let individuals X, Y, Z and T be as in Table 1, where X, termed *leader*, has a (comparatively) high fitness, and Y, Z and T, termed *losers*, all have a low fitness.

Bit 1 takes different values for the leader X and the losers Y, Z and T. By induction, this difference in their genotypes may be considered a "cause" for the difference in their fitness; one should therefore preserve this feature in X. Inversely, nothing can be said as to the influence of bit 5 regarding the fitness, as this bit takes same value for all individuals. Hence, mutation of X should rather affect bit 5 than bit 1.

Let the average of Y, Z and T be noted VL for *virtual loser*; the probability of mutating bit i in individual X should reflect how much it discriminates X_i from VL, that is, it should increase with $1 - |VL_i - X_i|$.

Table 1 : Comparing a winner and some losers

bit	1	2	3	4	5	Fitness
X	0	0	0	0	0	high
Y	1	1	1	1	0	low
Z	1	0	0	1	0	low
T	1	1	0	0	0	low
VL	1	.66	.33	.66	0	

In this framework, EC-memory consists of the virtual loser VL (in $[0,1]^N$). It is updated by relaxation from a fraction of the worse individuals in the current population. If α denotes some relaxation factor and $\mathtt{d}VL$ is the average of these worse individuals,

$$VL^{t+1} = (1-\alpha)VL^t + \alpha \cdot \mathtt{d}VL$$

3.3 Flee-mutation

Let X be the current individual and let p_i denote the quantity $1 - |VL_i - X_i|$. The flee-mutation operator can be viewed as a hill-climber where $(p_1, ..p_N)$ corresponds to the climbing direction (discrete gradient): the climbing direction consists in getting away from VL.

Let M denote the number of bits to mutate (see below). In a previous work, devoted to the no-memory case ($\alpha = 1$), the bits to mutate were selected by a roulette wheel on the p_i (Sebag & Schoenauer 1996). However, this showed ill-suited for $\alpha < 1$, as all p_i then get very close in the end of evolution. A strong selection pressure is therefore needed, which leads to mutate a restricted set of bits, which in turn entails a loss of genetic diversity and premature convergence.
In the present work, each bit to mutate is selected by tournament. This requires to know in advance the total number of bits to mutate in an individual.

Flee-mutation thus differs from standard mutation in that the probability of mutating one bit depends on both the bit itself and the individual at hand. The difference with self-adaptive mutation (mostly used in continuous search spaces (Schwefel 1981), but which has been extended to boolean search spaces (Bäck & Schütz 1995)), is that the individual climbing directions are here correlated by force through the EC-memory, whereas these are independent in self-adaptive mutation.

3.4 Flee-mutation step size

The tradeoff between exploitation and exploration in the flee-mutation scheme is governed by the number M of bits to mutate per individual, termed flee step size: like standard mutation, flee-mutation achieves exploitation for small values of M and does more and more exploration as M increases.

Adaptive adjustment. The ideal solution would be to adaptively adjust the number of bits to mutate. Several heuristics to this aim were tried: self adaptation of M, encoded as an integer in the individuals; global adjustment of M based on rewards, *a la* Davis (Davis 1989). Unfortunately, these strategies rapidly lead to flip only one bit, and flee-mutation thus gets trapped in local optima.
In retrospect, it appears that rewards-based adjustment tends to favor options that bring small frequent gains rather than large rare ones: the less a risky option is chosen, the less it is rewarded, and the less it will be chosen... The same goes for self-adaptation: both mechanisms are risk adverse and favor conservative options. And obviously, small values of M do result in more frequent, if smaller, performance gains.

At the population level. It was therefore decided to use fixed schedules to set the flee step size M^t at the evolution step t. The simplest possibility consists of using a single M for all evolution steps. A more sophisticated possibility is taken from (Bäck & Schütz 1996): M^t decreases with t according to an hyperbolic schedule (determined as optimal for the sphere problem). It reads

$$M^t = \frac{1}{\frac{1}{M^0} + t \cdot \frac{1-\frac{1}{M^0}}{T-1}} \quad (1)$$

where T is the allowed number of generations and M^0 the initial value.

At the individual level. In most approaches, mutation employs a probability of mutation per bit (Goldberg 1989; Schwefel 1981; Bäck & Schütz 1996). If the sum of these probabilities equals M^t, the number of flipped bits is indeed M^t on the average. But in the flee-mutation scheme, the number of bits to mutate must be known in advance for these bits are selected by tournament (section 3.3). Again, the simplest possibility consists of mutating exactly M^t bits for each individual of the current population. But this makes impossible to explore the whole search space, in the case where M^t is constant and even. And in all cases, some "stripes" of the space will be more difficult to reach than others.

Varying the number of bits to mutate at the individual level thus appears highly desirable. This is obtained by taking advantage of the following remark. The limit of the sum of N independent boolean random variables of probabilities q_i, when N goes to infinity while $\sum_{i=1}^{N} q_i$ goes to a finite value λ, is the Poisson distribution of parameter λ (Pitman 1993). The Poisson distribution of parameter λ is given by $P[X=k] = e^{-\lambda}\frac{\lambda^k}{k!}$, and is easily numerically realized by counting the number of uniform random variables in [0,1] needed before their product becomes less than $\exp -\lambda$. The Poisson approximation for the sum of random boolean variables is considered accurate for high values of N ($N > 50$), and hence holds for the problems considered in the following ($N = 900$).

How does this apply to our problem ? If flee-mutation were based on probabilities of mutation per bit, the sum of these probabilities should be M^t; and the number of bits effectively mutated would follow a Poisson distribution of parameter M^t. One may thus select an integer according to the Poisson distribution of parameter M^t, and use this integer as the number of bits to mutate for a given individual. Thereby, an – integer – number of bits to mutate per individual is determined, approximating the – real – M^t on the average. Yet, this number may widely vary from one individual to another.

Four kinds of flee-mutation are finally considered, by combining the following possibilities:

- At the population level, the flee step size M^t can be set to a user-supplied value M, or it can decrease from an initial value M^0 according to the hyperbolic schedule given by equation (1);
- At the individual level, the flee step size can be set to M^t or it can be selected according to the Poisson distribution of parameter M^t.

3.5 Overview of the algorithm

Flee-mutation is embedded into $(\mu + \lambda)$-ES: the population includes μ parents, λ offspring are derived from the parents by flee-mutation, and the selection is deterministic among the μ parent and λ offspring. Besides μ and λ, this algorithm involves 6 parameters:

- the scope and fading of inhibitions, meant as the fraction of the worse individuals used to update VL and the relaxation factor α (section 3.2).
- the strength of inhibitions, meant as the tournament size used to select the bits to mutate according to VL (section 3.3).
- the speed of evolution, parameterized by either the fixed flee step size M, or its initial value M^0 within an hyperbolic schedule (section 3.4).
- the variability of inhibitions over individuals (boolean), controlling whether the number of bits to mutate per individual is fixed over the population or obtained by Poisson-based selection (section 3.4).

4 Experimental Validation

The experimentations consider some problems taken from (Baluja 1995).

4.1 Test cases

These problems aim at optimizing six functions on $\{0,1\}^{900}$. These respectively correspond to the binary and Gray encoding of three functions F_1, F_2 and F_3 of 100 numerical variables x_i coded on 9 bits each and varying in [-2.56,2.56[.

$$\begin{aligned} y_1 &= x_1 \\ y_i &= x_i + y_{i-1},\ i \geq 2 \end{aligned} \qquad F_1 = \frac{100}{10^{-5} + \Sigma_i |y_i|}$$

$$\begin{aligned} y_1 &= x_1 \\ y_i &= x_i + sin(y_{i-1}),\ i \geq 2 \end{aligned} \qquad F_2 = \frac{100}{10^{-5} + \Sigma_i |y_i|}$$

$$F_3 = \frac{100}{10^{-5} + \Sigma_i |.024 * (i+1) - x_i|}$$

All functions admit a single optimum. The discretized F_1 and F_2 have same optimum as the continuous F_1 and F_2, (10^7 is obtained for $(0, \ldots, 0)$). The discretized F_3 culminates at 416.649 (the optimum 10^7 is obtained for ($x_i = .024 * (i+1)$), which does not belongs to the discretized space).
Note that F_3 with Gray coding is equivalent to the Onemax problem: for any individual X, there exists a path ($X_0 = X, X_1, ...X_{max}$) linking X to the optimum, with X_i and X_{i+1} differing by one bit, and $\mathcal{F}_3(X_i) < \mathcal{F}_3(X_{i+1})$.

4.2 Reference algorithms

Multiple restart hill-climbers denoted HC1 and HC2, as well as two genetic algorithms denoted SGA and GA-scale, have been used in (Baluja 1995) as reference algorithms:

- HC1 randomly considers the neighbors of the seed (differing from the seed by one bit); the seed is replaced by the first neighbor which strictly improves on the seed. If all neighbors have been considered, HC1 is restarted with another seed.

Table 2: Average best fitness (20 runs) after 200,000 evaluations. Results of AES and TES have been obtained using the algorithms described in the text; other results are taken from (Baluja 1995). On the average, PBIL slightly outperforms Adaptive ES and both outperform other algorithms, except on the Onemax-like F3 gray.

	HC1	HC2	SGA	GA-scale	AES	TES	PBIL
F1 binary	1.04	1.01	1.96	1.72	**2.37**	1.87	2.12
F1 Gray	1.21	1.18	1.92	1.78	2.04	1.66	**2.62**
F2 binary	3.08	3.06	3.58	3.68	3.94	3.61	**4.40**
F2 Gray	4.34	4.38	3.64	4.63	5.18	4.66	**5.61**
F3 binary	8.07	8.10	9.17	12.30	9.06	10.46	**16.43**
F3 Gray	**416.65**	**416.65**	28.35	210.37	380.3	**416.65**	366.77

Table 3: Average best fitness (20 runs) after 200,000 evaluations, obtained for flee-mutation and free-mutation (no inhibitions), and corresponding to optimal values of M and M^0 (between parentheses). The best strategy appears to decrease the mutation step size over generations and keep it constant over a population.

	Flee-Mutation				Free-Mutation			
	M_t fixed		M_t decreases		M_t fixed		M_t decreases	
	Poisson	Cst	Poisson	Cst	Poisson	Cst	Poisson	Cst
F1b	2.48 (4)	2.92 (3)	2.36 (9)	**2.99** (9)	2.48 (4)	2.79 (3)	2.35 (9)	**2.98** (90)
F1g	2.30 (3)	2.65 (3)	2.26 (450)	**2.69** (45)	2.11 (4)	2.48 (4)	2.09 (450)	2.61 (450)
F2b	4.53 (4)	4.93 (4)	4.52 (450)	**5.35** (450)	4.16 (4)	4.52 (4)	4.09 (45)	5.13 (45)
F2g	5.85 (3)	6.46 (3)	5.74 (45)	**6.78** (45)	5.12 (3)	5.82 (3)	5.42 (450)	6.48 (90)
F3b	12.83 (4)	12.94 (3)	16.38 (90)	17.89 (45)	12.71 (4)	13.66 (3)	16.74 (450)	**18.85** (90)
F3g	414.96 (3)	72.43	**416.65**	246.23 (9)	**416.65**	246.22 (3)	**416.65**	385.90 (9)

- HC2 differs from HC1 in that it allows to replace the seed by a neighbor having similar fitness.
- SGA is a standard GA which uses two point crossover, with crossover rate 100%, mutation rate 10^{-3}, population size 100 and elitist selection.
- GA-scale differs from SGA in two respects: it uses uniform crossover with rate 80%, and the fitness of the worst individual is subtracted from the fitness of all individuals in the population before selection. A more detailed description of these algorithms, as well as their results on the considered problems, can be found in (Baluja 1995).

Two additional reference algorithms have been considered here:

- TES (for *Traditional evolution strategy*) is a binary $(\mu + \lambda)$-ES involving a single mutation rate per bit σ ; σ is modified according to the Rechenberg's 1/5 rule (Rechenberg 1973). The geometrical factor used to increase σ ranges from 1.1 to 2.
- AES (for *Adaptive ES*) is a binary $(\mu + \lambda)$-ES that uses the adaptive mutation of (Obalek 1994) as described in (Bäck & Schütz 1995): each individual is attached one mutation rate per bit σ which itself undergoes mutation according to the following *Obalek's rule*:
$$\sigma := \left(1 + \frac{1-\sigma}{\sigma}.exp(\gamma\ .\ N(0,1))\right)\ ;\ \gamma = \frac{0.5}{\sqrt{2\sqrt{N}}}$$
We further require $\sigma \geq 1/N$ to guarantee effective mutation. In both cases, μ is 10 and λ is 100.

4.3 Results

The experimental settings of the flee-mutation scheme are the following: the ES parameters are $\mu = 10$ and $\lambda = 50$; the inhibitions are constructed from the only worse offspring, and the relaxation factor α is set to .01; the strength of inhibitions is set to 3. These parameters were found rather robust for all problems. On the opposite, the value of M appears critical depending on the fitness landscape.

In order to check the importance of inhibitions, the flee-mutation scheme is compared to an "free" scheme (legend *Free-mutation* in Table 3); free-mutation performs like flee-mutation based on void inhibitions ($VL = (.5, \ldots, .5)$). In both cases, the number of bits to mutate M^t is either fixed or decreases after an hyperbolic schedule at the population level, and it is either constant or follows a Poisson distribution at the individual level. Note that the free scheme with Poisson-based selection of the bits to mutate corresponds to flipping all bits with probability M^t/N, as done in (Bäck & Schütz 1996).

Several values of M (in the fixed schedule) and M^0 (in the hyperbolic schedule) have been tried. Table 3 reports the best results, together with the corresponding value of M (varying in 2..5) and M^0 (varying in 9, 45, 90, 450).

All algorithms are allowed 200,000 fitness calculations per run. Results are averaged over 20 independent runs. The best results are indicated in bold (Table 2 and Table 3; several results are in bold for $F1b$, as their difference is not statistically significant from the standard deviations).

4.4 Discussion

These results ask for several comments.
First of all, the best results obtained with the flee-mutation scheme improve on PBIL and significantly outperform standard GAs et ESs. But the question of determining the optimal value for M or M^0 remains open.

Second, as expected and in agreement with (Bäck & Schütz 1996), results are better when the flee step size decreases along evolution (hyperbolic schedule). Yet, if we except the Onemax-like F_3 Gray, evolution stops quite far from the global optimum; it therefore should not need fine tuning ($M = 1$).

Third, and this was unexpected, varying the flee step size over the individuals (Poisson distribution) degrades the results compared to keeping it constant. For both flee- and free-mutation, the best choice is to decrease the flee step size M^t along evolution, and mutate exactly $(int)M^t$ bits for all individuals in the population.

Last, flee-mutation improves on free-mutation — but not that much. This may imply that the real efficiency rather comes from setting the mutation step size, than the mutation direction; and from the third remark, it is better to use a fixed mutation step size, rather than scattering the offspring. Still, this may be an artifact due to the artificial test functions considered; on the Long Path problem for instance (Horn & Goldberg 1995), no wonder tremendous results were obtained by setting the mutation step size M to 2 (Sebag & Schoenauer 1996).
Another possible explanation is that the contents and strength of inhibitions were too restricted to cause a great difference: inhibitions are constructed from the single worse individual and the inhibition strength was low (the bits to mutate were selected with a tournament size of 3). Still, the gain is statistically significant.
Further work will address both issues: consider less regular binary functions, e.g. obtained from F_1, F_2 and F_3 by a rotation on $\mathbb{R}^{100}$; and investigate what happens when the influence of inhibitions is increased.

5 Conclusion and Perspectives

In the framework of biologic evolution, the acquisition of inhibitions based on the past of evolution might be irrelevant: such inhibitions may turn out to be deceptive as the environment and the fitness landscape change.

But most EC applications deal with fixed fitness landscapes: refraining from repeating past errors thus constitutes a sensible inhibition. The proposed scheme therefore combines evolution and some ideas of Tabu search: the list of past trials is replaced by a distribution, the *virtual loser*. Flee-mutation evolves the individuals to get away from the virtual loser; it extends recombination (modifications are biased according to the past populations) and yet enjoys a desirable property of mutation: the distance between offspring and parents (flee step size) can be controlled.

The potentialities of this scheme are demonstrated on several large sized problems. The main weakness precisely concerns the setting of the flee step size. Further research will investigate how a priori estimates, inspired for instance by the Fitness Distance Correlation (Jones & Forrest 1995), could support this setting.

Another direction of research is to extend flee-mutation to continuous search spaces. The memory itself can still be represented as a distribution over the search space, but the critical point will be how to use it.

Last, the ideal scheme would be to take advantage of two memories: that of errors and that of successes (acquiring motivations as well as inhibitions). Comparing both memories might also give hints regarding the setting of the flee step size.

References

Bäck, T., and Schütz, M. 1995. Evolution strategies for mixed-integer optimization of optical multilayer systems. In McDonnell, J. R.; Reynolds, R. G.; and Fogel, D. B., eds., *Proceedings of the 4^{th} Annual Conference on Evolutionary Programming*. MIT Press.

Bäck, T., and Schütz, M. 1996. Intelligent mutation rate control in canonical gas. In Ras, Z. W., and Michalewicz, M., eds., *Foundation of Intelligent Systems 9th International Symposium, ISMIS '96*, 158–167. Springer Verlag.

Baluja, S., and Caruana, R. 1995. Removing the genetics from the standard genetic algorithms. In Prieditis, A., and Russel, S., eds., *Proceedings of ICML95*, 38–46. Morgan Kaufmann.

Baluja, S. 1995. An empirical comparizon of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University.

Davis, L. 1989. Adapting operator probabilities in genetic algorithms. In Schaffer, J. D., ed., *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, 61–69. Morgan Kaufmann.

Dorigo, M., and Giambardella, L. 1996. A study of some properties of Ant-Q. In Voigt, H.-M.; Ebeling, W.; Rechenberg, I.; and Schwefel, H.-P., eds., *Proceedings of the* 4^{th} *Conference on Parallel Problems Solving from Nature*, volume 1141 of *LNCS*, 656–665. Springer Verlag.

Eiben, A., and Ruttkay, Z. 1996. Self-adaptivity for constraint satisfaction: Learning penalty functions. In Fukuda, T., ed., *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, 258–261. IEEE Service Center.

Fogel, D. B. 1995. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press.

Glover, F. 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences* 8(1):156–166.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley.

Hansen, N.; Ostermeier, A.; and Gawelczyk, A. 1995. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In Eshelman, L. J., ed., *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, 57–64. Morgan Kaufmann.

Horn, J., and Goldberg, D. 1995. Genetic algorithms difficulty and the modality of fitness landscapes. In Whitley, L. D., and Vose, M. D., eds., *Foundations of Genetic Algorithms 3*, 243–269. Morgan Kaufmann.

Jones, T., and Forrest, S. 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L. J., ed., *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, 184–192. Morgan Kaufmann.

Jones, T. 1995. Crossover, macromutation and population-based search. In Eshelman, L. J., ed., *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, 73–80. Morgan Kaufmann.

Michalski, R. 1983. A theory and methodology of inductive learning. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann. 83–134.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* 18:203–226.

Mühlenbein, H. 1992. How genetic algorithms really work: I. mutation and hill-climbing. In Manner, R., and Manderick, B., eds., *Proceedings of the* 2^{nd} *Conference on Parallel Problems Solving from Nature*, 15–25. Morgan Kaufmann.

Obalek, J. 1994. *Rekombinationsoperatoren für Evolutionsstrategieren.* Ph.D. Dissertation, Universität Dortmund, Fachbereich Informatik.

Pitman, J. 1993. *Probability.* Springer Verlag.

Ravisé, C., and Sebag, M. 1996. An advanced evolution should not repeat its past errors. In Saitta, L., ed., *Proceedings of the* 13^{th} *International Conference on Machine Learning*, 400–408.

Rechenberg, I. 1973. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* Stuttgart: Fromman-Holzboog Verlag.

Reynolds, R., and Matelik, 1993. The Use of Version Space controlled Genetic Algorithms to Solve the Boole Problem. In *Int. J. on Artificial Intelligence Tools*, Vol 2, N2, 219–234.

Reynolds, R. 1994. An introduction to cultural algorithms. In *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, 131–139. World Scientific.

Schwefel, H.-P. 1981. *Numerical Optimization of Computer Models.* New-York: John Wiley & Sons. 1995 – 2^{nd} edition.

Sebag, M., and Schoenauer, M. 1994. Controlling crossover through inductive learning. In Davidor, Y.; Schwefel, H.-P.; and Manner, R., eds., *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, 209–218. Springer-Verlag, LNCS 866.

Sebag, M., and Schoenauer, M. 1996. Mutation by imitation in boolean evolution strategies. In Voigt, H.-M.; Ebeling, W.; Rechenberg, I.; and Schwefel, H.-P., eds., *Proceedings of the* 4^{th} *Conference on Parallel Problems Solving from Nature*, 356–365. Springer-Verlag, LNCS 1141.