Fundamenta Informaticae 33 (1998) 1–38
IOS Press

Revisiting the Memory of Evolution

Michèle Sebag, Marc Schoenauer and Mathieu Peyral*

LMS and CMAP, Ecole Polytechnique,
91128 Palaiseau Cedex, France
{Michele.Sebag,Marc.Schoenauer,Mathieu.Peyral}@polytechnique.fr

Abstract. A new evolution scheme is presented, memorizing the extreme (best and worst) past individuals through distributions over the binary search space. These distributions are used to bias the mutation operator in a $(\mu + \lambda)$ Evolution Strategy, guiding the generation of newborn offspring: different *mimetic strategies* are defined, combining either attraction, indifference or repulsion with respect to the two distributions. These distributions are then updated from the best and the worse individuals in the current population. Experiments on large size binary problems allow one to delineate the niche of each of these mimetic strategies.

1. Introduction

The powerful process of natural evolution indeed produced biological chefs d'oeuvre. The field of artificial evolution is concerned with transposing and mastering the strengths of evolution within the machine world [24, 45, 12, 28]. A number of applications, ranging from optimization and design problems to adaptation and evolvable hardware, fully demonstrates the efficiency of the approach.

In its first period, artificial evolution was almost exclusively inspired from biology (adaptation [24], diploidy [19], introns [29], Baldwin effect [23] ... among others) and most authors praised artificial evolution as a universal tool [18]. Nowadays, it is widely acknowledged that one should take into account the specificities of the problem at hand, through the representation of the search space and the design of the evolution operators. The advantages of using "expert" representation and operators are illustrated for instance in the domain of combinatorial

*Address for correspondence: LMS and CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex, France

optimization [36] or shape design [44] (see [35] for general recommendations about representation/operators).
As noted by Janikow [26], this transition is quite similar to what happened in the field of artificial intelligence (see [41] for a survey): at first, people were fascinated by the generality of the principles at hand and they aimed at universal tools, e.g. the General Problem Solver [33]. Afterward, they realized that the difference between being able to solve the problem and actually solving it, was a matter of knowledge — and this led to developing knowledge-based systems (KBS) [7].

The history of artificial intelligence has perhaps one further lesson to offer artificial evolution. The KBS approach met many successes as more accurate ways of representing and using knowledge were designed. However, where does knowledge come from in the general case ? As noted by E. Feigenbaum, the main bottleneck of artificial intelligence was eventually the knowledge acquisition phase; this gave birth to a new field of AI: machine learning [31]. To put it into a nutshell, the history of artificial intelligence suggests that any advanced information processing tool needs knowledge, and that the only practical way to have knowledge is to acquire it, that is, to devote some efforts to learning. So what could be the role of a learning module within artificial evolution ?

To keep clear distinctions between evolution and machine learning, the knowledge to be automatically acquired by artificial evolution is referred to as *memory of evolution*, and the way it is learned is referred to as *memorization*. Memory and memorization are peripheral concerns for artificial evolution, though many works concerned with the control of evolution could be rephrased in these terms (more in section 2). Indeed, memory is not much present in natural evolution, barring the genetic material itself. And natural evolution still is the dominant paradigm for artificial evolution; one is prone to dismiss concepts and heuristics which are too far away from what one understands to be evolution[1].

On the other hand, as emphasized in [9], the "specifications" for natural evolution might be rather different from that of artificial evolution. Notably, natural evolution explores a changing fitness landscape while artificial evolution considers a fixed fitness landscape most of the times. When the context changes, recording the past of evolution does not make much sense: even if it were feasible, this would provide irrelevant or, even worse, misleading information. An accurate memorization process should then keep a cautious balance between memorizing and forgetting.

The situation is much more straightforward when the world does not change: in principle, evolution could plainly and soundly use its history to avoid the repetition of previous trials/errors, in the line of Tabu Search [16]. The question no longer regards the utility of memory, but rather its technical feasibility.

We propose a categorization of the possible memories of evolution, and the way these can be used in section 2. No wonder this categorization is close to that proposed for the control of evolution [22], as memory and control of evolution are tightly related. We introduce a fur-

[1] It is worth noting that our understanding of evolution is by no way complete, and could therefore prove misleading. In the same vein, much time has been lost in constructing flying machines based on our at-that-time models for birds [41].

ther distinction, as to whether the memory allows for the distinction between new offspring and previously generated individuals. If the distinction is possible, the memory can be used to favor the exploration of brand new individuals: evolution becomes irreversible. To do so, memory-based control can either proceed by incentives or inhibitions. The *Population Based Incremental Learning* (PBIL) [5, 4] algorithm proceeds by incentives: it constructs the memory of the past best individuals encountered during evolution, and use this memory as an attractor of the offspring. *Evolution by Inhibitions* (EbI) [47, 48] proceeds by inhibitions: it constructs the memory of the past worst individuals of evolution and use this memory as a beacon to draw the population away from the dead-ends.

In this paper, both approaches are combined: we investigate how evolution can take advantage of incentives and inhibitions altogether. Only binary search spaces ($\{0,1\}^N$) are considered. The memory of best/worst individuals does then belong to the continuous space $[0,1]^N$; it corresponds to a "virtual individual", or model. Individuals are provided with the two models memorizing the best and the worst past individuals, respectively termed the *Leader* and the *Repoussoir*[2]. Each individual uses the models as reference points, to decide where it should go next, i.e. where to localize its offspring. The offspring are generated as to be closer to, or farther away from, a model. Metaphorically, the individual imitates or rejects the models: its "social strategy" dictates the distribution of its offspring. For instance, one natural strategy is to reject the *Repoussoir* and imitate the *Leader* (termed *Sheep* strategy); another one, termed *Lone Rider* strategy, is to reject both the *Leader* and the *Repoussoir.*

The presented scheme of evolution, termed *Mimetic Evolution*, thus abandons the biology paradigm, and rather finds its metaphors in the field of sociology. Mimetic evolution involves a single evolution operator, termed mimetic mutation, which replaces both standard mutation and crossover. The material in each individual is modified according to the models, and depending on the social strategy of the individual. We restrict ourselves to consider a single social strategy for all individuals during all evolution.

The paper is organized as follows. Next section discusses the possible roles of memory depending on whether the fitness landscape changes or not. It briefly reviews some related work concerned with the control of evolution, and focuses on irreversible control.

Section 3 details the *Mimetic Evolution* scheme combining *PBIL* and *Evolution by Inhibition.* This scheme involves mimetic mutation as single evolution operator. Like standard mutation, mimetic mutation considers one individual at a time. The choice of the bits to mutate is based on the memories and can achieve the diversification or the recombination of individuals depending on the chosen social strategy.

Experiments on large sized binary problems are presented in section 4. These show that the *Sheep* and *Lone Rider* strategies are relevant to many problems — not all, as could have been expected. Interestingly, the best strategy for a problem gives some insight into the difficulty

[2] *Repoussoir* is a French word intended for: the person you would not like to be, or look, like.

of the problem: the *Sheep* strategy shows more adapted to climb simple slopes, in a gradient-like manner; the *Lone Rider* strategy primarily preserves the diversity of the population, and thus escapes more easily from local optima. The paper ends with some perspectives for further research.

2. Memory in natural and artificial evolution

Obviously, artificial evolution only constitutes a coarse simplification of natural evolution. Our claim is that even greater simplifications are possible by taking advantage of the steadiness of the artificial milieu.

Simplifications are obtained through a better use of the available information in each step. We assume in the remainder of the paper that the optimization problem at hand is sufficiently difficult, so it is worth spending a reasonable time looking for short cuts.

This section discusses earlier work related to the control of evolution. Concretely, control is a way of shifting the information processed by evolution from a low-level description (e.g. the current genetic pool, or *description in extension* of the problem) to a more abstract level: operator rates [8, 38], beliefs [39], rules [37], gradients [48] or even more directly, the distribution of the offspring [4], which can be viewed as a *description in intension* of the problem.

2.1. Changing versus Fixed Worlds

Our understanding of evolution is far to be complete, as the ends can only be conjectured [53]. However, there is a wide acceptance that natural evolution is "designed" for changing environments, and aims at adaptation [24]. As emphasized by [9], this means starting with a "rather good" initial population (avoiding the bootstrap problem [34]), and measuring the success from both the cumulated performance of the individuals, and the chances for a sufficient fraction of the population to survive a further change of the milieu.

Specifications for artificial evolution are different: most of the times, the goal is to find the optimum of a fixed fitness function. The success is only measured from the performance of the best individual.

In both contexts, the leading individuals are ceaselessly replaced by more fit individuals. But *within natural evolution*, these more fit individuals are *not necessarily new individuals*, as previous outsiders can come back to outperform the actual leaders. This is never true in a fixed environment.

The consequences for this are manifold. Natural evolution must somehow preserve whatever has been relevant in the past, for it can become useful again later. In the meanwhile, the relevance of all information must be periodically re-evaluated as the milieu changes. As the evaluation procedure only concerns individuals, any relevant information must be coded within the genetic material, i.e. in extension. Moreover, evolution cannot draw any negative conclusions, as to which material can be soundly discarded.

2.2. Challenge for an informed evolution

Let us go back to artificial evolution. Assuming that the world does not change allows — in principle — dramatic simplifications of evolution: re-evaluating an individual does not provide any further information, and can therefore be omitted.

The steadiness of the milieu induces a partition of the individuals into four subsets,

- A The set of the individuals which are outperformed; they do not need be considered again as their time has past and will never come back. These are the dead individuals.
- B The set of the currently fit individuals, which do not need be considered either but are somehow used to produce offspring; these are the living individuals.
- C The embryos include those candidate offspring given the current experience/state of evolution, which have not yet been evaluated.
- D All other individuals[3].

The only set which needs be evaluated and must therefore be characterized in extension, is the set of embryos [C]. The only set which cannot be characterized in extension is the set [A] of dead individuals, as the time and space complexity would be intractable after the first generations.

Along evolution, the size of [A] grows (there are more and more dead individuals) and that of [D] decreases; the size of [B] remains more or less constant, except for the loss of genetic diversity.

By construction, the offspring generated in each generation are distributed among the embryos [C] and the past individuals, either living [B] or dead [A]. But *the only interesting offspring are in [C]*. This leads to two approaches of the control of evolution, depending on whether the stress is put on the expected quality of the offspring, or on their novelty. These approaches are respectively referred to as *reversible* and *irreversible* control.

2.3. Reversible control

Control is most generally concerned with biasing the distribution of the offspring, to favor the discovery of interesting individuals.

Obviously, we only can make conjectures about where the interesting individuals lie; otherwise, a deterministic approach would be recommended. Such conjectures regard the way the distribution of the offspring should depend on the current state of the system: some parameters or structures of control are identified. The goal is to determine relevant contents for these parameters or structures of control.

Several criteria have been proposed to distinguish between the great many controls investigated in the literature [49, 22].
A first criterion considers whether the contents of the control is determined on-line or off-line [49]. Off-line control is adjusted via some preliminary analysis or experiments (see for instance

[3]It would be more exact to consider the notion of candidate offspring as a distribution (one individual has more or less chance to appear at a given step). Still, as no theoretical analysis will follow, and for the sake of simplicity, the notion of candidate offspring is taken boolean.

[20]), and is used within a static or dynamic schedule. For instance, the mutation rate can be set to a constant value, or decreased via a hyperbolic schedule [3]. As off-line control obviously is not concerned with the specificities of the run, it makes no part to the memory of evolution and is not considered thereafter.

On-line control, also termed *adaptive control*, can operate at different levels of evolution: the environment, the population, the individuals, or the genes [22]. The contents of control can either be deterministically adjusted depending on some predefined indicators (explicit control); or it can be carried by individuals themselves, and adjusted "for free" by evolution (implicit control).

2.3.1. Explicit control

Examples among others of explicit control are the 1/5th rule [38], the adjustment of the operator rates *a la* Davis [8], the early EP mutation [13], the constitution of pheromone trails [10], and the adjustment of penalty factors [11].

In [38, 12] the control only responds to the current state of evolution (the fraction of the mutation success, the relative fitness of an individual); this can be viewed as a reflex control.

In [8, 11, 10, 39, 37], the control rather takes into account the whole past of evolution and proceeds by reinforcing the good options or the relevant choices (reinforced control).

2.3.2. Implicit control

Implicit control lets evolution itself adjust the contents of the control. This is most usually done by coding the parameters of control within the individuals, such as the mutation step size in self-adaptive mutation [45, 1] or the type or mask of crossover in genetic algorithms [43, 49]. Practically, the control-related part of the individual is first evolved, then used to derive a genotype. The offspring is then composed of the current control part, and the derived genotype. Though evolution can only evaluate the genotype, it expectedly optimizes both the genotype and the control-related part of the individuals. This does not rely on any magic of evolution: rather, individuals carrying an irrelevant control part disappear as their genotype is most likely to be unfit or non viable.

Hybrid control, interleaving global deterministic and local non-deterministic indicators, has also been proposed [21].

2.4. Irreversible control

In all above approaches, the time of evolution is reversible: one could obtain the parents of given offspring by applying the same operators as used to build the offspring from the parents. Basic genetic operators indifferently produce ascendants, or descendants, of the current individuals[4].

[4] This is particularly clear in boolean search spaces: e.g., by applying a given crossover mask, one can derive two offspring X' and Y' from two parents X and Y *OR* derive the parents X and Y from the offspring X' and Y'.

The offspring might actually be new embryos (from set [C]), or old individuals (living or dead).

Still, the fact that the world does not change implies that the only worth offspring are new individuals. This can be used as a constraint to prune the set of candidate offspring, in the spirit of the Tabu search [16]. The advantage is that biasing the distribution of offspring toward the embryos (set [C]) effortlessly (so to speak) increases the efficiency of the search, everything else being equal. As a matter of fact, memory could in principle tell whether an offspring is a new individual, as all necessary information has been available at some point in the past. On the opposite, no mechanism (except evaluation) could tell whether an offspring will turn out to be fit.

Irreversible control enforces the novelty of the offspring, at the cost of memorizing the past of evolution.

2.4.1. PBIL

Population Based Incremental Learning – PBIL [5, 4] takes advantage of the similarity between evolutionary algorithms and generate-and-test methods (see [31] for a survey of Machine Learning (ML)). In both cases, the system generates questions (individuals in the artificial evolution context, examples in the machine learning context); these questions are answered by the oracle (individuals are evaluated, examples are labeled); the answers are used in turn by the system to refine its current hypothesis and generate next questions. The target "hypothesis" is an accurate description of either the distribution of the optima or some explanatory concept.

One key difference between generate-and-test methods and artificial evolution lies in the level of description of the internal state of the system. Generate-and-test methods handle high-level representations; the current hypothesis is described in intension [31]. On the opposite, artificial evolution handles the individuals themselves: its "hypothesis" is described in extension, through the population.

PBIL achieves artificial evolution by learning a single high-level genotype: a distribution $\mathcal{M}$ = $(\mathcal{M}_1, \ldots \mathcal{M}_N)$ over the search space $\{0,1\}^N$. Each component $\mathcal{M}_i$ stands for the probability that bit i is 1. *PBIL* initializes $\mathcal{M}$ to the most general distribution ($\mathcal{M}_i = .5$). $\mathcal{M}$ is alternatively updated by relaxation from the best current offspring (section 3.2), and used to generate from scratch a new population. $\mathcal{M}$ can be viewed as the memory of the past best offspring.

PBIL thereby sidesteps all evolution operators and selection needed to transform a population into another one. The diversity and novelty of the offspring can be directly enforced by means of the selectivity and the fading of the memory $\mathcal{M}$ (see section 3.2).

2.4.2. Evolution by inhibitions

Symmetrically, *Evolution by Inhibitions* (EbI) gradually constructs the memory $\mathcal{R}$ of the past worst offspring [48]. The difference with *PBIL* is that $\mathcal{R}$ does not provide enough information to generate the offspring from scratch. Rather, it tells where the offspring should *not* be: they should not be close to $\mathcal{R}$.

EbI thus uses its memory to ensure that the offspring are sufficiently different from outperformed individuals, and by extension, fall outside previously explored regions of the search space. It thereby directly enforces the novelty of the offspring.

2.5. Directions for the current study

Both above schemes involve an explicit memory of evolution, but they use it in different ways.

In *PBIL*, the memory is interpreted as a distribution over the search space: it replaces the genetic pool of evolution as it can be used to generate a population from scratch. Much attention is paid to prevent the premature convergence of the mechanism, by controlling the selectivity of the memorization process (section 3.2.1). One main limitation of *PBIL* is that it deals with a restricted space of distributions, assuming the independence of the genes. In other words, *PBIL* efficiently processes a high-level information, but within a language which might be insufficient to fit complex fitness landscapes. Some evidence for this remark has been presented in [47], considering the Long Path problem [25]: the distribution, even close to one point of the path, hardly meets the narrow optimal region, the path.

EbI actually transmits the genetic material from one individual to its offspring; but it uses its memory to constrain the generation of the offspring. The memory can here be viewed as a gradient, in the sense that it gives preferred directions of move. This approach can fail in two ways: it can suffer from the loss of genetic diversity in the current population, like standard evolution; it might also get irrelevant if the the memory gets too rigid and induces deterministic behaviors, like *PBIL*.

These approaches can be combined in different ways. A loose coupling is obtained by generating two subpopulations, one from each algorithm. Another possibility is to extend *PBIL* to accommodate both memories, that of the best and of the worst individuals. Actually, some *PBIL* variant already memorizes some information from the worst offspring [4] (section 3.2).

The approach investigated in the remainder of this paper rather extends the mechanism of *EbI* and preserves the transmission of the genetic material from one individual to its offspring. This choice is motivated the fact that indeed, the population can follow any fitness landscape, more accurately than any predefined-shaped distribution, — provided the population remains diversified.

The generation of the offspring is extended to account for the two available memories, reflecting the past best and worst individuals.

3. Mimetic evolution

This section starts with an outline of the global scheme proposed, termed *Mimetic Evolution*, then details and discusses its different components.

3.1. Outline

Basically, mimetic evolution alternatively evolves the population using its memories, and updates the memories from the remarkable individuals of the current population. These memories denoted $\mathcal{L}$ (for *Leader*) and $\mathcal{R}$ (for *Repoussoir*) respectively summarize the best and the worst individuals encountered by evolution.

These memories are used to guide the reproduction of individuals, as follows. Let $\mathcal{M}$ be a given memory, and consider all possible offspring Y obtained by flipping a given number of bits in X. These offspring can be ranked with respect to their probability of being generated from $\mathcal{M}$, noted $p(Y|\mathcal{M})$:

$$p(Y|\mathcal{M}) = \prod_{i=1}^{N} (1 - |Y_i - \mathcal{M}_i|)$$

If $\mathcal{M}$ is considered *desirable*, one will prefer offspring Y maximizing $p(Y|\mathcal{M})$; if on the contrary, $\mathcal{M}$ is considered *undesirable*, one will prefer offspring minimizing $p(Y|\mathcal{M})$. In other words, $\mathcal{M}$ induces a polarization of the search space which constrains the moves of X depending on its interpretation of $\mathcal{M}$. Metaphorically, X will decide to imitate or reject the memory, or *model*: its reproduction depends on its *social strategy.*

Practically, mimetic evolution uses a single evolution operator, termed *mimetic mutation*, to evolve the current population (section 3.4). Mimetic mutation depends on the individual at hand, the chosen social strategy, and the two memories $\mathcal{L}$ and $\mathcal{R}$.

Let us first detail how the memories are constructed.

3.2. Memorization and models

In mimetic evolution as in *PBIL* and *EbI*, the memory of past remarkable individuals in $\{0,1\}^N$ is represented as an element of $[0,1]^N$. The terms of *model* and *memory* are indifferently used in the following.

The models can be used in two ways.

3.2.1. Requirements for a distribution-like model

In *PBIL*, the model is used to generate new individuals from scratch. Indeed, any element $\mathcal{M}$ in $[0,1]^N$ can be interpreted as a distribution on $\{0,1\}^N$:

$$Proba(X_i = 1) = \mathcal{M}_i$$

At one extreme lies the uniform (or most general) distribution ($\mathcal{M}_i = .5$ for $i = 1 \ldots N$); at the other extreme lie degenerate (or most specific) distributions, when $\mathcal{M}$ is in fact a boolean individual[5].

[5]From a mathematical point of view, no $\mathcal{M}_i$ ever takes the value 0 or 1, which means that the probability for generating *any* boolean individual according to distribution $\mathcal{M}$ is never null. From a computational point of view, however, evolution never recovers its diversity when $\mathcal{M}$ gets too close to a boolean individual.

PBIL has investigated two heuristics to prevent the premature convergence of $\mathcal{M}$ [4]. One is to update $\mathcal{M}$ from the two best offspring, instead of the best one[6]. The second one is to add small gaussian perturbations to a small percentage of the $\mathcal{M}_i$; this way, the distribution of the offspring is durably perturbed.

3.2.2. Requirements for a gradient-like model

In *Evolution by Inhibitions* and *Mimetic Evolution*, the interpretation of $\mathcal{M}$ is radically different: the goal is to modify a given individual X, that is, to find a relevant climbing direction (the question of the climbing step will be addressed in section 3.4). $\mathcal{M}$ is more to be seen as a gradient than a distribution.

Assume that $\mathcal{M}$ represents a region to avoid. One thus wants the offspring of individual X to be as far away from $\mathcal{M}$ as possible. This leads to preferably mutate the bits that do not discriminate X from $\mathcal{M}$ ($|X_i - \mathcal{M}_i|$ is comparatively small). Inversely, if $\mathcal{M}$ represents a desirable region, one preferably mutates the bits discriminating X and $\mathcal{M}$, thereby producing offspring closer to $\mathcal{M}$ than was X.

The offspring of an individual X are thus constrained from model $\mathcal{M}$, and the "interpretation" of $\mathcal{M}$ by X, or *social strategy* of X. The social strategy defines a direction of evolution, as it induces a preference on the moves of any individual. However, this direction might turn irrelevant either at the individual, or at the population level:

• The direction deduced from a model and a social strategy is irrelevant when it draws the individual away from the optimum. For instance, imitating the *Repoussoir* is a priori (and also experimentally) irrelevant: think of climbing a gradient on the wrong side.
A social strategy can also becomes irrelevant in particular circumstances. For instance, imitating the *Leader* when evolution stagnates causes premature convergence. The flexibility of the *Leader* depends on the progress of evolution: the less flexible the model, the more deterministic the moves, and the less evolution can progress.
In the same vein, rejecting the *Repoussoir* can cause oscillations around the optimum. The *Repoussoir* pushes the individual in the right direction as long as the individual is "between" the *Repoussoir* and the optimum. If the individual passes the optimum, rejecting the *Repoussoir* will draw the individual away from the optimum — until the *Repoussoir* is behind again.

• The direction might also be irrelevant at the population level. This happens when the population is symmetrical with respect to the model. Any move based on the model (either to imitate or to reject it) in fact exchanges the individuals without much modifying the population. For instance, assume the population belongs to the two schemas $01***..$ and $10***...$ Assume that $\mathcal{M}$ reflects this symmetry, with $\mathcal{M}_1 \sim \mathcal{M}_2 \sim .5$, and assume further that at least two bits must be modified (see section 3.4). If the strategy is to reject $\mathcal{M}$, bits 1 and 2 will not be modified — and the offspring still belong to the same schema as it parent. But, if the strategy is to imitate $\mathcal{M}$, bits 1 and 2 are both modified — and the offspring again belong to the initial

[6] This way, $\mathcal{M}_i$ is generalized for all bits i where the best two offspring differ. It is also possible to use the worst offspring, in order to specialize $\mathcal{M}_i$ for all bits where the best two offspring differ from the worst one.

schemas: the model will remain symmetrical, until the symmetry is broken by some external factor.

But all such deadlocks and cycles are less likely to occur, if two models are considered: the influence of each model acts as a perturbation with regard to the other, and enhances the diversity of the offspring when the other model gets stuck. In other words, the more models, the more specific they can harmlessly be.

3.2.3. Updating the models

From these considerations, we construct rather specific models: the *Leader* model $\mathcal{L}$ is constructed by relaxation from the best offspring in the current population and the *Repoussoir* model $\mathcal{R}$ is constructed from the worst two offspring in the current population.

Relaxation is commonly used (e.g. in neural nets) to ensure smooth updates and prevent numerical oscillations. It reads:

$$\mathcal{M}^{t+1} = (1-\alpha)\mathcal{M}^t + \alpha\Delta\mathcal{M} \tag{1}$$

where $\mathcal{M}$ is the model, $\alpha \in [0,1]$ is the relaxation factor, or memory fading; $\Delta\mathcal{M}$ is computed from the current state of the system, defining the selectivity of the memory.

Table 1 illustrates on a simple 5-bits example how $\mathcal{L}$ and $\mathcal{R}$ are respectively updated from the best and the two worst offspring.

	1	2	3	4	5	Fitness
X	1	1	0	0	0	high
S	0	0	0	1	0	low
T	1	0	1	1	1	low
$d\mathcal{R}$	0.5	0	0.5	1	0.5	

$$\mathcal{L}^{t+1} = (1-\alpha)\mathcal{L}^t + \alpha X$$
$$\mathcal{R}^{t+1} = (1-\alpha)\mathcal{R}^t + \alpha d\mathcal{R}$$

Table 1:*Individuals and Models*

The same relaxation factor α is used to update both the *Leader* and the *Repoussoir*, though the worst offspring are likely more diversified than the best one. However, complementary experiments (not reported in the paper) show that using different relaxation factors for the *Leader* and the *Repoussoir* does not make any significant difference.

3.3. Mimetic mutation

Let X and m thereafter denote the current individual and the number of bits to mutate in X (details on the setting of m in section 3.4).

3.3.1. Social strategies based on one model

Assume first that X is evolved from a single model $\mathcal{M}$.

For each bit i, the probability p_i of mutating bit i depends on whether X_i is close to to $\mathcal{M}_i$, and whether $\mathcal{M}$ is considered positively or negatively. As already stated, if $\mathcal{M}$ is desirable, p_i increases with $|X_i - \mathcal{M}_i|$: the more X_i differs from $\mathcal{M}_i$, the more X_i should be modified. If, on the opposite, $\mathcal{M}$ is not desirable, p_i decreases as $|X_i - \mathcal{M}_i|$ increases.

In preliminary experiments with a no-memory setting ($\alpha = 1$) [47], we used a roulette wheel to select the bits to mutate, with p_i being proportional to $|X_i - \mathcal{M}_i|$. But the roulette wheel selection shows inefficient when $\mathcal{M}$ actually memorizes several generations of worst offspring ($\alpha < 1$), and for large sized problems: after a few hundreds of generations, most $\mathcal{M}_i$ are close to 0 or 1 up to 10^{-5}. In such a context, no general and accurate way to compute p_i from $|X_i - \mathcal{M}_i|$, was found.

We therefore use a selection by tournament among bits: each bit to mutate is selected as the bit i_j optimizing $|X_i - \mathcal{M}_i|$ among T bits drawn with uniform probability in $\{1 \ldots N\}$. By the way, the same mechanism can be used to find offspring *imitating* $\mathcal{M}$ (by mutation of bits *maximizing* $|X_i - \mathcal{M}_i|$) or *rejecting* $\mathcal{M}$ (by mutation of bits *minimizing* $|X_i - \mathcal{M}_i|$).

```
for each l = 1 ..  m           // number of bits to mutate
   Select T bits i_1 ... i_T uniformly in 1...N              c
   Mutate X_{i_j} with i_j = Arg optimum {|X_{i_k} - R_{i_k}|,  i_k = i_1 ... i_T}
```

3.3.2. Social strategies based on two models

Consider now the two models constructed by evolution, the *Repoussoir* and the *Leader*. A most natural social strategy, referred to as the *Sheep* strategy, is to imitate the *Leader* and reject the *Repoussoir*. This strategy can be accommodated within the above tournament selection, now based on the maximization of $|X_i - \mathcal{L}_i| - |X_i - \mathcal{R}_i|$.
But, more generally, one can choose to imitate, reject, or even ignore independently each model. The selection of the bits to mutate then again proceeds by tournament, now optimizing criterion:

$$\delta_L |X_i - \mathcal{L}_i| + \delta_R |X_i - \mathcal{R}_i|$$

where δ_R (respectively δ_L) indicates whether X is to imitate, reject or ignore $\mathcal{R}$ (respectively $\mathcal{L}$):

- $\delta_M > 0$ means that X imitates $\mathcal{M}$;
- $\delta_M < 0$ means that X rejects $\mathcal{M}$;
- $\delta_M = 0$ means that X ignores $\mathcal{M}$;

A social strategy can thus be represented as a pair of coefficients (δ_R, δ_L). A particular case, termed *Ignorant* strategy, ignores both models ($\delta_R = \delta_L = 0$). For all other strategies, the parameters (δ_R, δ_L) can be normalized by requiring $\delta_R^2 + \delta_L^2 = 1$, without modifying the tournament-based selection. Every strategy except the Ignorant one, is thereafter represented as angle θ in $[0, 2\pi]$, with $(\delta_R, \delta_L) = (cos\theta, sin\theta)$.

Figure 1 represents all social strategies but the Ignorant on the unit circle. By convention, angle 0 is associated to ($\delta_R = 1, \delta_L = 0$), that is, imitating the *Repoussoir* and ignoring the

Leader. Angle $\pi/2$ corresponds to $(\delta_R = 0, \delta_L = 1)$, that is, imitating the *Leader* and ignoring the *Repoussoir*.

Some social strategies have been given nicknames for the sake of convenience. We distinguish mainly:

- The *Entrepreneur* imitating the *Leader* and ignoring the *Repoussoir* (angle $\pi/2$);
- The *Sheep* imitating the *Leader* and rejecting the *Repoussoir* (angle $3\pi/4$);
- The *Phobic* rejecting the *Repoussoir* and ignoring the *Leader*; mimetic evolution with a phobic strategy just reproduces *Evolution by Inhibitions* (angle π);
- The *Lone Rider* rejecting both the *Leader* and the *Repoussoir* (angle - $3\pi/4$);
- The *Rebel* rejecting the *Leader* and ignoring the *Repoussoir* (angle - $\pi/2$).

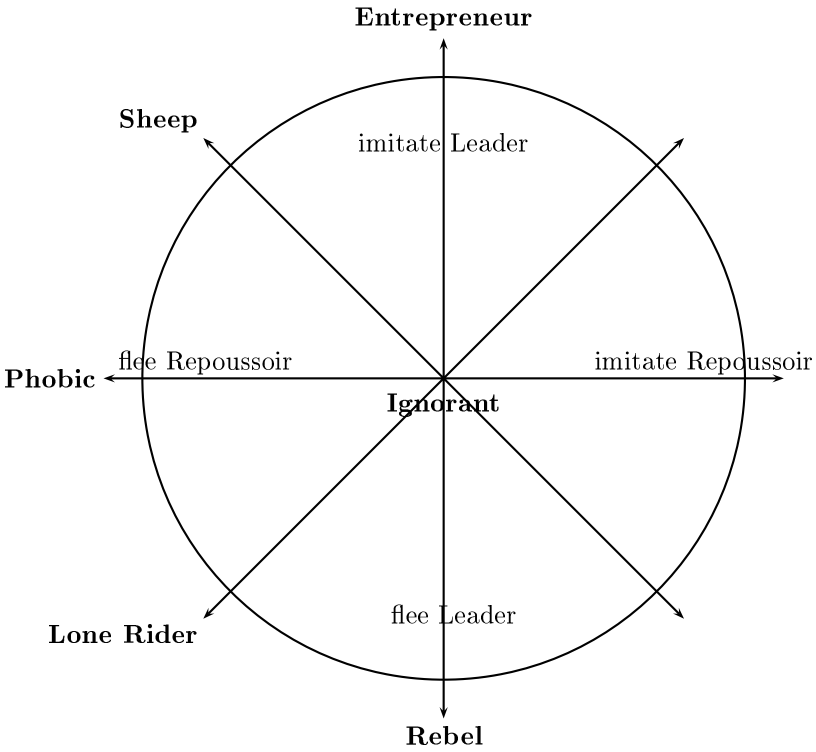


Figure 1. *Social Strategies*

The *Leader* and the *Repoussoir* together with the current individual, can be thought of as a system of coordinates. A social strategy is a direction in this system of coordinates: the corresponding angle constrains the trajectory of the individual, intended as its possible offspring. The constraint can be made more or less severe, making the trajectory more or less deterministic, by tuning the tournament size T; this parameter controls the variance of mimetic mutation, i.e. the predictability of the trajectory in a given situation. The ignorant strategy, setting no constraint on the offspring, serves as reference to determine the relevance of the other strategies.

In the remainder of the paper, we restrict ourselves to considering a single social strategy for all individuals, fixed along evolution and with fixed tournament size T. At first sight, making all the population follow a single fixed direction, with the same degree of predictability, dictates the system a poor evolution. Still, this direction is defined with respect to the individual and the system of coordinates, which itself evolves together with the population.

3.3.3. Single-Model Dynamics

Consider first a single model $\mathcal{M}$, and let M denote the boolean individual closest to $\mathcal{M}$. Let $\mathcal{S}$ be the set of bits candidate to mutation in the current individual X. The dynamics of mutation depends on how $\mathcal{S}$ varies using the feedback provided by the model.

When X imitates $\mathcal{M}$, two kinds of bits are preferably mutated: the bits which discriminate X from M ($X_i \neq M_i$), and the most general bits in $\mathcal{M}$($\mathcal{M}_i$ close to .5). The discriminant bits were good candidates to be mutated in the previous steps; this is only possible if $\mathcal{M}_i$ was relatively general. Inversely, $\mathcal{M}_i$ can be general only if individuals with $X_i \neq M_i$ were recently considered to update $\mathcal{M}$. In summary, the more a bit was recently mutated, the more it is mutated.

The set $\mathcal{S}$ of bits that are candidates to mutation does not depend on the individual, but rather converges to a fixed set, making mimetic mutation unable to explore the whole search space (violating the ergodicity requirement [35, 40]).

If $\mathcal{M}$ stands for the *Leader* (*Entrepreneur* strategy), the population converges toward M. Incidentally, mimetic mutation here resembles the BSC operator of Syswerda [50]: but BSC actually uses the average of the current population to evolve the individuals, instead of the memory $\mathcal{L}$.
If $\mathcal{M}$ stands for the *Repoussoir*, the loss of genetic diversity does not occur as the population cannot converge toward M; rather, the strategy oscillates between optima closest to M.

When X rejects $\mathcal{M}$, the bits preferably mutated are such that $X_i = M_i$, and $\mathcal{M}_i$ is as specific (close to 0 or 1) as possible. Thereby, mimetic mutation modifies all bits which have not been recently modified, neither in the individual nor in the model. In any case, the set $\mathcal{S}$ of bits candidates to mutation is much larger than in the imitation strategy: almost all bits $\mathcal{M}_i$ get more specific in each generation (all bits except at most $2 \times m$, that is, the maximum number of individuals used to update $\mathcal{M}$ times the maximum number of bits modified in each individual), and the more specific a bit, the more likely it is mutated.

There is one only case where $\mathcal{S}$ converges: when $\mathcal{M}$ recommends a move and the resulting offspring are considered neither to update the model nor the population[7]. As the mechanism is then deprived from any feedback, it can only persist in its wrong decision.

[7] This occurs when the offspring are not fit (otherwise they would be retained in the population) and the model is the *Leader* (otherwise, the bad offspring would be accounted for in the model).

The reject strategy enforces the genetic diversity of the population, for the following reason. A bit which discriminates some individuals of the population is unlikely to be mutated: recent moves on this bit have been registrated; hence the corresponding $\mathcal{M}_i$ is more general than for the other bits.

The weakness of the reject strategy thus comes from the fact that it prevents the recombination of current individuals and the convergence of the population.

3.3.4. Two-Model Dynamics

When both models are considered, they altogether provide any desired feedback on the previous moves: unsuccessful moves are memorized within the *Repoussoir*, successful moves are memorized within the *Leader* and the population. Three categories of bits might then be roughly identified, and the success of evolution relies on a sufficient turn-over among these categories:

- To preserve exploration, one must be able to mutate bits which have not been recently modified. Such bits are characterized by both $\mathcal{L}_i$ and $\mathcal{R}_i$ specific. Mutating these bits implies that either one or both models must be rejected (avoid the strategies in the quadrant $[0, \pi/2]$). Typically, no bit should stay in that category for too long!
- The bits which have been recently modified successfully (i.e. the resulting offspring have been kept in the population) are characterized by both $\mathcal{L}_i$ and $\mathcal{R}_i$ general. Mutating such bits is desirable as far as recombining the individuals is desirable, i.e. when the building blocks hypothesis holds. In this case, the social strategy must incorporate some imitation of the *Leader* (prefer the quadrant $[\pi/2, \pi]$).
- The bits which have been recently but unsuccessfully modified (the offspring have **not** survived), are characterized by $\mathcal{L}_i$ specific and $\mathcal{R}_i$ relatively general. The decision of mutating these bits depends on the gap between the current optimum and the next basin of attraction.
  If there are large gaps between local optima, some obstinacy is required: one must be able to consider again bits which have been recently unsuccessfully modified. Practically, the social strategy must to some extent reject the *Leader* (prefer the quadrant $[\pi, 3\pi/2]$).
  Otherwise, a shallow and hopefully faster exploration can be achieved by discarding the bits which have been recently unsuccessfully modified (quadrant $[\pi/2, \pi]$).

Obviously, there exists nothing like a universal strategy; each strategy could prove the most adequate in some context. Furthermore, the social strategy is not the only factor ruling out the balance exploitation/exploration achieved by mimetic mutation: While mimetic mutation primarily concentrates on the choice of the bits to mutate, another key factor is the choice of the number of bits mutated in each individual.

3.4. Mimetic strength

Binary mutation traditionally uses a very low mutation rate [17], though good results obtained with high-rate mutation have also been reported [27, 32]. However, as the only operator of mimetic evolution is mutation, the mutation rate must certainly be high. Besides, the mutation rate governs the balance between exploration/exploitation achieved by mimetic mutation: exploration is encouraged for high mutation rates and exploitation for low mutation rates. Typically, mutating one only bit in any individual will cause evolution to get trapped in the nearest local optimum, though it is theoretically proved to be faster on unimodal problems [15].

In binary mutation, the probability of mutation is most usually constant (especially when it is very low). However, it can also be either dynamically adjusted, or adapted at the population level, or self-adapted at the individual level. Based on the different adaptation techniques used for mutation in evolutionary parameter optimization (again, see [22] for a survey), we have investigated the following heuristics to adjust m^t, the mutation rate at generation t:

- Constant scheme: m^t is set to a constant value. The adjustment of m can be done by considering mimetic mutations with different values of m as different exclusive operators; their probability can be adjusted them *à la* Davis [8] by rewarding the values leading to good offspring.
- Hyperbolic scheme: m^t decreases from an initial value m^0 to 1, according to the hyperbolic schedule borrowed from [3]. The value for m^t for all individuals is

$$m^t = (\frac{1}{m^0} + t \cdot \frac{1 - \frac{1}{m^0}}{T - 1})^{-1} \tag{2}$$

  where T is the maximum number of generations and t denotes the current generation.

- 1/5th rule: Following [38], m^t is geometrically increased (resp. decreased) by a factor 1.2 if the number of offspring more fit than their parent in the last 10 generations is greater (resp. smaller) than 1/5.
- Self-adaptive scheme: m is encoded in the individual and evolved according to the *Obalek's rule* described in [2]:

$$m^{t+1} = \left(1 + \frac{1 - m^t}{m^t.exp(\gamma\ .\ N(0,1))}\right)\ ;\ \gamma = \frac{0.5}{\sqrt{2\sqrt{N}}} \tag{3}$$

In the three latter cases, the continuous value of m^t was transformed into an integer value (as the tournament-based mechanism of mutation requires to know in advance the number of bits to mutate), either by taking its integer part, or by selecting an integer value m from a Poisson distribution of parameter m^t [48] (it is well known that the binomial distribution $B(N, m)$ tends to the Poisson distribution $P(\lambda)$ if Nm goes to λ as N goes to infinity).

In all cases, m is lower-bounded by 1 to guarantee that mimetic mutation is effective.

3.5. The algorithm

Mimetic mutation is embedded into a $(\lambda + \mu)$ evolution strategy [45]. Besides the method used to set m^t, mimetic evolution includes five parameters:

- The population size μ and birth rate λ — in $\mathbb{N}$
- The relaxation factor α used to update $\mathcal{L}$ and $\mathcal{R}$ — in $[0, 1]$
- The tournament size T — in $\mathbb{N}$
- The social strategy θ — in $[0, 2\pi]$

Here is the pseudo code of the algorithm.

Mimetic Evolution

```
Parameters:  μ, λ, α, T, Strategie

Initialize μ parents X_1 ... X_μ in {0,1}^N
Compute F(X_i), i = 1...N.
Set L_i = R_i = .5, i = 1...N
Repeat
   for each X_i = X_1 ... X_μ
      for each j = 1 ... λ/μ
         Offspring = Mimetic Mutation(X_i, L, R)
         Compute Fitness(Offspring)
   End while
   Sort parents + offspring
                                                    Update the models
   dL = best of { parents + offspring}
   L = (1 − α).L + α.dL
   dR = average of the two worst { parents + offspring}
   R = (1 − α)R + αdR
                                                    Update the population
   Parents = best μ individuals in { parents + offspring}
Until Stop
```

Mimetic Mutation$(X, \mathcal{L}, \mathcal{R})$

```
Y = X
For i = 1...m()              m() returns the number of bits to mutate
   k_i = Tournament(X, L, R)
   Y_{k_i} = 1 − Y_{k_i}
End for
return Y
```

Tournament$(X, \mathcal{L}, \mathcal{R})$ *Depends on the global social strategy* θ

For j = 1..., T
  select k_j randomly in $\{1, \ldots N\}$
  $p(k_j) = cos(\theta) \times |X_{k_j} - \mathcal{R}_{k_j}| + sin(\theta) \times |X_{k_j} - \mathcal{L}_{k_j}|$
Return $k_j = argmax \ \{ \ p(k_1) \ldots p(k_T) \ \}$

4. Goal of experiments

As no optimization scheme could possibly dominate all other schemes [54], any new evolution algorithm should be presented together with its "niche".

Mimetic evolution was designed for large-sized problems, for the following reason. It is based on a structure of control, ordering the possible moves of the individuals. As any control, this entails some computational overhead that can only be balanced if selecting the moves at random is *very* often unsuccessful. This occurs iff the size of the space is large enough; otherwise, exploring the neighborhood of the individuals at random might do as well.

In this large-space context, we shall focus on the two main choices of mimetic evolution: how to adjust the mimetic strength and how to choose a social strategy. After early preliminary experiments, these two points are briefly discussed, and the experimentation goal is then defined.

4.1. Mimetic Strength: Simple options are retained

Preliminary experiments showed that the on-line adjustment of the mimetic strength m^t was not robust. Rather, all heuristics used (section 3.4) including the Davis-like approach, the 1/5th rule and self-adaptation, were found to fail: all rapidly lead to set $m^t = 1$, causing the premature convergence of evolution.

In retrospect, this failure can be explained from the fact that on-line adjustment rewards options bringing the most improvements, rather than the most significant ones [9]. The above heuristics thus show risk-adverse and favor the conservative option $m^t = 1.$, which produces the most improvements on the whole. Favoring conservative options might still bring improvements in a continuous search space. But in binary search spaces, the strong causality principle is violated (there is nothing like a "very small" mutation), and conservative options simply are inactive. No wonder then that the convergence results of evolution strategies [45, 1] cannot be transposed.

Furthermore, fixed-step mutation (mutating exactly m^t bits for all individuals at time t) shows more effective than variable-step mutation (mutating on the average m^t bits in the population, distributed on the individuals according to a Poisson law).
This is unexpected, as fixed-step mutation severely restricts the distribution of the offspring

from the current population. Traditionally, mutation must be able to make arbitrarily large steps [35], to prevent evolution from being trapped in some local minima. In the meanwhile, making large steps can be beneficial as short cuts can be discovered.

However, experiments demonstrate that fixed-length mutation does not cause evolution to stop during the observations (limited to 200,000 evaluations) — this was unexpected. Everything happens as if interesting offspring can always be found at a given distance of some individual of the population! In the meanwhile, fixed-step mutation appears faster than variable-step mutation: if it prevents from discovering short cuts, it also saves a lot of bad moves.

In the experiments, two possibilities have then been investigated. One is to set m^t to a constant value chosen in $\{1, 3, 5, 7, 9\}$; the other is to adjust m^t according to a hyperbolically decreasing schedule (section 3.4), starting from $m^0 = N/2$ and reaching $m = 1$ at the end of evolution.

4.2. Social Strategies and Significance of Models

Mimetic evolution will thus be experimented on large sized problems, with fixed or hyperbolic mimetic strength, and compared in this context to reference genetic algorithms, evolution strategies, and *PBIL*.

Experiments are designed to answer the following questions:

Q1 *Relevance of the models.*
A particular test is to compare mimetic evolution based on actual models, with what happens with void models. Whenever the Ignorant strategy (using no models, or void models) outperforms all other strategies, this means that the models hinder, more than guide, evolution.

Q2 *Robustness of the scheme.*
Assuming that there exists a social strategy outperforming the ignorant strategy, the next question regards the robustness of mimetic evolution: does there exist a wide range of social strategies valid for a given problem; does there exist a social strategy valid for a range of problems; how do they compare to reference algorithms (canonical genetic algorithms, evolution strategies, PBIL).

Q3 *Optimal control of the scheme.*
This question regards what is the optimal strategy for a given problem. Hopefully, problems having the same optimal strategy present other similar features. Hence the optimal social strategy for a problem could then be used as a difficulty criterion.

5. Experiments

We first describe the problems considered and the reference algorithms. The experiment setups are then detailed. A global overview of the results over all functions is presented and the general trends are discussed. The section ends with some conclusions on the "niche" of Mimetic Evolution.

5.1. Problems

All experimentations consider the optimization of functions of 100 continuous variables, discretized through a binary or a Gray coding. Function F_2 is taken from [4]. The Griewank, Rosenbrook and Rastrigin functions have been thoroughly studied in the literature up to 20 or 50 continuous variables [55].

- **Function** F_2 (maximization). Search space $\{0,1\}^{900}$.
  The domain of each continuous variable x_i is set to $[-2.56, 2.56]$; x_i is coded on 9 bits. The optimum is 10^7, reached for $x_i = 0$.

$$y_1 = x_1 \qquad F_2(x_1, \ldots x_{100}) = \frac{100}{10^{-5} + \sum_{i=1}^{100} |y_i|}$$
$$y_i = x_i + sin(y_{i-1}), \ i = 2 \ldots 100$$

- The **Griewank function** (minimization). Search space $\{0,1\}^{1400}$.
  The domain of each continuous variable x_i is set to $[-100, 100]$; x_i is coded on 14 bits. The optimum is 0, reached for $x_i = 0$.

$$Griewank(x_1, \ldots x_{100}) = 1 + \frac{1}{4000} \sum_{i=1}^{100} x_i^2 - \prod_{i=1}^{100} cos(\frac{x_i}{\sqrt{i}})$$

- The **Rosenbrook function**(minimization). Search space $\{0,1\}^{1400}$.
  The domain of each continuous variable x_i is set to $[-30, 30]$; x_i is coded on 14 bits. The optimum is 0, reached for $x_i = 1$.

$$Rosenbrook(x_1, \ldots x_{100}) = \sum_{i=1}^{100} [100(x_i^2 - x_{i+1}) + (1 - x_i)^2]$$

- The **Rastrigin function**. Search space $\{0,1\}^{1400}$.
  The domain of each continuous variable x_i is set to $[-5.12, 5.12]$; x_i is coded on 14 bits. The optimum is 0, reached for $x_i = 0$.

$$Rastrigin(x_1, \ldots x_{100}) = \sum_{i=1}^{100} [x_i^2 + 10(1 - cos(2\pi x_i))]$$

In the following, the Gray and binary encodings of a same function will be considered as different optimization problems, termed e.g. F_2-binary and F_2-Gray.

5.2. Reference algorithms

The following algorithms will be considered to give the reference results to which the results of mimetic evolution will be compared.

- **SGA**. The first reference algorithm is a simple GA. Several setups have been considered (see below). However, these experiments were rather control experiments, as in the range of setups considered, SGA was consistently outperformed by PBIL – as reported in [4].
- **PBIL**. The second reference algorithm is Baluja's *PBIL*, with same setup as in [4]: population size 100, update of the model based on the two best offspring with relaxation factor α, with possible reinforcement from the worst offspring with relaxation factor $\alpha/2$. Parameter α was varied as detailed below.
- **ES**. The third reference algorithm is a $(\mu + \lambda)$ evolution strategy, where the mutation probability per bit follows a hyperbolic schedule with initial value 1/2 and final value 1/N. This deterministic schedule was chosen after [3], which concludes that the hyperbolic schedule seems more efficient and more robust than both the self-adaptive scheme described in section 3.4, and the fixed mutation rate of $\frac{1}{N}$. The population size μ and natality λ are varied as detailed below.
- **Ignorant**. The last reference algorithm is a $(\mu + \lambda)$ evolution strategy, where the number m of bits mutated in each individual is constant. This actually corresponds to a mimetic evolution following the Ignorant strategy (i.e. without any memory involved). Parameter m is varied as in other mimetic evolution schemes.

The results of both ES and the Ignorant strategy will be examined together as these reference algorithms only differ by the setting of their mutation rate.

5.3. Experimental settings

Each run is allowed an arbitrary number of 200,000 evaluations of the fitness function (in order to compare with the results of [4] in the first place).

Two steps of experiments are then performed: First, few runs (11) of a large numbers of parameter settings are launched. This allows us to evaluate and discuss the general trends (section 5.4). The best settings of all algorithms are then retained, and are studied in more detail (section 5.5).

The range of setups considered is as follows:

- For SGA, the population size is set to 50 or 100, with uniform or 2-point crossover, at rate 0.5, 0.75 or 1.0, and mutation rate of $\frac{1}{N}$, $\frac{3}{N}$ or $\frac{5}{N}$. As SGA is outperformed by the other reference algorithms, its results will not be discussed any further.
- For PBIL, the relaxation factor α is taken in $\{.5, .2, .1, .05, .02, .01\}$. Only the results obtained with values $2, .1$, or $.05$ are then detailed as the other values gave lousy results (except for the Rosenbrook function see section 5.5).

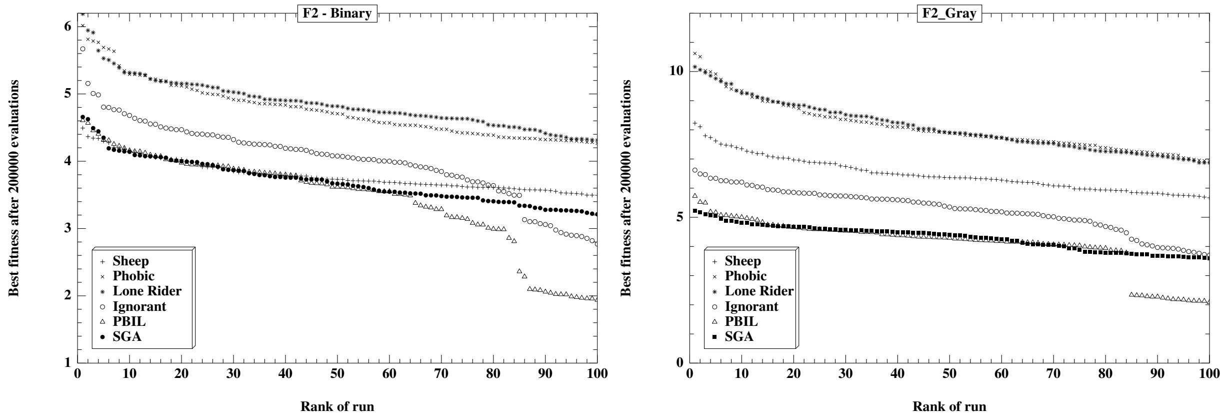
- For all schemes based on $(\mu+\lambda)$-ES (ES, Ignorant and Mimetic Evolution), the population size μ and natality λ are taken in $(1+30)$, $(1+50)$, $(7+30)$, $(7+50)$.
- For the Ignorant and Mimetic Evolution (section 3.3), the mimetic strength m takes values $1, 3, 5$, or 7; the relaxation factor of the models is set to .01 and the tournament size T is set to 50 to keep reasonable the number of options.
- The strategy of Mimetic Evolution is varied in 0, 45, 90, 135, 180, 225, 270, 315 (section 3.3); however, only the Sheep, the Phobic and the Lone Rider strategies (respectively corresponding to angles 135, 180 and 225) gave any good results and will be mentioned in the detailed study.

5.4. Global Overview

This section presents and discusses the offline performance of all algorithms, from the results of 11 runs of each of the settings described in the preceding section. The six algorithms are SGA, PBIL, ES + Ignorant, (the reference algorithms), and the Sheep, the Phobic, and the LoneRider mimetic strategies.

In the following plots (Figures 2 to 5), each dot represents the off-line performance of a single run, i.e. the best fitness reached after 200000 fitness evaluations. The X-axis is simply the rank of the run among all runs for that algorithm, whatever the parameter settings (only the 100 best runs are shown, for readability reasons). Each curve shows how the scheme behaves at its best (beginning of the curve), and the sensitivity of the performance to the setting (the slope).

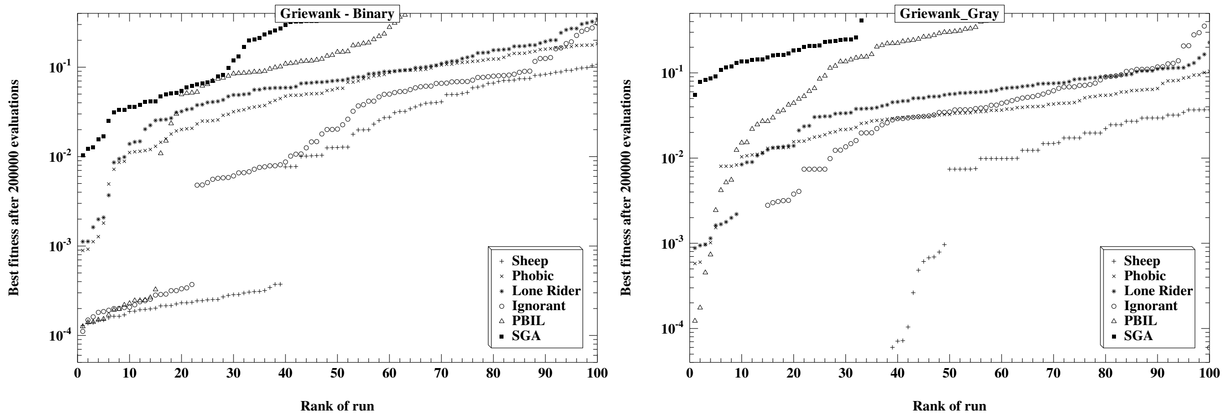
Binary and Gray coding of a same function obviously result in quite different landscapes, which may either hinder or favor evolution [35, 52]. On three out of four functions (F_2, Griewank and Rosenbrook) the Gray coding shows more suited to evolutionary optimization than the binary coding. On the last one (Rastrigin), binary and Gray coding lead to similar results.



(a) Binary coding (b) Gray coding

Figure 2 *Maximization of function F_2, general overview*

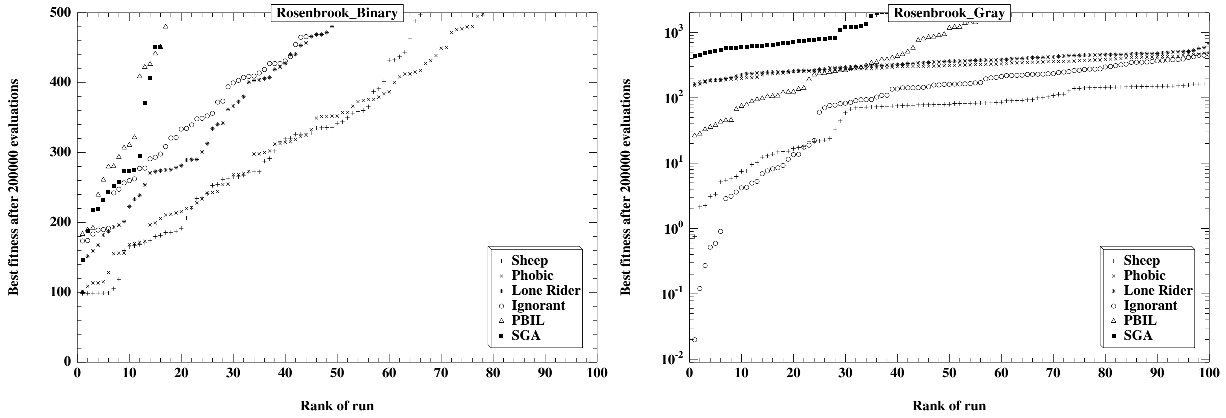
On function F_2, the best performances are obtained for the Phobic and the Lone Rider mimetic strategies for both encodings. These strategies perform equally well, and consistently better than all other strategies. In the meanwhile, functions F_2-binary and F_2-Gray can be considered difficult, as the best results are still very far from the optimum (7 for F_2-binary and 10 for F_2-Gray vs 10^7 for the actual optimum).



(a) Binary coding (b) Gray coding

Figure 3 *Minimization of Griewank function, general overview*

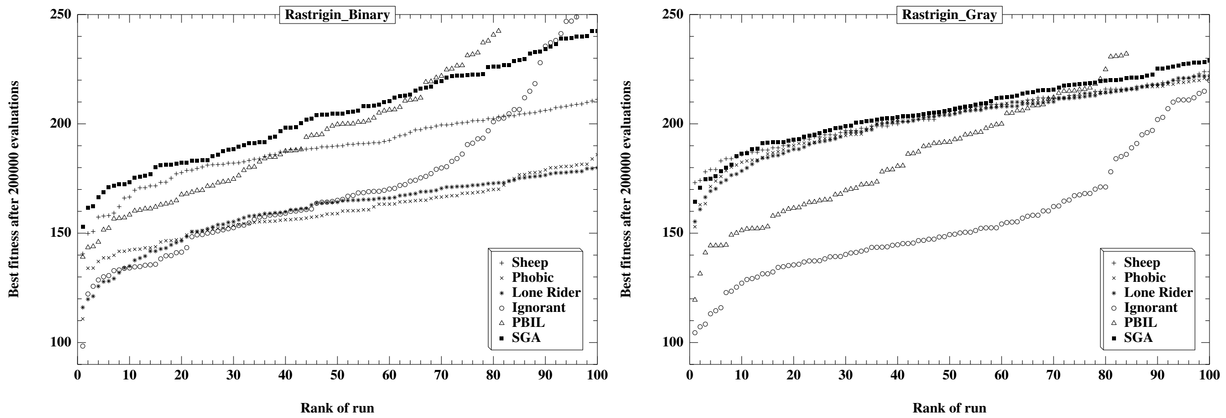
On the Griewank function, whatever the coding, the best performances are obtained for the Sheep strategy, the Ignorant strategy performing almost as good. Functions Griewank-binary and Griewank-Gray can be considered easy, as the optimum is almost reached (beware of the log-scale: 10^{-4} for Griewank-binary and 0 for Griewank-Gray vs 0). Typically, the best 40 runs for the Sheep and the best 15 runs for the Ignorant fall below the bottom line of the drawing.



(a) Binary coding (b) Gray coding (logarithmic scale)

Figure 4 *Minimization of Rosenbrook function, general overview*

On the Rosenbrook function, the best scheme and the difficulty depends on the coding. The Rosenbrook-binary problem is difficult: no algorithm gets to values lower than 100. The best strategies are the Sheep and the Phobic, that strike this barrier value many times. On the other hand, such a barrier does not appear on Rosenbrook-Gray, for which the best strategy is the Ignorant, that reaches values around 10^{-2}. Note that, apart from the 10 best runs, the Sheep performs almost equally well, and can even be considered more robust with respect to the parameter settings.



(a) Binary coding (b) Gray coding

Figure 5 *Minimization of Rastrigin function, general overview*

The Rastrigin problem is difficult for both encodings: no algorithm gets to values lower than 100. However, the picture is different in both cases: On Rastrigin-binary, the best strategies are the Ignorant and the Lone Rider, closely followed by the Phobic. Again, the Ignorant strategy seems less robust with respect to the parameter settings. On Rastrigin-Gray, the best strategy is by far the Ignorant, followed by PBIL, whereas all mimetic strategies perform equally bad.

According to the respective performances of the schemes, one distinguishes three categories among the eight test problems considered here:

- Problems on which the Sheep strategy performs comparatively well. Such problems (Griewank-binary, Griewank-Gray, Rosenbrook-Gray) tend to be rather easy: the problem can be solved by iteratively memorizing the optimum and sampling its neighborhood.
- Problems on which the Lone Rider strategy performs well. Such problems (F_2-binary, F_2-Gray and Rastrigin-binary) tend to be difficult: Indeed, following the Lone Rider amounts to somehow fleeing the past optima; if this shows appropriate, the problem is in some sense deceptive.
- Problems on which the Ignorant strategy performs well. Such problems (Rastrigin-binary or Rastrigin-Gray) are difficult too, but the way memory is used (e.g. by mimetic evolution) seems to rather mislead than guide evolution.

5.5. Detailed comparisons

The previous section gave a general idea of how the different schemes behave at their best. We now study in more detail the best settings of mimetic evolution and compare them with the best reference algorithms on each problem. Each evolution scheme is evaluated from the average or median best performance out of 21 runs. As recommended by [14], the median should be preferred to the average whenever the fitnesses involved show very different orders of magnitude (e.g. when the results get close to the optimum at 0). The criterion used here was to present the average and standard deviations in general, and to use the median whenever the standard deviation was high (not significantly smaller than the average value). When the standard deviation was small, the average and the median results were close in all cases presented here anyway.

5.5.1. Function F_2

Table 2 below presents the offline reference results for F_2 problems, while Figure 6 shows the median online evolution for ES and Ignorant algorithms. Table 3 and Figure 7 show the same results obtained by the mimetic algorithms.

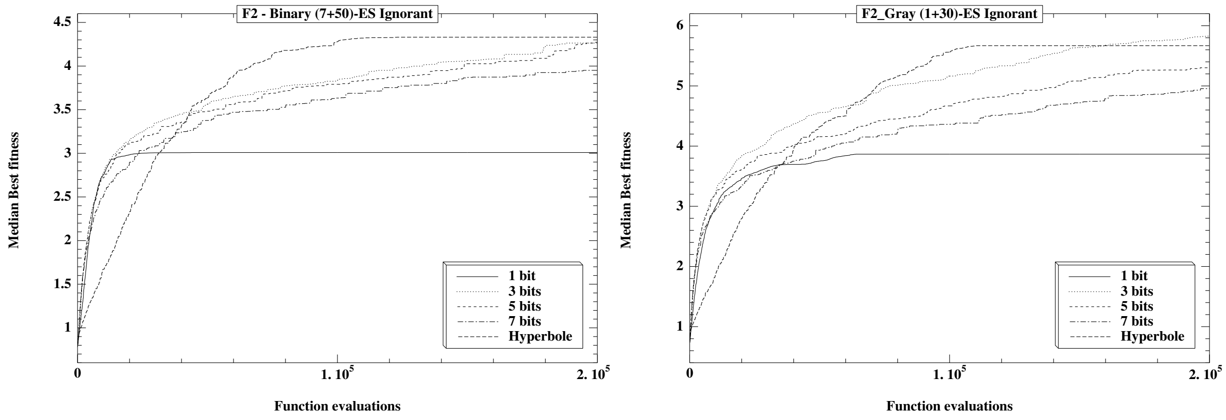
	PBIL			ES	IGNOR			
	.05	.1	.2	(Hyp)	1	3	5	7
Bin.	3.73 (.34)	**4.63** (.40)	3.99 (.32)	4.40 (.43)	3.07 (.19)	4.31 (.32)	4.25 (.35)	3.98 (.25)
Gray	4.07 (.27)	5.35 (.24)	4.66 (.30)	5.65 (.35)	3.87 (.24)	**5.96** (.27)	5.41 (.35)	4.99 (.35)

Table 2: *Reference results for F_2; Average offline results (standard deviation) over 21 runs of 200,000 evaluations.*

On this problem, the best reference algorithms are PBIL and the Ignorant. For PBIL, the result is very sensitive to α. For the Ignorant, it is sensitive to the number m of bits to mutate. As expected, $m = 1$ leads to bad results (for $m = 1$, the Ignorant resembles a standard Hill-Climber). Best performances are obtained for $m = 3$ for both encodings. The performances do not depend much on the population size μ and natality λ.

	Mutation	1	3	5	7	Hyp.
Binary	Sheep	2.91 (.16)	2.48 (.33)	3.79 (.30)	3.91 (.35)	3.90 (.33)
	Phobic	3.01 (.20)	4.68 (.35)	4.77 (.51)	4.60 (.38)	4.56 (.41)
	Lone Rider	3.16 (.23)	**4.97** (.52)	**4.99** (.39)	4.51 (.34)	4.51 (.37)
Gray	Sheep	4.31 (.26)	6.34 (.48)	6.10 (.44)	5.84 (.43)	5.76 (.40)
	Phobic	4.28 (.30)	**8.38** (.77)	7.09 (.54)	6.57 (.40)	6.93 (.50)
	Lone Rider	4.20 (.2)	**8.43** (.58)	7.25 (.46)	6.32 (.41)	6.78 (.61)

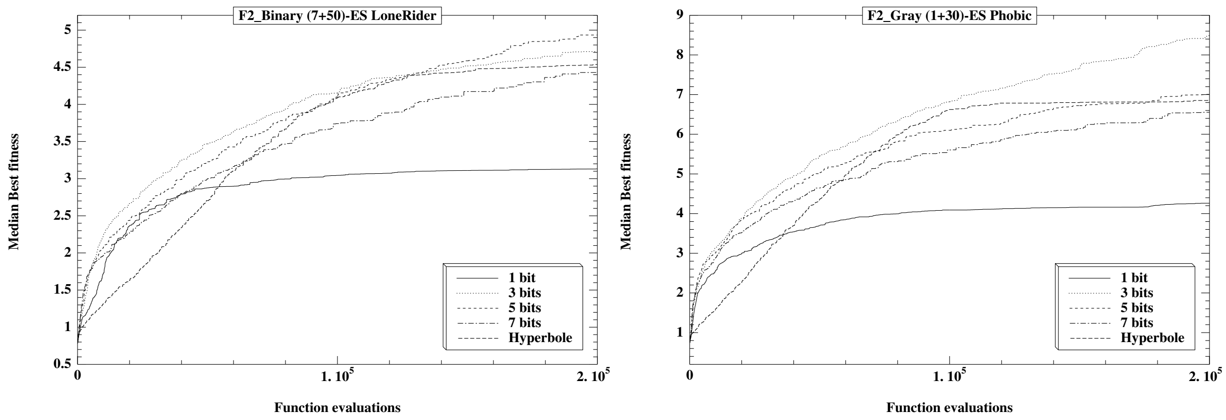
Table 3: *Mimetic results for function F_2; Average offline results (standard deviation) over 21 runs of 200,000 evaluations.*



(a) Binary coding (7+50) (b) Gray coding (1+30)

Figure 6 *ES/Ignorant on function* F_2

The best mimetic strategies are the Lone Rider and the Phobic, which behave equally well on both problems, as in the general overview. The results similarly depend on m, with best results obtained for $m = 5$ (with binary coding) and $m = 3$ (with Gray coding). The overall performance is more sensitive to the population size and natality μ and λ than for the Ignorant strategy.



(a) Binary coding, LoneRider, (7+50) (b) Gray coding, Phobic, (1+30)

Figure 7 *Mimetic Evolution on function* F_2

Figures 6 and 7 show some similarities, especially regarding the behavior of the Hyperbole (m follows a hyperbolic schedule decreasing from $N/2$ to 1). After a bad start (mutation is too active in the early generations), the hyperbole curve soon catches up all other curves and passes them. But in the second half of evolution, the hyperbole curve gets stuck (almost horizontal), and is passed by some fixed schemes. This is due to the fact that the actual strength of mutation is one.

However, a clear effect of the mimetic strategies is to increase the slope of all curves – and this is even clearer for the 1-bit and Hyperbole curves, which definitely end horizontally for the ignorant strategy (Figure 6) while slowly but steadily increasing for the mimetic strategy (Figure 7).

5.5.2. Griewank Function

As in the preceding section, Table 4 and 5 below below presents the offline results for Griewank problem, while Figure 8 and 9 show some plots of online median results.

On this problem, the best reference algorithms are ES and the Ignorant. For both codings, the lower m the better: surprisingly, best reference results are obtained for $m = 1$. The hyperbolic ES catches up the Ignorant with some delay. This, added to the fact that the Ignorant almost finds the optimum, suggests that Griewank-binary is actually rather easy. The performances of the Ignorant are rather sensitive to μ and λ.

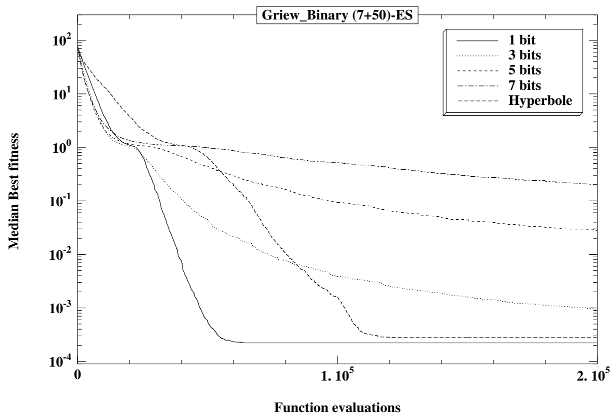
	PBIL			ES	IGNOR			
	.05	.1	.2	(Hyp)	1	3	5	7
Bin.	127 (9)	14.3 (5.3)	4.95 (*)	**0.028** (*)	**0.022** (*)	0.95 (*)	2.79 (*)	21.5 (9.3)
Gray	119 (3.5)	29 (8.4)	2.3 (*)	**1.47** (*)	**1.23** (*)	2.96 (*)	3 (*)	5.15 (2.5)

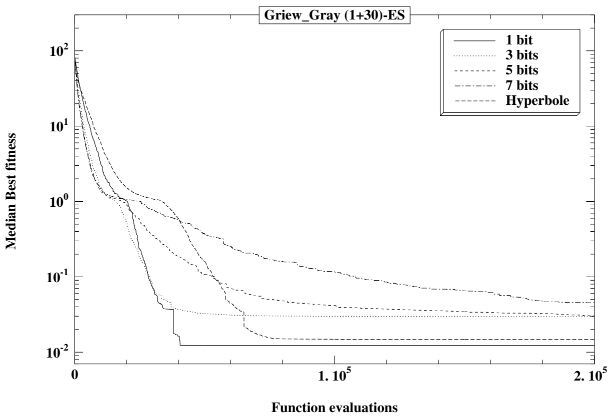
Table 4: *Reference results for Griewank function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations, except for cases (*) where the averages and standard deviations are non-significant; the figure is then the* **median** *of the 21 runs. All figures have been multiplied by 100.*

	Mutation	1	3	5	7	Hyp.
Binary	Sheep	16.85 (*)	3.15 (*)	1.36 (*)	**0.13** (*)	0.76 (*)
	Phobic	16.78 (*)	19.43 (3.64)	68.44 (9.01)	105.68 (1.21)	2.49 (*)
	Lone Rider	7.64 (*)	48.46 (8.86)	97.73 (6.10)	113.16 (2.37)	5.78 (*)
Gray	Sheep	1.97 (*)	2.97 (*)	**0.0018** (*)	0.85 (*)	1.23 (*)
	Phobic	1.23 (*)	1.78 (*)	2.86 (*)	8.44 (*)	0.99 (*)
	Lone Rider	**0.002** (*)	1.20 (*)	3.48 (*)	10.57 (*)	0.99 (*)

Table 5: *Mimetic results for Griewank function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations, except for cases (*) where the averages and standard deviations are non-significant; the figure is then the* **median** *of the 21 runs. All figures have been multiplied by 100.*

As already mentioned in the general overview, these problems are comparatively easy. On both problems, the Sheep gets the best results. On Griewank-binary, the Sheep is slightly behind the Ignorant and optimal results are obtained for high values of m (optimum at $m = 7$ for the Sheep on Griewank-binary). On both problems, too, the Lone Rider is one order of magnitude better than the Phobic - though slightly worse that the Sheep in the binary case.



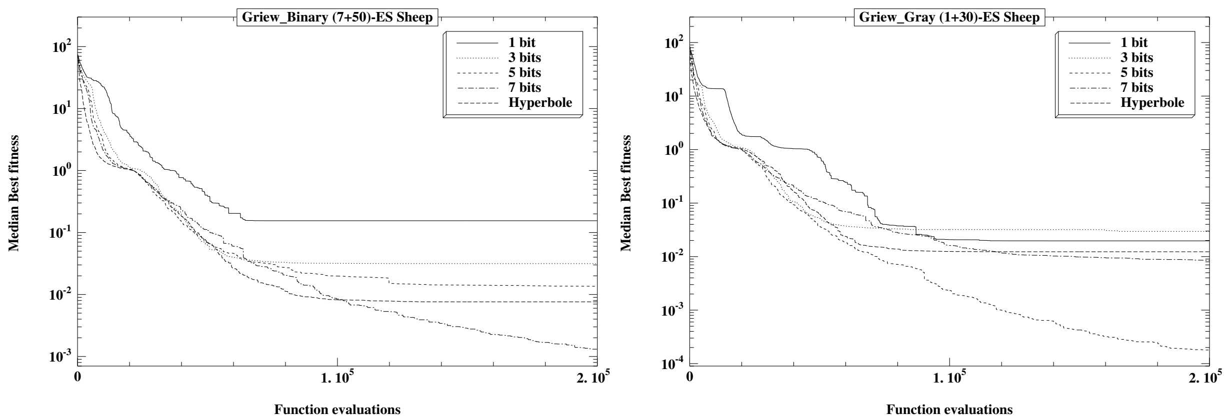


(a) Binary coding (7+50) (b) Gray coding (1+30)

Figure 8 *ES/Ignorant on Griewank function (logarithmic scale)*

What is surprising is that the best results are obtained with a high m for the Sheep and for $m = 1$ for the Lone Rider. The fact that the Sheep can afford much larger mutation steps than the Ignorant can be explained from the memory: intuitively, more (good) information allows to find short-cuts in the fitness landscape. On the other hand, the mandatory small steps of the Lone Rider might be explained from the balance between exploration and exploitation: Increasing the value of m favors exploration over exploitation. Moreover, using the Lone Rider instead of the Sheep similarly favors exploration (fleeing away instead of imitating the previous optima) over exploitation. Using the Lone Rider together with a large mutation step might result in too strongly bias toward exploration.

For all mimetic strategies, the performances are more sensitive to the population size μ and natality λ than for the Ignorant strategy.



(a) Binary coding, Sheep, (7+50) (b) Gray coding, Sheep, (1+30)

Figure 9 *Mimetic Evolution on Griewank function*

As in Figures 6 and 7 (function F_2), the hyperbolic curve shows higher slope that the fixed schemes in the first half of the plots of Figure 9. Afterward, it behaves as a 1-bit mimetic evolution (but the logarithmic scale makes the hovering less visible). In Figure 8, where the 1-bit mimetic scheme is the best one, the hyperbolic scheme simply catches up that best behavior a little after 100000 evaluations.

Another striking fact in Figure 9 is the outstanding best value for the mutation strength (7 for Binary, 5 for Gray). However, complementary experiments show that no further improvement is brought by increasing the value of m.

5.5.3. Rosenbrook Function

Again, the results on Rosenbrook problems are presented in Table 6 and 7 below for the offline results, and in Figure 10 and 11 for some sample median online plots.

	PBIL			ES	IGNOR			
	.1	.2	.5	(Hyp)	1	3	5	7
Bin.	816 (*)	963 (*)	905 (*)	552 (*)	1752 (*)	638 (*)	**437** (*)	**427** (*)
Gray	1146 (289)	259 (78)	75 (33)	154 (50)	**5.25** (*)	115 (67)	258 (53)	409 (60)

Table 6: *Reference results for Rosenbrook function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations, except (*) where the very different orders of magnitude of the fitnesses makes again averages and standard deviations non-significant: the given figure is then the* **median** *of the 21 runs. Also note the range of parameter α for PBIL differs from all other similar tables.*

On this problem, the best reference algorithm is the Ignorant. However, the different codings seem to shape very different fitness landscapes.

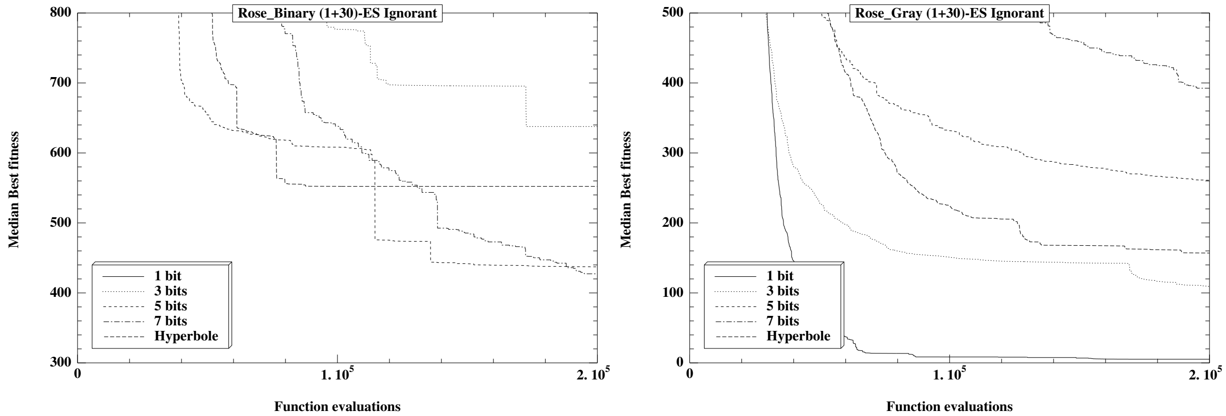
For Rosenbrook-binary, the best performance is obtained for rather high values of m ($m = 7$) and optimization ends very far from the actual optimum (427 vs 0).

On the opposite, for Rosenbrook-Gray, the best performance is obtained for $m = 1$ (i.e. with a simple Hill-Climber), and optimization reaches 5.25. Also the best results of PBIL were obtained for $\alpha = 0.5$, i.e. with rapidly changing distribution.

This suggests that indeed the binary coding creates more local optima and difficult barriers (e.g. the so-called Hamming cliffs) than Gray coding [52], requiring larger steps to overcome these difficulties. The characteristic stair-like shapes of the plots of Figure 10 witnesses such sudden changes (whenever some cliff is over-passed).

	Mutation	1	3	5	7	Hyp.
Binary	Sheep	1368 (*)	1223 (*)	**325** (*)	418 (*)	380 (*)
	Phobic	700 (*)	628 (*)	2275 (487)	9280 (3017)	914 (*)
	Lone Rider	380 (*)	511 (*)	652 (*)	694 (*)	815 (*)
Gray	Sheep	**11.6** (6.8)	126 (57)	167 (79)	225 (77)	183 (59)
	Phobic	**12.8** (6.8)	265 (58)	308 (49)	416 (69)	279 (52)
	Lone Rider	**14.3** (6.7)	238 (57)	319 (59)	453 (63)	298 (65)

Table 7: *Mimetic results for Rosenbrook function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations, except (*) where the very different orders of magnitude of the fitnesses makes again averages and standard deviations non-significant: the given figure is then the* **median** *of the 21 runs. For instance, the Sheep strategy on the binary coding with one bit mutation has an average of 14000+ and a standard deviation of 27000+!*



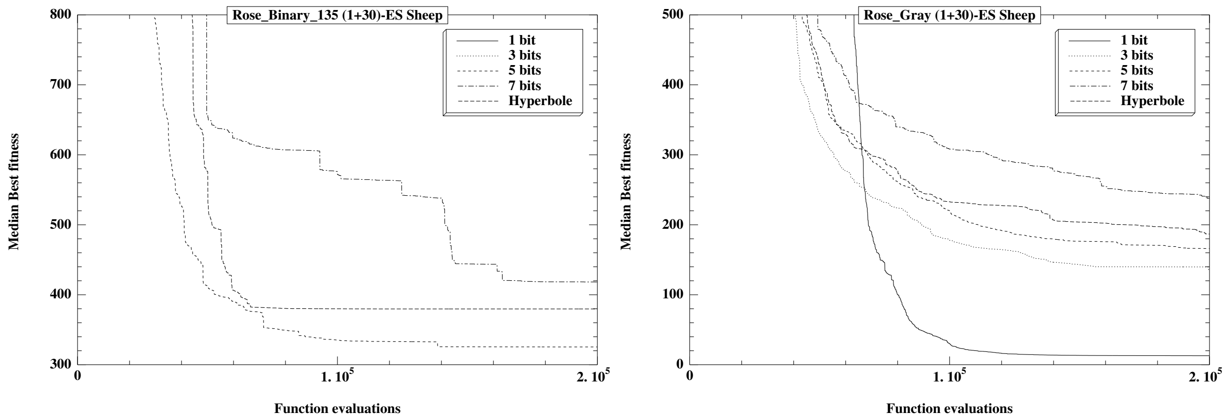
(a) Binary coding (1+30) (b) Gray coding (1+30)

Figure 10 *ES/Ignorant on Rosenbrook function*

As for Griewank problems, the Sheep appears the more suited mimetic strategy: it behaves well with both encodings. Moreover, the situation for the binary problem is quite similar to that of the Griewank binary problem: the Sheep outperforms all other strategies (including the Ignorant) when using 5-bits mutation, while the Lone Rider performs a little worse, but with 1-bit mutation only.

However, the Rosenbrook Gray problem shows a picture quite different from the Griewank Gray: All strategies get their best results with a 1-bit mutation, and are slightly outperformed by the Ignorant. It seems that no useful information can be obtained from the past evolution, no short can be found. Even worse, the memory might give false indications, resulting in worse results than the Ignorant strategy.

On Figure 11-a, the stair-like shape of the 7-bit curve, and the very poor performance of the 1- and 3-bits curves (not visible!) again suggests the existence of barriers in the fitness landscape. In the same line, the Hyperbole switches from the behavior resembling the 7-bit curve to values close to those of the 5-bit curve before getting stationary as the mutation strength reaches smaller values.



(a) Binary coding, Sheep, (1+30) (b) Gray coding, Sheep, (1+30)

Figure 11 *Mimetic Evolution on Rosenbrook function*

On Figure 11-b, it can be seen that the 1-bit mutation needs some time before finding a quick way toward good values. On the other hand, the Hyperbole never finds such way down, probably trapped by its first steps in some completely different region of the search space.

5.5.4. Rastrigin Function

For Rastrigin problems, only the off-line results are presented in Table 8 and 9. Indeed, the on-line behavior of all algorithms does not provide much useful information: almost the same comments than for function F_2 can be made.

	PBIL			ES	IGNOR			
	.05	.1	.2	(Hyp)	1	3	5	7
Bin.	202 (14)	158 (20)	158 (15)	158 (17)	254 (23)	156 (23)	**152 (18)**	177 (21)
Gray	191 (16)	172 (14)	162 (15)	162 (14)	211 (14)	144 (11)	**138 (14)**	140 (16)

Table 8: *Reference results for Rastrigin function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations.*

The unique characteristic of these results is that all reference schemes behave almost the same whatever the coding of the problem. Further, there is not much difference between ES, the Ignorant and PBIL.

	Mutation	1	3	5	7	Hyp.
Binary	Sheep	248 (21)	195 (17)	170 (17)	197 (18)	195 (19)
	Phobic	231 (28)	165 (19)	**148** (17)	**150** (19)	156 (16)
	Lone Rider	230 (16)	173 (22)	**154** (12)	**154** (14)	**151** (14)
Gray	Sheep	217 (17)	205 (18)	200 (22)	188 (17)	210 (17)
	Phobic	242 (14)	224 (22)	194 (17)	**170** (19)	179 (18)
	Lone Rider	235 (15)	219 (13)	196 (13)	**171** (15)	185 (16)

Table 9: *Mimetic results for the Rastrigin function; Average offline results (standard deviation) over 21 runs of 200,000 evaluations.*

The best performances are obtained for the Phobic and the Lone Rider, both with rather high values of m ($m = 5$ or 7 for Rastrigin-binary, and $m = 7$ for Rastrigin-Gray. Nevertheless, the mimetic strategies give results similar to the Ignorant strategy for the binary case, and are slightly outperformed on the Gray problem – though no algorithm gets any close to the global optimum.

5.6. The niche of Mimetic Evolution

As a summary of the results of the previous section, consider Table 10 below, ranking all strategies on each of the test problems.

Function	1st	2nd	3rd	4th
F_2-b	Lone Rider (3,5)	Phobic (3)	Ignorant(3)	Sheep (7)
F_2-G	Lone Rider (3)	Phobic(3)	Sheep (3)	Ignorant (3)
Griew-b	Ignorant (1)	Sheep (7)	Lone Rider(1)	Phobic (1)
Griew-G	LoneRider(1), Sheep(5)	Phobic(1), Ignorant(1)		
Rosenb.-b	Sheep (5)	Lone Rider (1)	Ignorant (7)	Phobic (1)
Rosenb.-G	Ignorant (1)	Sheep (1)	Phobic (1)	Lone Rider (1)
Rastr-b	Phobic (5)	Ignorant (5)	Lone Rider (5)	Sheep (5)
Rastr-G	Ignorant (5)	Lone Rider, Phobic (5)	Sheep (7)	

Table 10: *Rank of the different mimetic strategies over all problems.*

On Griewank-binary and Rosenbrook-Gray, the Ignorant with $m = 1$, i.e. an algorithm resembling a simple Hill-Climber, is the best option and falls close to the optimum. Clearly, mimetic evolution brings no definite advantage as a memory-less evolution can do the job. These problems should be excluded from the scope of mimetic evolution, as too "easy". It is most interesting, incidentally, that mutating a fixed number of bits in any individual appears more efficient than using a probability of mutation p_i per bit. This can be explained as, when probability p_i is low (around $1/N$), mutation happens to be very frequently inactive[8].

[8] Typically, for $N = 100$, the probability of mutating no bit is $(1 - 1/N)^N$, that is, circa 36%.

On Rastrigin-Gray, the Ignorant with $m = 5$ shows the best option, though it falls far from the optimum. Such functions can be considered symmetrically as too difficult for Mimetic evolution: either mimetic evolution fails to construct a relevant memory, or it does not use the models in a proper way. Indeed, the use of models offers room for improvement, and some perspectives of further work on this point will be discussed in the next section.

On the other problems, mimetic evolution brings some improvement over the reference algorithms, with two different settings: the Sheep with rather high values of m (e.g. $m = 5$ for Griewank-Gray and Rosenbrook-binary), and the Lone Rider or the Phobic with moderate values of m ($m = 3$ for F_2-binary and F_2-Gray, $m = 5$ for Rastrigin-binary).

In the latter case, the fact that the Lone Rider and the Phobic strategies are the best ones gives some hints into the structure of the fitness landscape: either recombining the individuals is not relevant (e.g. macro-mutation would be more appropriate than crossover [27]); or maintaining the diversity of the population is more important than recombination.

In the former case, the fact that the Sheep strategy outperforms the other ones symmetrically implies that a fast recombination-diversification of the individuals is relevant. This could be confirmed by the fact that large gaps can be viewed in the performances (Figure 3): the landscape is composed of local optima with large basins of attraction. The population climbs toward the local optimum, then waits for finding the good direction toward another basin of attraction.

Most surprisingly, mutating a constant number of bits shows sufficient to reach good performances in many cases, which implies that bounded-mutation is sufficient to escape many local optima. This contradicts the intuition that mutation must be able to make very large steps, even with low probability, in order to prevent evolution from premature convergence [9] But the fact that many basins of attraction can be escaped with a small jump (0.5% of the total number of bits) might come from the high dimension of the problems considered. Indeed, the size of the neighborhood grows exponentially with the dimension of the space, which modifies the rarity of local optima [51].

6. Conclusion and Perspectives

This paper is concerned with the possible forms and uses of memory in artificial evolution, and focuses on the role of explicit common memory. In the line of PBIL [5] and Evolution by Inhibition [48], we investigate through the mimetic evolution scheme, how to use the memory of the past best and worst individuals generated in the previous generations.

These memories can be interpreted in various ways. They can first be considered as distributions on the search space. A restricted distribution space was considered here, i.e. the variables are assumed locally independent. But richer distribution space can be considered: This mech-

[9] For instance, [55] recommends the use of a Cauchy distribution instead of a Gaussian law to evolve the auto-adaptive mutation rate in continuous Evolution Strategies. Making high steps more probable than with a Gaussian distribution is observed to speed up evolution or prevent it from premature convergence on many benchmarks functions.

anism was recently extended to Genetic Programming [42], continuous optimization [46], and combinatorial optimization [6].

The memories can also be viewed as a self-adaptive system of coordinates in the search space, varying along evolution. The goal is to define the relevant direction (i.e. some gradient information) with respect to this system of coordinates, or mimetic strategy. Only fixed strategies were investigated in this paper (section 3.3). An open issue is to automatically adjust the relevant strategy along evolution, either at the population level, or at the individual level. Still, on the restricted set of functions studied in this paper, only three strategies are worth considering. Further, they appear useful in different situations: when diversity is important for the Phobic and Lone Rider strategies; when evolution has to jump many times from an optimum to another for the Sheep strategy – coupled with large mutation steps in that case.

Indeed, dealing with a direction rather than with a distribution raises the additional question of how to set the strength of the mutation (the size of the mutation steps m). Only fixed and hyperbolic (decreasing m from N/2 to 1) settings have been considered so far. However, some hints are given by the experimental results: First, mutating a fixed number of bits m per individual should be preferred to mutating all bits with a fixed probability per bit – provided the optimum value for m is found, as in many case there is a clear optimum value (Figures 6 to 9); Second, a decreasing schedule might prove beneficial – though the hyperbolic scheme proposed in [3], decreasing from $N/2$ to 1 should be fine-tuned: if 1-bit mutation is not the optimum value, mutating one bit during the last half of the run is useless. Ongoing research is concerned with investigating other simple schedules for adjusting m, using either an hyperbolic decrease toward some $m_{end} > 1$, or an online adaptation mechanism.

Of course the issue of determining the optimal values for the mutation strength remains open, and at the moment still relies on extensive numerical experiments for a large number of settings of mutation schedules and mimetic strategies. But on the other hand, such experiments might help to understand the very nature of the fitness landscape at hand, and an unexpected achievement might be a new background to assess problem difficulties – independently of how well mimetic evolution

A last perspective of research is concerned with extending mimetic evolution to multi-objective optimization. The objective would be to construct models sampling the Pareto front. In that context, the control of evolution shifts toward how to determine the natality of each model, and how to update the models from the current population (e.g., should a non-dominated individual be used to update any model? how to determine match individuals and models . . .).

More generally, when considering an explicit memory, evolution shifts from the phenotype-genotype paradigm [30] to the paradigm of *distribution of* phenotypes/*distribution of* genotypes. This new search space is always larger, thus such approach should indeed be more powerful. But only if the following open problems can be solved: find adequate evolution operators in a distribution space; characterize the benefit of distribution-based evolution, i.e. the class of functions relevant to a memory-based approach.

References

[1] T. Bäck. *Evolutionary Algorithms in theory and practice.* New-York:Oxford University Press, 1995.

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

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

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

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

[6] Shumeet Baluja and Scott Davies. Fast probabilistic modeling for combinatorial optimization. In *American Association for Artificial Intelligence*, 1998.

[7] B. Buchanan and E. Feigenbaum. DENDRAL and META-DENDRAL. *Artificial Intelligence*, 11:5–24, 1978.

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

[9] K. A. DeJong. Are genetic algorithms function optimizers ? In R. Manner and B. Manderick, editors, *Proceedings of the 2^{nd} Conference on Parallel Problems Solving from Nature*, pages 3–13. North Holland, 1992.

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

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

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

[13] D. B. Fogel, L. J. Fogel, W. Atmar, and G. B. Fogel. Hierarchic methods of evolutionary programming. In D. B. Fogel and W. Atmar, editors, *Proceedings of the 1^{st} Annual Conference on Evolutionary Programming*, pages 175–182, La Jolla, CA, 1992. Evolutionary Programming Society.

[14] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the 4^{th} Conference on Parallel Problems Solving from Nature*, number 1141 in LNCS, pages 584–593. Springer Verlag, Sept. 1996.

[15] J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. Technical Report 389, CMAP, Ecole Polytechnique, 1998.

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

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

[18] D. E. Goldberg. Zen and the art of genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, pages 80–85, 1989.

[19] D. G. Goldberg and R. E. Smith. Nonstationary function optimization using genetic dominance and diploidy. In J. J. Grefenstette, editor, *Proceedings of the 2^{nd} International Conference on Genetic Algorithms*, pages 59–68. Lawrence Erlbaum Associates, Inc., 1987.

[20] J. J. Grefenstette. Virtual genetic algorithms: First results. Technical Report AIC-95-013, Navy Center for Applied Research in Artificial Intelligence, February 1995.

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

[22] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the Fourth IEEE International Conference on Evolutionary Computation*, pages 65–69. IEEE Press, 1997.

[23] G.E. Hinton and S.J. Nowlan. How learning can guide evolution. *Complex Systems*, 1(495-502), 1987.

[24] J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[25] J. Horn, D.E. Goldberg, and K. Deb. Long path problems. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the 3^{rd} Conference on Parallel Problems Solving from Nature*, pages 149–158. Springer Verlag, 1994.

[26] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228, 1993.

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

[28] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution.* MIT Press, Massachussetts, 1992.

[29] J. R. Levenick. Inserting introns improves genetic algorithm success rate : Taking a cue from biology. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4^{th} International Conference on Genetic Algorithms*, pages 123–127, 1991.

[30] R. C. Lewontin. *The Genetic Basis of Evolutionary Change.* Columbia University Press, 1974.

[31] T.M. Mitchell. *Machine Learning.* MIT Press, 1995.

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

[33] A. Newell and H. Simon. *Human Problem Solving.* Prentice Hall, 1982.

[34] S. Nolfi and D. Floreano. How co-evolution can enhance the adaptive power of artificial evolution: implications for evolutionary robotics. In P. Husbands and J.A. Meyer, editors, *Proceedings of EvoRobot98*, pages 22–38. Springer Verlag, 1998.

[35] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–20, 1991.

[36] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann, 1995.

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

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

[39] R.G. Reynolds and J.L. Matelik. The use of Version Space controlled Genetic Algorithms to solve the boole problem. *Int. J. on Artificial Intelligence Tools*, 2(2):219–234, 1993.

[40] G. Rudolph. Convergence of non-elitist strategies. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 63–66. IEEE Press, 1994.

[41] S. Russell and A. Norwig. *Artificial Intelligence, a modern approachg.* Prentice Hall, 1995.

[42] R. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.

[43] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms*, pages 36–40. Morgan Kaufmann, 1987.

[44] M. Schoenauer. Shape representations and evolution schemes. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the* 5^{th} *Annual Conference on Evolutionary Programming*, pages 121,129. MIT Press, 1996.

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

[46] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In Th. Bäck, G. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, pages 418–427. Springer Verlag, 1998.

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

[48] M. Sebag, M. Schoenauer, and C. Ravisé. Toward civilized evolution: Developping inhibitions. In Th. Bäck, editor, *Proceedings of the 7^{th} International Conference on Genetic Algorithms*, pages 291–298. Morgan Kaufmann, 1997.

[49] W. M. Spears. Adapting crossover in a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4^{th} International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991.

[50] G. Syswerda. A study of reproduction in generational and steady state genetic algorithm. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[51] V. N. Vapnik. *The Nature of Statistical Learning*. Springer Verlag, 1995.

[52] D. Whitley, S. Rana, and R. Heckendorn. Representation issues in neighbourhood search and evolutionary algorithms. In D. Quadraglia, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Sciences*, pages 39–58. John Wiley, 1997.

[53] C. Wills. *The wisdom of genes*. Elsevier, 1991.

[54] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical report, Santa Fe Institute, 1995.

[55] X. Yao and Y. Liu. Fast evolution strategies. In B. Reynolds and M. Conrad, editors, *Proceedings of the 6^{th} Annual Conference on Evolutionary Programming*, LNCS 1213, pages 151–162. Springer Verlag, 1997.