International Journal on Artificial Intelligence Tools
Vol. 0, No. 0 (1993) 000—000

AN EVOLUTIONARY APPROACH FOR TIME DEPENDANT OPTIMIZATION

PHILIPPE COLLARD, CATHY ESCAZUT, ALESSIO GASPAR
University of Nice-Sophia Antipolis, Laboratory I3S — CNRS
250 Av. A. Einstein, Sophia Antipolis, 06560 Valbonne FRANCE
{pc,escazut,gaspar}@unice.fr

Received (received date)
Revised (revised date)

Many real-world problems involve measures of objectives that may be dynamically optimized. The application of evolutionary algorithms in time dependant optimization is currently receiving growing interest as potential applications are numerous ranging from mobile robotics to real time process command. Moreover, constant evaluation functions skew results relative to natural evolution so that it has become a promising gap to combine effectiveness and diversity in a genetic algorithm. This paper features both theoretical and empirical analysis of the behavior of genetic algorithms in such an environment. We present a comparison between the effectiveness of traditional genetic algorithm and the dual genetic algorithm which has revealed to be a particularly adaptive tool for optimizing a lot of diversified classes of functions. This comparison has been performed on a model of dynamical environments which characteristics are analyzed in order to establish the basis of a testbed for further experiments. We also discuss fundamental properties that explain the effectiveness of the dual paradigm to manage dynamical environments.

Keywords: Evolutionary computation, Genetic algorithms, Dual genetic algorithm, Dynamic environment, Diversity.

1. Introduction

Origins of evolutionary approaches are old since they date back to the founding fathers of cybernetics' old dream: they wanted to construct biological-like machines. Unfortunately, the technology available at this time did not allow the implementation of these ideas. We had to wait for almost 40 years to see the evolutionary concepts applied to artificial systems.

Nowadays, evolutionary computing is one of the emergent technologies. Being essentially a software activity, it has been successfully applied in various areas such as automatic programming, optimization, machine learning, immune systems .... This paper only focus on optimization tasks performed by genetic algorithms faced to time dependant problems.

In this article, we first outline the features of *genetic algorithms*. Then, the following section is an overview of *dynamical environments*. The next section is aimed at presenting how *diversity* can be maintained within GAs faced to a dynamical problem. Section 5 presents our new class of GAs: *dual genetic algorithms*. It details why and how we have introduced duality within the system. Then, we explain why diversity can be implicitly maintained in such systems. Section 7 experimentally investigates the dual approach, comparing it to the standard one. We finally discuss the theoretical and empirical results obtained, and then enlighten further works currently investigated.

2. Genetic Algorithms

Genetic algorithms (GAs) were introduced in the sixties by J. Holland.[1] These evolutionary optimization methods are inspired from natural selection: from a generation to the next one a set of new artificial structures is created using parts of the best individuals of the previous generation, and creating new ones. While the *exploration* in GAs is largely performed randomly, these algorithms are not a simple hazardous walk. Indeed mathematical foundations have proved the whys and therefores of GAs.

2.1. *Basic principles*

According to the neo-Darwinian evolution theory, GAs apply natural selection and genetic-like operators to a *population* of individuals coding in their(s) chromosome(s) a potential solution for the considered problem. Members of this population are generally represented by a binary string of length λ. They also are assigned a *fitness* value which quantifies how much it is able to solve the given problem. When optimizing with a GA, the reward is simply computed by applying a fitness function to the value encoded by the chromosome. If we are looking for a global optimum, the higher the value of a chromosome when decoded and applied the function is, the higher its fitness is and the lower the probability to be facing a local optimum is.

The highly fit individuals are given opportunities to "reproduce": they are first randomly *selected* following a scheme which favors the more fit members, and then recombined typically using the mechanisms of *crossover* and *mutation*. Thus, this produces new "offspring", which share features taken from each "parent". A new population is so created, containing a higher proportion of the characteristics possessed by the good members of the previous generation.

This way, the GA evolves, over consecutive generations, a population where best individuals spread thanks to the exchange during crossover of genetic material belonging to selected parents. This genetic exploration of the search space is repeated until a halting criterion such as convergence ratio or time constraint is met. Figure 1 details the steps of GAs.

The more basic forms of crossover and mutation are as follows:

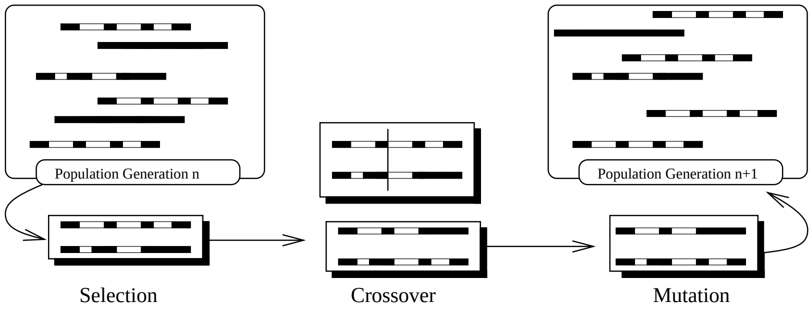

Figure 1: Standard Genetic Algorithm

- *Crossover* takes two individuals of the current population, and cuts their chromosome strings at some randomly chosen positions. Swapping the fragments between the two parents produces two offsprings. This crossover is known as the *n-point crossover* where n is the number of crossover sites. An other well-known crossover is the *uniform* one: rather than taking contiguous segments from each parent to form the offspring, each gene is chosen from one or the other parent, with a 50% probability for each parent. Crossover is usually applied with some probability typically chosen between 0.6 and 1.0. Crossover allows a rapid exploration of the search space.

- *Mutation* is applied to each child after crossover. This operator alters each gene of the chromosome string with a small probability (typically 0.001). Mutation provides a small amount of random search, and helps ensure that all points in the search space can be reachable independantly of individuals present in current population.

With these two techniques, the successive populations will evolve so that the fitness of the best and the average individual in each generation increases toward the global optimum. As a result, the final population may have largely converged toward the best individuals found to time, and thus only contains instances of them.

Selective pressure and recombination of genetic material between individuals has lead, as expected by J. Holland, to a robust and adaptive evolutionary process.

Let us now see why such a simple process performs well.

2.2. *Mathematical foundations*

We saw that GAs manipulate individuals. However, most of the theoretical works focus on the implicit processing of *schemata*. Holland's *schema theorem* was the first rigorous explanation of how GAs work.[1] Schemata are implicit similarity templates: they identify subsets of individuals sharing certain characteristics. They are usually defined over the ternary alphabet $\{0, 1, \star\}$ where '$\star$' is a "don't care" or a wild card match symbol. For instance, all the individuals of a problem coded with just 3 bits and having the last bit set to '1' can be represented by the schema $[\star\star 1]$.

From a geometrical point of view, we can notice that each schema corresponds to a *hyperplane* in the search space. Let us consider again the simple 3-dimensional space, the whole search space can be represented as a cube with the string [000] at the origin. The corners in this cube are numbered by bit strings in such a way that all adjacent corners are labeled by bit strings that differ by exactly 1 bit (see figure 2). The front plane of the cube represents the schema [0⋆⋆], the right one represents the schema [⋆⋆1], the upper one stands for [⋆1⋆] …

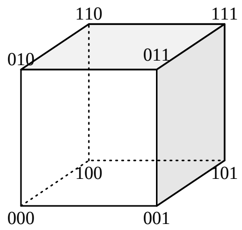

Figure 2: Geometrical view of a 3-dimensional search space.

The *order* of a schema is the number of specified genes (i.e. non-⋆ symbols), and its *defining length* is the distance between the first and the last bit in the schema with value either 0 or 1. For instance, the schema [0⋆⋆1⋆] is order-2 and its defining length is equal to 3. Two schemata are said to be *compatible* if there exists at least one individual being an instance of both. For instance, schemata [0⋆⋆1⋆] and [⋆⋆01⋆] are compatible while [0⋆⋆1⋆] and [1⋆01⋆] are not.

Based on how schemata are processed, the *schema theorem* proves that certain good schemata, called *building blocks*, get an exponentially increasing number of representatives in the population. Building blocks are high-performance, short-defining-length and low-order schemata.[2]

Let us now have a particular interest on the schema concept, bringing to the fore the properties that favor the emergence and the manipulation of the building blocks.

2.3. *Schemata and their properties*

Radcliffe defines four properties he thinks necessary to a useful representation in GAs.[3] They are:

1. *the closure:* the intersection of any pair of compatible schemata should itself be a schema.

2. *the respect:* crossing two instances of any schema should produce another instance of this same schema.

3. *the proper assortment:* given instances of two compatible schemata, it should be possible to cross them in order to produce a child which would be an instance of both schemata.

4. *the ergodicity:* it should be possible, through a finite sequence of applications of genetic operators, to access any point in the search space given any population.

Unfortunately, demonstrating these four properties is not enough to represent all the hyperplanes of the space. Indeed, one can notice that each schema is represented by one and only one hyperplane, but the converse is not true. For instance, the diagonal hyperplane {[000], [001], [110], [111]} grouping the four individuals having the two first genes identical, can not be expressed by a single schema.). Now, let us see how the dynamics of a GA can be simulated.

2.4. *An executable model*

Numerous studies in the field of GAs try to observe and explain their behavior thanks to experiments on a given set of function optimization problems.[4,2] Then a statistic analysis enables to draw conclusions and, for instance, to identify the optimal parameter setting. However, these experiments take a long time and are hazardous.

A solution consists in simulating the working of a GA, studying the evolution of each individual without the non-deterministic choices which are usually used. Simulations enable a quick test for the relevance of new approaches. We propose to simulate the behavior of GAs using the executable model proposed by Whitley.[5] It uses an infinite population, no mutation and requires the enumeration of all points in the search space. Assuming proportionate reproduction, one point crossover, and random mating of the reproduction products, a simple non linear dynamic system can be used for modeling the evolution in time, of the expected proportion of each individual. Thus, the following quadratic equation computes the proportion of individual k in the next generation:

$$\begin{aligned} p_k^{t+1} = \quad & x_k \\ - \quad & \textstyle\sum_{s=1}^{\lambda} p_c \cdot \sum_{k/\chi_s^\lambda(k,l)\neq k} x_k \cdot x_l \\ + \quad & \textstyle\sum_{s=1}^{\lambda} p_c \cdot \sum_{i,j/\chi_s^\lambda(i,j)=k} x_i \cdot x_j \end{aligned}$$

where $x_i = p_i^t \cdot \frac{F_i}{\overline{F}}$, p_i^t being the proportion of the individual i at generation t, F_i its fitness, and $\overline{F}$ being the average fitness of the current population. $\chi_s^\lambda(i,j)$ is the crossover on site s applied to the λ-bit strings i and j. The first term of this equation represents the reproduction operator. The two other ones deal with the crossover effects: the destruction of the individual k combining k and l, its birth after a crossover manipulation between i and j. Let us note that it is possible to develop exact equations for problems up to approximately 15 bits in the coding. This formula generalizes the equations given by Goldberg.[6] This model may be compared to the Vose and Liepins' formalism and their interpretation using matrices.[7,8]

The following section presents *dynamical environments* and how they can be optimized using GAs.

3. Dynamical Environments

This section introduces *dynamical environments* as fitness landscapes featuring fitness variations over time. At first we establish the link between *time dependant optimization* and *static optimization* using the concept of *fitness dynamism*.

3.1. *Fitness dynamism*

Fitness dynamism occurs in a GA's run when the fitness landscape being optimized is non-stationary. That is, *structures* as individuals or schemata have a fitness changing over time. This rather simple definition allows ourselves to immediately distinguish between *exogenous fitness dynamism* and *endogenous fitness dynamism*. While the former concerns fitness variations induced by the environment itself (i.e. the fitness function), the latter is related to fitness changes only due to the own dynamics of the GA.

To illustrate these categories we can take first, as an example of *endogenous dynamism*, the *hyperplane sampling* process which is central to GAs exploration's capabilities. Hyperplanes (represented by schemata) are sampled according to the average fitness of their representatives in the current population. When working with finite populations (i.e. not using a simulation), this average evaluation clearly induces a bias as soon as all individuals subsumed by a particular schema are not equally represented in population. Moreover, as successive populations change according to genetic transformations, this average value may also change at each generation. Thus, a non relevant search space area may appear promising during the very first generations of the genetic exploration process. Endogenous fitness dynamism also occurs when applying *sharing scaling*, or any fitness adjustment techniques computing one individual's fitness on the basis of other ones values.[9,?].

On the other hand, *exogenous dynamism* concerns individual fitness changes due to environmental causes. Dynamical environments appear as a particular case of fitness dynamism where alterations are not a direct consequence of dynamics of the GA. Applications modeled by a dynamical environment are numerous and include for example mobile robotics as well as chemic processes control or even optimization of any fitness function incorporating a time parameter or fluctuating variables.

What is then the common issue of both stationary and dynamic optimization tasks ? When using a GA, the population tends to get homogenized. After convergence it eventually contains instances of the best chromosome found so far. If the environment changes at this moment, crossover will not be able to introduce any diversity and the GA will have to rely only on mutations to restart a new exploration. When confronted to static optimization, a lack of diversity in initial population may also mislead the overall genetic search in case of deceptive functions. Our approach to address dynamical optimization issues rely on the enhancement of genetic algorithm's adaptiveness. This goal is achieved by an improvement of the exploration/exploitation tradeoff, thus leading to a qualitatively best diversity management. Another way to achieve such a goal would be to introduce a *cognitive* rather than a *reactive* method to handle dynamical environments. This one should be aimed at modeling *phylum memory* and adaptive systems foretelling abilities that sound very close to the abductive problematic for *artificial life* (ALife).

This paper will focus on the endogenous approach. Let us now formally describe major features of dynamical environments.

3.2. *Characterizing dynamical environments*

We previously defined non-stationary fitness functions as based on the fitness landscape's alteration over time. In such a context, we call *transition* the altering step. Following parameters are thus fundamental to define dynamical environments:

- *Period duration (i.e Transition Speed) (δ_s)*: It is the number of generations during which fitness landscape remains unchanged.
- *Transition distance (δ_d)*: A transition changing low fitted individuals fitness is careless in an optimization context as the GA has already converged toward an homogeneous population containing good chromosomes not concerned by such changes. The average fitness will be kept quasi-unchanged. In opposition, when changing near optimal individual fitnesses, the actual population gets destabilized and the system will need some generations to get adapted again to the new environment. The transition distance measures, in unimodal environments, hamming distance between the genotype of the "old" optimum and the "new" one. In multimodal environments, this criterion can easily be adapted by averaging distances to new optima from previous ones.
- *Average shape alteration ($\overline{\delta}$)*: Rather than focusing on optima, this more general criterion takes into account the fitness variations in the overall fitness landscape. Let us note $\overline{\delta} = \sum_{\omega \in \Omega} F_{\omega}^{t-1} - F_{\omega}^{t+1}$ this alteration quantification criterion also measuring a kind of instability of the landscape. It may be low for a landscape only featuring *moving optima* and high for another one changing all fitnesses except optima's ones. It is thus not correlated to the static difficulty of optimization but rather to what should be called *dynamical activity*.

All these criteria rely on temporal aspects of the dynamism, i.e. when and how the fitness landscape is altered. It is also worth considering, among structures implicitly (schemata) or explicitly (individuals) handled, which ones are about to be altered by transitions. We thus define the *order of dynamism* of a given dynamical environment, noted $\mathcal{O}_{dy}$, as being the highest order of non stable schemata. By non-stable schemata we mean any schema which average fitness value based on all its representatives in the search space is not constant over time. If at least one schema is non-stable at order i we can say that any schema of higher order will be so too. This is straightforward to see that, being dynamic at order i means that the sum of some individuals fitnesses in an hyperplane is non constant over time. As the concerned hyperplane is subsuming hyperplanes of lower order, these ones will neither be stable and so on until we reach individuals responsible for this fitness instability.

These criterion form a set of tools that can be used according to the kind of landscape being optimized to quantify its dynamical difficulty. Part of them will be directly used in the remaining of this paper to construct models of dynamical environments easily analyzable.

3.3. *Models of dynamical environments*

We present, hereafter, the models of dynamical environments that will be investigated in the following.

3.3.1. *Minimal dynamic problem*

We define the *minimal dynamic problem* over $\lambda = 2$ gene long chromosomes. The plotting 3 shows the function used in our simulations; each of the 2^λ possible chromosomes is rewarded by a fitness that changes over time and ranges from 0 to 1 for the optimum. Each individual will be consecutively the optimum over periods due to the *translation* of the Gaussian-like fitness function over the search space Ω over time. By using a little search space (4 individuals), we get a function with a good fitness differentiation between chromosomes. When a transition occurs, the fitness of the previous optimum decreases of a relevant amount, encouraging the GA to rapidly find its successor. We will see that when working with longer chromosomes, which involves finer covering of such a function, we can get into trouble when transitions only lightly affect the previous optimum's fitness.

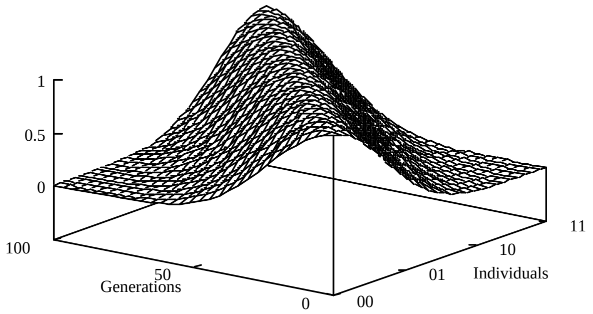

Figure 3: Simple dynamical Gaussian.

As we didn't used a gray code to enumerate successive optima, hamming distance between them will not be constant. This leads to diversified and non trivial transition distance values ranging from 1 for simple transitions as $[00] \rightarrow [01]$ and $[10] \rightarrow [11]$, to 2 (i.e. λ) for the middle one ($[01] \rightarrow [10]$).

This model is a simple one, but it introduces all relevant characteristics featured by dynamical environments and offers an opportunity to study them in details.

3.3.2. *Dynamical multi-dimensional Gaussians*

This second model is an extension of the previous one, allowing ourselves to experiment on relevantly sized search spaces. It also relies on a simple unimodal non-deceptive Gaussian-like fitness function sliding over Ω in order to center its top on each consecutive individual of the search space.

We generalize this kind of function over n dimensions by :

$$\frac{1}{n}\sum_{i=1}^{n} e^{-\pi*\left(\frac{x_i}{2^{(\lambda-1)}}-t\right)^2}$$

In this formula, t stands for the number of generations (time parameter), λ represents the chromosome's length and π a *peakness coefficient* parameterizing the Gaussian top's shape. This parameter has been introduced to avoid a "soft" top that induces small fitness differences between individuals belonging to the optimum's neighboring.

With such a function, the GA will consider the same individual as the optimum during consecutive transitions without suffering from a relevant drawback as the fitness difference between the real and the misleading optimum is not significant. In this case, our analysis criterion consisting in plotting the fitness of the best individual versus time does not reflect the fact that the GA hanged to the new optimum. Thus, we decided to use a Gaussian featuring a sharper peak and so we used a relatively high peakness coefficient.

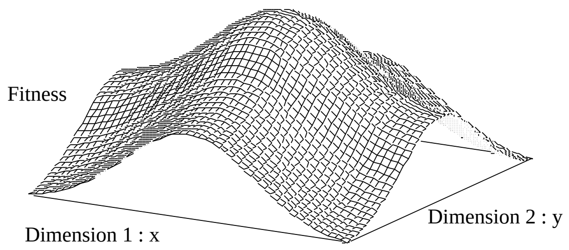

Figure 4: General shape of a dynamical Gaussian.

Figure 4 illustrates our function at date t in two dimensions. We chose to experiment with several dimensions each coded by a 8 genes long segments in the chromosome. This choice has been dictated by an other problem: as working with binary coded integers, the GA may, in case of a too high transition speed, only optimize the most significant genes according to the decoding algorithm. For example when handling 20 genes long chromosomes, the optimization of the first 5 bits induces a fitness above 95 % of the maximal one. This leads to final populations only consisting in partially adapted individuals. To avoid such a behavior, we decided to use chromosomes long enough to implement a relevant search space but coding for different variables. The transition will consist in a translation of the Gaussian's top over 2^8 individuals each composed of 8 identical sub-segments. For two dimensions, as shown in figure 4, this corresponds to simply following of Ω plan's diagonal. As expected, this motion will introduce variable hamming distances between two consecutive optima.

3.3.3. *Pattern tracking*

Our third model of dynamical environment is based on the control of the transition distance value. The fitness, F_x, of an individual x is thus the hamming distance between x and an arbitrary chosen optimum. At each transition, optimum is replaced by a random one located at distance δ_d from it. We will investigate *progressive pattern tracking* where the δ_d parameter undergoes an "acceleration" by being increased every δ_s generations. This environment has been suggested by previous work of Pettit and Swigger[10] and then Vavak and Fogarty[11].

3.3.4. *Modeling dynamical environments*

Let us now consider the three previously described models comparing them to existing works on the subject. We can distinguish several classes of dynamical environments:

- *Alternative fitness landscapes*: They are fitness landscapes that purely and simply change from one period to the following thus implying a high *average shape alteration*. Goldberg's work on diploidy was based on two alternative knapsack problems.[12] John Koza also investigated *Genetic Programming* in dynamical environments by directly changing the function to find every periods.[13]

  Such environments need a very strong adaptiveness as there is apparently no logic in transitions. It is hard to quantify their difficulty as they are characterized by the third criterion we stated, which is the less related to any difficulty evaluation.

- *Dynamical encoding*: Rather than altering the fitness landscape over time, one can directly change interpretation of alleles for particular loci. After investigation alternative fitness landscape for diploidy, Goldberg has also used such a dynamical environment.[12] Still using a knapsack problem, the allele of a particular locus was taken as itself during odd periods and as it complement during even periods.

  Such dynamical environments are quite similar to the ones applying geometrical transformations to the search space such as symmetry, slidings, ... That is why we think that these environments may be very relevant for the new class of GAs we propose and expound in section 5.

  It is worth noticing that pattern tracking environments can be considered as examples of dynamical encoding too. Let us say we want to track individual [000...00], changing the tracked individual over time is exactly similar to alleles interpretation alteration as in both cases, the results is that we reward presence of '1' where previously rewarding '0'. This allows ourselves to generalize this category by the following:

- *Morphic Fitness Landscapes*: Eventually, we consider in this third class fitness landscapes altered according to some *topological rules*. For example our model of sliding multi-dimensional Gaussian belongs to this category. The main feature is that from one period to the following, fitness landscape has been applied a regular geometric transformation such as sliding over the search space fitness values. Grefenstette proposed such a landscape where 3D-Gaussians were continuously altered over the fitness landscape.[14,15]

  We may expect our new GA to be able to take advantage of the regularities featured by this category of fitness landscapes. Comparing to the previous one, the tremendous difference rely on the method used to introduce geometrical transformations while both introduce regularity in function's dynamical aspects.

Now that the tremendous relevance of a diversity management has been enlightened, next section will focus on how to achieve this goal. We will consider entry points of the Genetic Algorithm that may enhance diversity, thus providing an overview of traditional and currently investigated approaches to this issue.

4. Diversity in Genetic Algorithms

The previous section presented the essential of dynamical environments, we now focus on how GAs can enhance the diversity management using improvements based on a few entry points of the evolutionist approach.

4.1. *From genotype to phenotype*

Within a GA, the *genotype* is the binary chromosome and the *phenotype* its decimal decoded value on which fitness function will be applied. To extend this concept of phenotype, we need a *Transcription* step which maps the chromosome (genotype) into a *phenotype* representing the results of a genes expression mechanism. This phenotype, which is a λ genes long chromosome, is then decoded in a decimal value to which is eventually applied the fitness function. Modeling two different levels for artificial evolution is one of the research areas for the entire ALife community which allows diversity mechanisms to be applied at the genotypical level without disturbing phenotypical convergence as will be detailed later. [16,17,18,19]

In response to the tremendous relevance of the choice of a good encoding scheme, an endogenous control relying on *meta-genes* appeared as the best way to introduce a meta level in the artificial evolution while enhancing its biomimetic aspects. In biology, such genetic material are called *introns* in opposition to *exons* (i.e. genes transcripted directly).[17,20,?] This classification has straightforwardly inspired to the evolutionist community three classes of introns:

- *Neutral introns*: Levenick has early underlined the relevance of inserting non-coding genes in a chromosome in order to decrease disruptiveness of relevant schemata while escaping encoding related issues like hitch-hiking.[18,19]

- *Pseudogenes* : Modeling pseudogenes is an approach to the recessive-dominant scheme already investigated by diploid approaches. The key idea is to have, in the chromosome, genes expressed and some non expressed together along with some kind of meta-genes specifying that pseudogenes will be expressed in the remaining of the chromosome and normal genes (exons) will no more be. For example, let us note with an overlined binary alphabet $\{\overline{0}, \overline{1}\}$ the pseudogenes and by the symbol $\diamond$ the gene switching expressiveness between exons and pseudogenes. The genotype $[0\overline{1}\diamond 1\overline{0}0]$ thus ignores pseudogenes before the switch gene and after it ignores exons. The expressed phenotype will thus be [00]. This is obviously a new encoding scheme for diploid-like approaches where diversity is ensured by the existence of non coding material thus not submitted to selective pressure and fitness function constraints. This material is potentially activated by any alteration on *switch genes*.

- *Meta-genes*: Meta-genes are used in our new model of GA (see section 5) as well as in a recent extension named *Folding Genetic Algorithms* (FGAs) and generalizing the use of meta-genes.[21] Basically, meta-genes concern all exons located beyond them until the next meta-gene. This exon segment is taken as it is in the phenotype if the meta-gene is inactive and taken as its complement otherwise. This mechanism that will be presented in more details later, leads ourselves to allow the GA to choose among different codings for a similar phenotype thus introducing diversity in populations.

4.2. *Fitness evaluation*

Operators such as sharing, fitness scaling or sigma truncation can, without neither changing mutations, crossover nor selection methods, drastically change the behavior of a GA by simply changing fitness of individuals.

A fitness evaluation is thus a good candidate as a method to improve diversity management in artificial evolution. We mainly distinguish two ways of evaluating the fitness of a given individual. The first one, and the most used, is an *absolute* evaluation as the individual is only evaluated according to the fitness function requirements on its transliterated genotype. The second main class of fitness functions concerns fitnesses no more computed only from an isolated individual but rather from a populational context. Such *relative fitness evaluations* are recognized as being closer to the behavior of natural systems than wide spread absolute fitnesses.[22] They correspond to the concept of *endogenous dynamism* previously enlightened. To be more precise, we distinguish three sub-categories of relative fitnesses on the basis of what interactions occur between individuals during the fitness evaluation process.

- *Competitive behaviors*: In competitive approaches, one individual can only gain *resources* at the depends of another one. Here the resource will be the fitness for *sharing* or *K-Tournaments* methods, places in the population for *crowding* or even reproduction facilities (χ_c rate) for *restricted mating*. In all cases, we determine if two individuals are sharing a particular resource according to some concept of *distance* (2^{nd} characteristic) which may be considered at several levels:

  - *Genotypic niching*: the distance is the hamming one between chromosomes.
  - *Selective niching*: the distance is here proportional to fitnesses difference absolute value.
  - *Topological niching*: this method needs to dispatch individuals on a toroidal grid and then to consider a neighboring relation to share a particular resource.

  The fundamental law of sharing is the following: "all individuals located at a certain range (*niche's radius*) from each other will share their resources making them decrease proportionally to the density of individuals located in the niche's radius".

- *Cooperative scheme*: In competitive scheme, each individual encodes a complete potential solution but is then evaluated according to its context. Individuals may *gather* and share good features before being evaluated. The main fact is that, although a unique individual may be evaluated, he can also cooperate with other ones in order to improve its own fitness or preserve diversity at a *macro-individual* level to survive eventual transitions. We are studying such a possibility based on individuals linkage in the population. Without entering into details in describing this approach which is out of the scope of this paper, individuals are now able to link to each others so that they may form *macro-individuals* coevolving along with normal chromosomes.

- *Symbiotic scheme*: In the previous model, cooperation is provided as an alternative to individual evaluation, *symbionts* are rather defined as entities needing to gather in order to form a complete solution. One can decide to split a solution encoding into several chromosomes and let them coevolve with the constraint to be evaluated only if coupled with a complementary chromosome encoding the other part of the solution. Such a work has been already investigated by Plotter in his PhD and leads to interesting results from both an efficiency point of view as well as from a model of symbiosis in artificial evolutionist systems.[?]

4.3. *Operational Diversity*

We consider this last class grouping as the most straightforward approaches to increase diversity in the search process of a GA. These *direct approaches* rely mainly on the mutation rate and work by introduction of a new genetic operator regulating it. According to the way this mutation rate control is coupled to environment, we can draw the following three classes:

- *Triggered*: The mutation rate is changed according to a condition reached by the population. thus, it needs a somewhat external monitoring as opposed to emergent of the population. The best illustration of such a technique is given by Grefenstette with triggered hypermutation which fixes the mutation rate to a particularly very high value when population fulfills a predicate implying its lack of adaptiveness to the new environment.[14,15]
- *Blind Adaptation*: Hypermutation is also proposed as a new operator taking at each generation a constant mount of individuals and replacing them by random brand new ones. An application of *simulated annealing* techniques to the mutation rate may also be efficient but arise a similar problem: what difference is there, from an ALife point of view, between a system not reacting to the environment and another always non-stable ? Both are not reacting; their dynamics are not coupled to the environmental one and range from absolute determinism to randomness, avoiding the particularly tremendous chaos behavior that should emerge from the adequation of the dynamics of the GA and the one of its environment.
- *Emergent Adaptation*: This is the most interesting case where control parameters of the genetic algorithm directly emerge from its own dynamics. This will be illustrated later in this paper by the so-called *implicit mutation rate* featured by the new class of genetic algorithms that we introduced (see §6.1).

Among these three classes only the third one conciliates efficiency with a real autonomy of the system. Indeed, both others need an external human expertise to tune or directly set dynamics parameters. Emergent Adaptation corresponds to a kind of meta approach where the system appear as a black box adapting by itself to its environment. Such work is more about to be classified in the context of Artificial Life researches while the former rather concerns commanded systems.

The next section, presents our new model of GA, that takes into account all the characteristics we previously enlightened as being useful for the optimization of time dependant problems.

5. Dual Genetic Algorithms

Dual genetic algorithms (DGAs) offer an improved robustness against most intrinsic weaknesses of standard GAs (SGAs) presented in section 2 .[23,24] They are based upon the introduction of a *transliteration* phase during the evaluation of a chromosome. This one may be compared with the RNA transcription in natural biology as it introduces a *gene expression* mechanism.

5.1. *Maintaining diversity with dualism*

Non-stationary environments present a problem for adaptive systems.[14] Indeed, when the SGA has converged to a current optimum, the population is almost fulfilled with individuals representing this optimum. Thus, the SGA might have some difficulties in reintroducing other individuals, and so diversity, in order to find out the next optimum. The only way to do is to use a mutation operator. But on the other hand this operator may have some disruptive effects on building blocks, and thus may lead to a non convergence of the SGA. This suggests that mutation has to be used at a very low rate.

Our approach introduces diversity in a natural way. The behavior of our GA tends to create redundancy in the search space. For all that, we take inspiration from a biological metaphor. Indeed, we can observe diversity in the human genetic code and also in the anti parallel double helical structure of DNA. This structure is composed of self-complementary sequences of nucleotides. Like the DNA molecule, the individuals of our GA are composed of two complementary strings. Thus a point in the search space has two totally different appearances. We call *doubles* the structures of such a pair. *Dualism* induces a redundancy in the search space since an individual has a single *phenotype* (i.e. behavior) but can be represented in two different *genotypes* (i.e. appearances).

5.2. *Implementation of dualism*

In order to keep the simplicity of genetic operators, we propose a quite simple implementation of the DGA: each chromosome is added a meta-gene in front of it. For a value of '0', the meta-gene is inactive and does not change the transcription of the following genes. But, when activated (meta-bit set to '1'), we transcribe each of the following genes simply by considering its complementary allele. This expressed phenotype will be evaluated by the fitness function we want to optimize. For instance, if we consider the chromosome [011] it can be transliterated by $[\dot{0}\ 011]$ or $[\dot{1}\ 100]$.

In a more formal way, let us consider λ-bit strings and let us note $\Omega = \{0,1\}^\lambda$ the search space. We define the *dual space* as $<\!\Omega\!> = \{0,1\} \times \Omega$. We also have to define a mapping T, called *transliteration function*, from the dual space $<\!\Omega\!>$ to the basic one Ω as:

$$\forall \omega \in \Omega, \quad T(\dot{0}\ \omega) = \omega \text{ and } T(\dot{1}\ \omega) = \omega'$$

where ω' is the complementary of ω.

Thus, chromosomes handled by a DGA are $\lambda + 1$ genes long but transliterated into λ long strings for evaluation. This clearly involves two facts: at first we have doubled, for a binary allelic alphabet, the size of the search space. But this one has also become symmetrical and now the GA works on populations that may contain symmetrical strings, *dual chromosomes*, featuring identical fitness (as they are transcribed into same phenotypes) while having complementary genotypes. DGAs feature interesting properties directly stemming from the use of such symmetrical structures combined with the dynamics of a SGA.[25,26,27]

Among these, let us consider their ability to converge from a phenotypical point of view while keeping complementary chromosomes in their population. A special operator called *mirroring* that transforms any chromosome in its dual counterpart with a constant rate τ, helps at keeping a minimal amount of diversity in the population. We will see later that this diversity can be used by the crossover operator to keep acting on dual pairs of chromosomes even when the GA has converged and so to introduce new individuals without the help of mutation. But, let us see now how schemata are performed within DGAs.

5.3. *Schemata and dualism*

Since strings manipulated by DGAs are of length $\lambda + 1$, so are schemata. Thus, considering the dual space $<\Omega>$, schemata are strings of $\lambda+1$ characters taken from the alphabet $\{0, 1, \star\}$. Let us note Ω^* the set $\{0, 1, \star\}^\lambda$ and $<\Omega>^*$ the set $\Omega^* \times \{0, 1, \star\}$. Depending on the value of the meta-bit, we can distinguish two cases:

1. The meta-gene is *specified* (it is set to '0' or to '1'): the transliteration is obvious. For instance the schemata [$\dot{0}$ 1$\star$1] and [$\dot{1}$ 0$\star$0] belonging to $<\Omega>^*$ are associated to the schema [1$\star$1] of Ω^*.

2. The meta-bit is *undetermined* (it is set to '$\star$'): the use of variables is needed. For example, let us consider the schema [$\dot{\star}$ 01$\star$0] of $<\Omega>^*$. If the meta-bit is for a '0' the transliteration gives the schema [01$\star$0]; if it is for a '1', the transliteration results in the schema [10$\star$1]. In fact those two schemata from Ω^* express the fact that the first and the last bits are identical, but different from the second one. This can be summarized by the *relational schema* (or R-schema): [XX'$\star$X] where X and X' represent two complementary bits. More generally, the transliteration of such a schema of $<\Omega>^*$ is a R-schema of $\{X, X', \star\}^\lambda$ obtained as follows:

   (a) substitute the first specified value and its occurrences, for the symbol X,

   (b) substitute all the complementary values for X',

   (c) keep all the undetermined gene.

We just saw that R-schemata i.e. strings belonging to the set $\{X, X', \star\}^\lambda$, express relations between genes of a schema. Standard schemata, i.e. those defined over $\{0, 1, \star\}^\lambda$, can only express values on the different positions of the string. That is why, in the rest of this paper we will call them *positional schemata* or P-schemata. Let us now interest ourself in the properties of R-schemata.

5.4. *Properties of R-schemata*

Here, we study R-schemata considering the four properties previously stated (cf. §5.4). Our aim is not to show that R-schemata are better than P-schemata. The purpose is to show that they are complementary models.

Before considering the four properties one after the other, we must first extend the notion of compatibility to R-schemata: two R-schemata are said *compatible* if there exists a chromosome being an instance of both. For instance, [$\star$X$\star$X] and [XX'$\star\star$] are compatible. We can refine the notion of compatibility, considering *corroborating* schemata: two compatible R-schemata are corroborating if they share at least one variable locus. For example, the two previous R-schemata are corroborating while [$\star$XX'$\star$] and [X$\star\star$X'] are not.

5.4.1. *The closure*

The intersection of any pair of corroborating R-schemata is itself a R-schemata. For instance, the intersection of [$\star$X$\star$X] and [XX'$\star\star$] is the R-schema [XX'$\star$X']. We can notice that two non-corroborating R-schemata are necessarily compatible. Nevertheless, their intersection cannot be expressed as a single R-schema. For example, the intersection of [$\star$XX'$\star$] and [X$\star\star$X'] can only be expressed by [XYY'X'] which is not a R-schema since there are two distinct variables. However, this expression can be represented by the disjunction: ([XXX'X'] or [XX'XX']). More generally, we can say that the intersection of any non-corroborating R-schemata is a disjunction of R-schemata. In this sense, we say that R-schemata are *semi-closed* for intersection.

5.4.2. *The respect*

The respect property is related to the crossover of schemata, which are strings explicitly handled by GAs. In our context, these strings are in the dual space <Ω>. We are going to show that implicit R-schemata possess the respect property. More precisely, for each R-schema a choice between dual strings exists and allows the crossover operator to respect R-schemata.[28] Let us specify this choice. Let ρ be an instance of a R-schema of $\{$X, X', $\star\}^{\lambda}$. If the variable X corresponds to a '0' (respectively '1') we choose to represent ρ in $\{0\}\times\Omega^*$ (respectively in $\{1\}\times\Omega^*$). We can show that this choice creates a *R-similarity* set in <Ω>closed by crossover. For example, let be the R-schema [X$\star$X']. The R-similarity set of Ω, {[001], [011], [100], [110]}, is not closed, but the corresponding R-similarity set of <Ω>, {[$\dot{0}$ 001], [$\dot{0}$ 011], [$\dot{1}$ 011],[$\dot{1}$ 001]} is the P-schema [$\dot{\star}$ 0$\star$1]. We can conclude that a DGA, through its choices between dual strings, allows to handle R-schemata of Ω^* which possess the respect property.

5.4.3. *The proper assortment*

Let us consider the corroborating R-schemata [$\star$X$\star$X] and [XX'$\star\star$]. Let [0000] and [0100] be two respective instances. A 1-point and a uniform crossover breed the children [0000] and [0100]. No offspring is an instance of the intersection [XX'$\star$X']. Let us choose to represent the string [0000] by the chromosome [$\dot{1}$ 1111] and [0100] by [$\dot{0}$ 0100]. A uniform crossover allows to obtain [$\dot{0}$ 0101] that can be transcripted by [0101] and which belongs to the intersection [XX'$\star$X']. More generally, let a and b be two instances of two corroborating R-schemata of $\{$X, X', $\star\}\lambda$. Two equivalent

P-schemata of $\{0,\ 1,\ \star\}^\lambda$ a' and b' exist and a uniform crossover between them generates an instance of the intersection. We can conclude that, through judicious choices between dual strings, a DGA allows uniform crossover to properly assort R-schemata.

5.4.4. *The ergodicity*

The property of ergodicity makes possible to access any point in the search space, given any population, through a finite sequence of application of genetic operators. The mutation operator usually ensures this property. Obviously, by applying a sequence of mutations, it is possible to reach any point in the space. This property is qualitatively independent of the space (Ω or $<\!\Omega\!>$) on which we apply the mutation operator. The only improvement we can expect using the dual space is quantitative. In the basic space Ω, the minimum number of mutations between two strings is the Hamming distance. For two strings of $\Omega(\lambda)$, this number is smaller than the length λ. We can show that, in the dual space, $<\!\Omega\!>$, the minimum number of mutations between two points is smaller than$(\lambda/2)+1$. Let us consider a λ-dimensional space. Each point in this space can be considered as a vertex of the λ-dimensional unit hypercube. The greatest Hamming distance between two points is equal to λ (this is the case when the two strings are complementary ones). So, with the standard approach, λ mutations are needed to go from one chromosome to the other. When we apply the mutation operator on the dual space, we are able to cross from a string in the basic space to its complement by only one mutation on the meta bit. In this way, two complementary strings are at a Hamming distance of 1 from each other.

Now we have shown the relevance of DGAs from a schemata point of view, let us now devote ourselves to see how DGAs maintain diversity.

6. Diversity and DGAs

We saw earlier that preserving diversity is essential for a GA when facing dynamical environments. Basically, the mutation operator is the simplest solution to maintain diversity. The mutation rate is mostly handled as a global, external parameter which is constant over time.[29] We enlighten here that within DGAs, explicit mutations are unnecessary. Indeed, dualism implies that mutation rates adapt during the search process. The reduction of the number of external parameters can been seen as a first step toward achieving a problem-dependant self-adaptation of the algorithm.

6.1. *Implicit mutations*

We describe here some of our basic concepts and the notations used. Let us first define the *d-space* $= \{(\dot{0}\ \omega, \dot{1}\ \omega')\}$ where $\omega \in \Omega$. This space contains all pairs of doubles. It is straightforward to verify that this space is closed for the crossover operator: for each pair of dual strings, the two offsprings form a dual pair. We can notice that the transliteration function, T, allows to define a mapping between the *d-space* and the search space:

$$\begin{array}{rrcl} T: & d\text{-}space & \longrightarrow & \Omega \\ & (\dot{0}\,\omega, \dot{1}\,\omega') & \longrightarrow & \omega = T(\dot{0}\,\omega) = T(\dot{1}\,\omega') \end{array}$$

This matching leads to an equivalence between the transformations on the *d-space* and the ones on Ω. Particularly, we can compare a crossover on the *d-space* with a transformation on the search space.

6.1.1. *Dual crossovers*

Let us note χ_s^λ the crossover applied on the site s to λ-bit strings, $\{\chi_1^\lambda, \ldots, \chi_\lambda^\lambda\}$ the set of crossover operators on the *d-space* and Γ_χ^λ the subspace generated by these crossover operators.

We can show that the subspace Γ_χ^λ is an *Abelian group*. As an example, let us consider binary strings of length $\lambda = 2$. The group Γ_χ^2 contains four elements: the crossover χ_1 on site 1, the crossover χ_2 on site 2 and the compound terms $\chi_1 \circ \chi_2$ and $\chi_1 \circ \chi_1$.* We can interpret in a geometrical way those four transformations:

1. χ_1 is the central symmetry,
2. χ_2 is the axial symmetry yy',
3. $\chi_1 \circ \chi_2$ is the axial symmetry xx',
4. $\chi_1 \circ \chi_1$ is the identity, we note χ_0.

As an example, let us examine the transformation χ_1. We have:

$$\begin{array}{rcl} ([\dot{0}\,00], [\dot{1}\,11]) & \xrightarrow{\chi_1} & ([\dot{0}\,11], [\dot{1}\,00]) \\ ([\dot{0}\,10], [\dot{1}\,01]) & \xrightarrow{\chi_1} & ([\dot{0}\,01], [\dot{1}\,10]) \\ ([\dot{0}\,01], [\dot{1}\,10]) & \xrightarrow{\chi_1} & ([\dot{0}\,10], [\dot{1}\,01]) \\ ([\dot{0}\,11], [\dot{1}\,00]) & \xrightarrow{\chi_1} & ([\dot{0}\,00], [\dot{1}\,11]) \end{array}$$

If we now map χ_1 on Ω, we obtain:

$$\begin{array}{lcl} [00] \xrightarrow{\chi_1} [11] & \qquad & [01] \xrightarrow{\chi_1} [10] \\ [10] \xrightarrow{\chi_1} [01] & \qquad & [11] \xrightarrow{\chi_1} [00] \end{array}$$

As expected, χ_1 is the central symmetry. Let us now have a look at the mutations.

6.1.2. *The space of mutations*

We have previously defined the subspace Γ_χ^λ generated by crossover operators on the *d-space*. Now, we study the subspace Γ_μ^λ generated by mutations operators on Ω. For instance, with binary strings of length $\lambda = 2$, the mutation μ_1 on the first locus is:

$$\begin{array}{lcl} [00] \xrightarrow{\mu_1} [10] & \qquad & [01] \xrightarrow{\mu_1} [11] \\ [10] \xrightarrow{\mu_1} [00] & \qquad & [11] \xrightarrow{\mu_1} [01] \end{array}$$

*$\chi_1 \circ \chi_2 = \chi_2 \circ \chi_1$ and $\chi_1 \circ \chi_1 = \chi_2 \circ \chi_2 = Id_\Omega$

Thus, the axial symmetry xx' is the geometrical interpretation of mutation μ_1. More generally, we have the four following interpretations:

1. μ_1 is the axial symmetry xx',
2. μ_2 is the axial symmetry yy',
3. $\mu_1 \circ \mu_2$ is the central symmetry,
4. $\mu_1 \circ \mu_1 = \mu_2 \circ \mu_2$ is the identity, noted μ_0.

6.1.3. *Dual crossovers vs. mutations*

Obviously, we are able to assert that groups Γ_χ^λ and Γ_μ^λ are isomorphic. Taking again our example in $\Omega(2)$, we can establish the mapping:

$$\begin{array}{rcl} \chi_1 & \longleftrightarrow & \mu_1 \circ \mu_2 \\ \chi_2 & \longleftrightarrow & \mu_2 \\ \chi_1 \circ \chi_2 & \longleftrightarrow & \mu_1 \\ \chi_1 \circ \chi_1 & \longleftrightarrow & \mu_1 \circ \mu_1 \end{array}$$

More generally, the mapping between mutations and crossovers is:

$$\forall i \in [1, \lambda[\quad \mu_i \longleftrightarrow \chi_i \circ \chi_{i+1} \quad \text{and} \quad \mu_\lambda \longleftrightarrow \chi_\lambda$$

The figure 5 represents this mapping for 2-bit chromosomes.

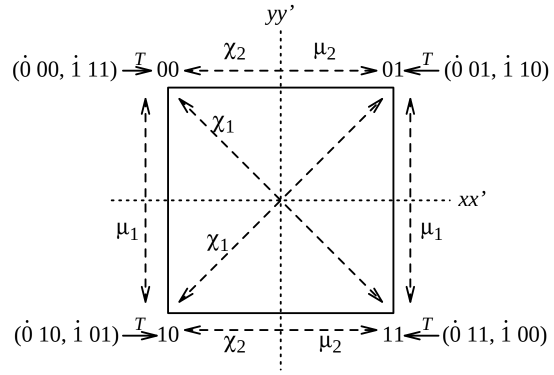

Figure 5: A geometrical interpretation of crossovers and mutations in $\Omega(2)$.

So, the effect of a mutation may be achieved by successive crossovers applied to a pair on two consecutive sites. For example, if a crossover on site 2 applied to the dual pair ([$\dot{0}$ 101], [$\dot{1}$ 010]) is followed by a crossover on site 1 on the obtained offspring pair ([$\dot{0}$ 110], [$\dot{1}$ 001]), then a mutation on locus 2 on chromosome [101] has been achieved.

This result shows that mutations can be *implicitly* performed with a DGA. Furthermore, in SGAs, the mutation rate is constant over time. We are going to show that within DGAs, the implicit mutation rate adapts itself during the search process.

6.2. *Adaptive mutation rates*

Obviously, the number of implicit mutations is dependant on the number of dual pairs in the population. We have already established that pairs of doubles tend to decrease as the DGA converges.[24] Thus, we can deduce from this, that the mutation rate decreases from an initial value to zero. From these observations we are able to conclude that the algorithm balances well between a mutation rate as high as needed for an efficient search, but without destroying useful information, thus it works on the borderline of efficient optimization and almost random walk.

6.2.1. *The implicit mutation rate*

To emphasize this evolution, let us define the *Implicit Mutation Rate* more strictly:

$$IMR = p_c \cdot \sum_{d\text{-}space} \min(P(\dot{0}\,\omega), P(\dot{1}\,\omega'))$$

where p_c is the crossover probability, $P(\dot{0}\,\omega)$ is the proportion of the chromosome $\dot{0}\,\omega$ in the current population and $P(\dot{1}\,\omega')$ is the proportion of its dual string. We are summing over all pairs i.e. on the *d-space*. To illustrate this notion let us take the example of the Goldberg's *Minimal Deceptive Problem* (MDP) which should cause a GA to diverge from the global optimum.[6] The function considered for this problem is $F([00]) = 4$, $F([01]) = 3$, $F([10]) = 1$ and $F([11]) = 5$, and the initial proportions of the individual of $<\Omega>$ are: $P([\dot{0}\ 00]) = 0.2$, $P([\dot{1}\ 11]) = 0.5$, $P([\dot{0}\ 01]) = 0.08$, $P([\dot{1}\ 10]) = 0.02$, $P([\dot{0}\ 10]) = 0.07$, $P([\dot{1}\ 01]) = 0.03$, $P([\dot{0}\ 11]) = 0.04$, $P([\dot{1}\ 00]) = 0.06$. If $p_c = 0.6$, we obtain an initial IMR of $0.6 \times (0.2 + 0.02 + 0.03 + 0.04) = 0.174$. Figure 6 shows the result of a simulation on our MDP: we can see the IMR decreases as the algorithm converges toward the optimum.[†]

6.2.2. *Implicit mutations and robustness*

Robustness expresses the fact that the disappearance of individuals from the population has little effect on the global system behavior. Particularly, after a "catastrophe", a robust GA must be able to steady itself. Obviously, this ability is linked to the diversity. We are going to show that robustness is, for DGAs, an intrinsic characteristic. For instance, to simulate a "catastrophe" in our MDP we remove all the instances of the optimum individual (represented as $[\dot{0}\ 11]$ and $[\dot{1}\ 00]$) from the population at generation 15. Figure 7 shows that the IMR increases just after the "catastrophe" revealing the increasing number of dual pairs in the current population. This growing is a spring of diversity and so allows the DGA to find the optimum individual again.

[†]It is worth noticing that a SGA faced with the same problem is unable to converge toward the global optimum.

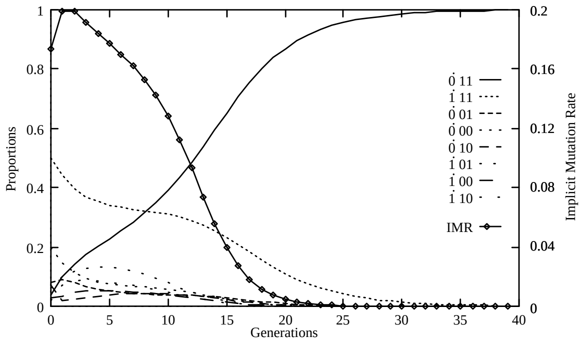

Figure 6: Evolution of the proportions and of the IMR over generations for the MDP problem.

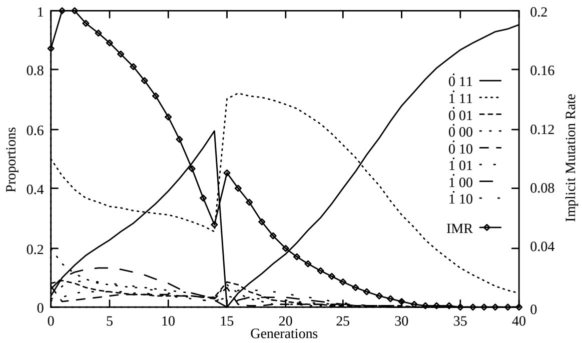

Figure 7: Evolution of the proportions and of the IMR over generations. A "catastrophe" is simulated at generation 15.

Preserving diversity makes DGAs to be great candidates for managing in environments featuring a fitness function changing over time.

7. Empirical Study

This section investigates the dual approach experimentally by comparing it to the standard one on the basis of the previously presented testbed (cf. §3.3).

7.1. *Simulation on a mathematical model*

Equational simulation of a genetic algorithm are a relevant tool to reveal whether one particular approach is efficient on a given problem. In the following, we present the results obtained by simulation of dynamics of both SGA and DGA in the minimal dynamical problem (cf. §2.4). Our analysis relies on the equational system previously presented (cf. §2.4). In both cases, the crossover probability is set to 0.8. For the SGA, initial proportions are: $P([00]) = 0.1$, $P([01]) = 0.2$, $P([10]) = 0.3$ and $P([11]) = 0.4$. Concerning the DGA, the same proportions are applied (i.e $P(\dot{0}\ \omega) + P(\dot{1}\ \omega) = P(\omega)$). They are: $P([\dot{0}\ 00]) = 0.07$, $P([\dot{1}\ 11]) = 0.03$, $P([\dot{0}\ 01]) = 0.05$, $P([\dot{1}\ 10]) = 0.15$, $P([\dot{0}\ 10]) = 0.15$, $P([\dot{1}\ 01]) = 0.15$, $P([\dot{0}\ 11]) = 0.15$, $P([\dot{1}\ 00]) = 0.25$. The mirrorring operator is applied with a rate set to 0.01. Figures 8 plot the density of each individual of the search space over time. For the DGA, the evolution of the IMR is also represented.

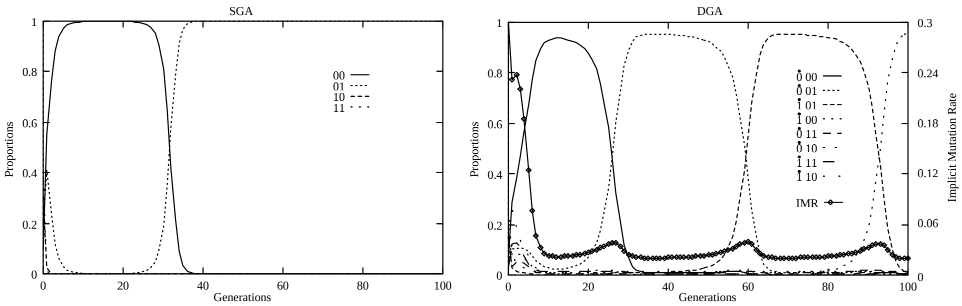

Figure 8: Simulations with the minimal dynamical problem.

The SGA only manages to find the two first optima (chromosomes [00] and [01]). As for the DGA, the four consecutive optima are found. It seems able to hang to each transition providing a new genetic exploration of the search space when needed. This exploration is enabled by the presence of complementary strings in the population (dual chromosomes).

Beyond these efficient behavior, we can observe that the proportion between dual chromosomes changes according to the transitions of the environment. As previously said, DGAs feature, via crossover, a new way to introduce diversity in the population.

The IMR rate reveals to evolve dynamically according to the changes of density within a dual pair. One can observe on the figure 8 that the IMR increases for each change of optimum, thus providing an enhanced exploration of Ω whenever needed by the GA to fit its environment.

These simple simulations obviously advocate our hypothesis that a DGA constitute a relevant approach to time dependant optimization. In next sections we are going to empirically investigate their efficiency on larger dynamical environments involving multi-dimensional functions.

7.2. *Multi-dimensional Gaussians*

This section is aimed at presenting an empirical evaluation of a DGA's effectiveness when confronted to more relevant dynamical environments based on 64 genes long chromosomes. The function used is the multi-dimensional Gaussian previously described in section 3.3.2) and generalized at 8 dimensions. Thus, each chromosome is encoding 8 variables on 8 genes.

In order to reduce the influence of the initial population, each result presented hereafter has been averaged over 10 experiments. Each experiment has been realized using 64 bits length chromosomes encoding 8 integer values each 8 genes long. The GA has been run over 5000 generations leading to a relevant translation speed. During this translation, each of the 8 segments of a chromosome has taken an identical value upon the 2^8 possible ones corresponding to an enumeration by a simple binary coding. For these experiments, the *peakness coefficient* has been settled to $\pi = 3$ in order to avoid the "needle in the haystack" phenomenon that would have lead to a function too difficult to optimize whatever optimization method we choose.

All other "classical" parameters have been settled with traditional values: crossover rate at 70%, mutation rate at 0.1% and mirroring rate at 1%. The crossover method was one point crossover and selection was assumed by a stochastic sampling method without elitism. The population has been sized to 100 individuals both for SGA and DGA. Results obtained with the latter should be interpreted as lower bound for its ability to deal with dynamical environments because of the doubling of search space size due to the meta-gene.

Figure 9 plots the fitness of the best individual for each generation. This is a relevant criterion for analysis according to the fact that our function features consecutive optima having the same fitness (1.0). An ideal plot would thus consist in an horizontal line at ordinate 1.0. This should involve that the GA has found the optima at each generation, thus providing an exceptional adaptiveness. Experimentally, plotting this criterion allows us to observe difficulties encountered by the GA during its adaptation. The figure clearly shows that using a DGA leads to better results than those obtained with a SGA. The average value, over all generations, of the best fitness value provides us with a relevant numerical measure of this advantage (0.936 for SGA versus 0.968 for DGA).

This clearly advocates that the DGA keeps nearer of the consecutive optima than the SGA thus featuring an improved adaptation over dynamical changes to

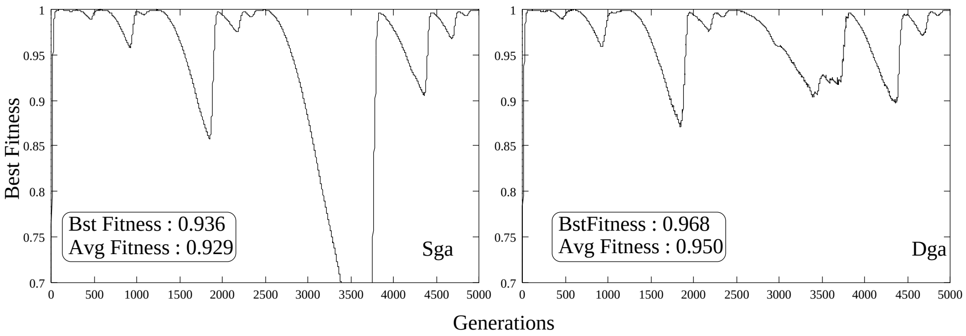

Figure 9: Best Fitness over Time.

the fitness function shape. If we look at the plotting, we can see that the difficulties encountered by both GAs are more important at regular locations. In fact, the hardness seems proportional to the hamming distance between two consecutive optima. For example, the SGA drops off mainly at the middle and quarter transitions precisely where hamming distances are respectively of λ and $\lambda/2$.

For the DGA the situation is almost inverse: during the more difficult transitions, the DGA keeps hanging on thus offering a more adaptive behavior. Moreover, this property seems proportional to difficulties encountered. Furthermore, the DGA keeps very close to the SGA for easier transitions. Results presented are thus encouraging for the dual approach as this one revealed to increase performances proportionally to transition's difficulty.

7.3. *Progressive Pattern Tracking*

In this section, the two GAs are faced to the progressive pattern tracking problem (cf. §3.3.3). Each chromosome is 40 genes long and the tracked optimum is changed every $\delta_p = 50$ generations for a new one located at a distance δ_d which is increased by 5 at each transition. Figure 10 plots the best fitness over time averaged on 10 experiments. All the parameters of both SGA and DGA were settled as in the previous experiment.

Results show that the difficulty of a dynamical environment seems effectively directly proportional to the δ_d parameter (SGA). First of all we can notice that average efficiency decreases over time as effects of increasing δ_d and thus difficulty is cumulated. With a longer δ_p parameter, the genetic algorithm is more about to recover its high fitness value. Increasing distances of transitions has lead us to confirm DGA's efficiency in somewhat harder dynamical environments. DGA also features a minimal bound for the unhook as they provide in the population, through mirroring, the dual counterpart of the optimal chromosome. Indeed it can not unhook of more than half the chromosome length ($\lambda/2$) as soon as population contains the dual counterpart of the current optimum. This property is fundamental

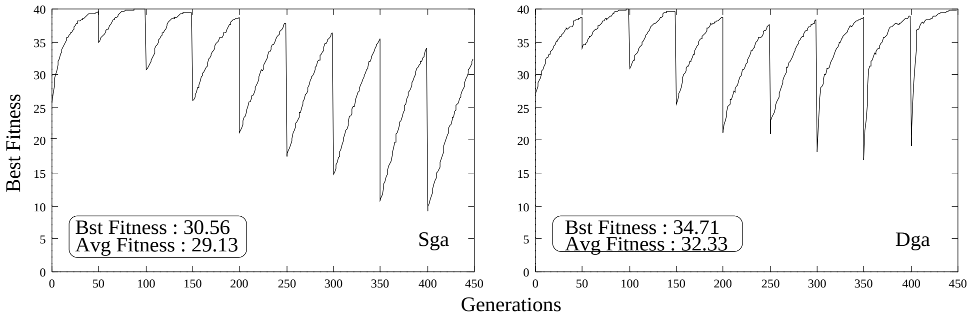

Figure 10: Pattern Tracking Results.

to explain their improved efficiency when compared to SGA thus complementing previous analysis.

8. Discussion and Further Work

We eventually summarize our analysis of the improved efficiency provided by the dual approach on diversified Dynamical Environments and then enlighten further extensions currently investigated.

8.1. *Dualism as an efficient tools*

All previously presented results can be entirely explained by the intrinsic nature of DGA. The transition difficulty is directly proportional to the hamming distance between two consecutive optima. This difficulty measure is only effective according to the hypothesis that the exploration of the GA's search space only relies on the concept of hamming distance. In the case of a DGA, there exists an other metric that can be considered during the search process. By its ability to deal with complementary chromosomes (dual chromosomes), the DGA is able to keep, even while converging from a phenotypical point of view, a genotypical diversity by the coexistence of both dual chromosomes which are decoding via transliteration to the same current solution.

Let us consider the hardest transition for a SGA; it only consists of taking the new optimum as a chromosome separated by a λ hamming distance from the previous one. This clearly involves to take the complementary string. In a DGA, the genetic material needed to obtain such a string by application of crossover operator only is directly available, thus the distance between the two optima is one (mutation of the meta-gene in one of the dual chromosomes for example). In fact, we can call this distance, defined over Ω, a *dual distance* that shows how a DGA can reach any point of the search space by taking the complementary string of its current location in a single step. If $\delta_H(X,Y)$ stands for the hamming distance, the dual distance is : $\delta_D(X,Y) = \delta_H(\overline{X},Y)$

Figure 11 first illustrates Hamming distance on a simple example involving $\lambda = 6$

genes long chromosomes. When considering Multi-dimensional Gaussian like environments, we would have as successive optima each individual according to a binary enumeration. We plotted, for such an enumeration of the search space, the hamming distance involved by each transition.

It is clear that worst transitions, i.e. the one involving the highest distance between the two optima, is a λ one for SGA. If we look back to experimental results in comparison with these plottings, we can state that a DGA performs better than a SGA when its metrics over the search space is the most suitable. Moreover, a DGA seems to be able to dynamically choose which distance, among the Hamming or the dual one, is more suitable to use depending on the δ_d of the transition it has to face. We can call this an *optimal distance* according to the exploration process and write it $\delta_{Opt}(X,Y) = Min(\delta_H(X,Y), \delta_D(X,Y))$

Indeed, DGAs have allowed to change the metrics in use over the search space and thus to decrease peaks of the precedent plotting. The second plotting on figure 11 summarizes and illustrates this optimal (i.e. effective) distance comparing to the Hamming one. As the hamming distance is fully explaining the SGA's behavior,

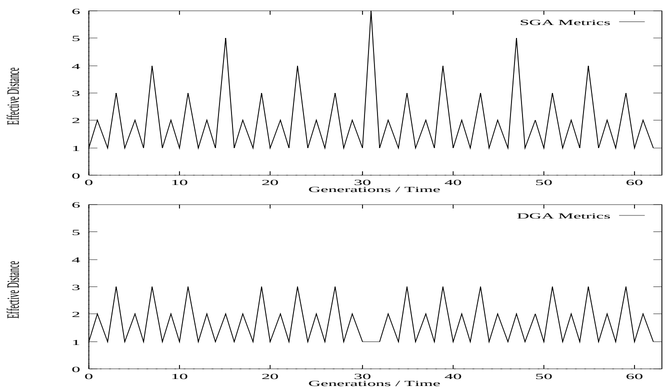

Figure 11: Distances between two consecutive optima (SGA vs DGA).

the optimal one corresponds exactly to results obtained by DGA.

The dynamical choice of a good metric over the search space Ω explains the behavior featured during experiments and the results presented in previous section. This also explains the superiority of the dual approach on this new field and opens to DGA perspectives closed to SGA. Choosing dynamically a metric on their search space appears to be one of the fundamental DGA's properties. Furthermore, this represents a new step toward the design of adaptive autonomous entities and its study may be a relevant part of the future work concerning the dual approach.

8.2. *Perspectives and further work*

Dynamical environments constitute an important challenge for any optimization method devoted to real time applications and moreover to all optimization methods that need keeping their adaptiveness after reaching a potential solution.

This paper has been partly devoted to introduce dynamical environments in a somewhat formal way by providing a way to quantify the dynamical difficulty. Dynamical Environments and fitness dynamism can be viewed as a generic framework engulfing most of the tremendous issues the genetic algorithms community is concerned with.

Along with this work on dynamical environments themselves, we provided a new approach called *Dual Genetic Algorithm* based on the use of *meta-genes* and tested it in three different dynamical environment. The empirical study has confirmed theoretical simulations: DGA revealed to be a particularly promising tool for time dependant optimization. This effectiveness has been shown to come directly from their ability to deal with transitions featuring high hamming distances between two consecutive optima. Th distance between old and new optimum has revealed to be a relevant measure of difficulty for these environments and suggests further work to clearly establish its usefulness to evaluate function difficulty from a dynamic point of view. The introduction of a transliteration phase has explained DGA's enhanced adaptiveness, allowing them to handle dual chromosomes and thus converging to an uniform population (from a phenotypical point of view) while keeping various genotypes. This diversity is maintained by the application of the mirroring operator which enforces the preservation of similar proportions of each member of a dual chromosomes pair. Such a behavior leads us to observe that the mirroring operator builds symmetry, while the natural dynamics of the GA breaks these attractors in the populations space. Both antagonist dynamics are present but acting at different levels; diversity for genotypes handled and uniformity for the phenotypes expressed. This property seems to be the key of an efficient dynamical function optimization.

Beyond this problem, we can notice that preserving diversity is also a promising gap for solving numerous problems GAs are confronted with. Thus, our work should be extended to investigate methods for preserving genotypical diversity and studying their effectiveness not only on dynamical functions but also on deceptive problems for example. We should also notice that they should cover domains like multicriterion optimization, multimodal optimization where diversity is the *sine qua non* condition for finding out a majority of niches for example. This work underlines a fundamental problem for GAs and investigating how to build A reacting faster to their environment yields to increasing their main characteristic : adaptiveness. Further work may consist in generalizing our dual model to the use of any number of meta-genes thus extending dualism inherent properties.

References

[1] J. H. Holland. *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press, 1975. Second printing in 1993, The MIT Pres.

[2] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley, 1989.

[3] N. J. Radcliffe. Forma analysis and random respectful recombination. In R. K. Belew

and L. B. Booker, editors, *ICGA'91: Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 222–229, San Mateo, CA, 1991. Morgan Kaufmann.

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

[5] L. D. Whitley. An executable model of a simple genetic algorithm. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 45–62. Morgan Kaufmann, 1993.

[6] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 74–88. Morgan Kaufmann, Los Altos, California, 1987.

[7] M. D. Vose and G. E. Liepins. Punctuated equilibria in genetic search. *Complex Systems*, 5:31–44, 1991.

[8] M. D. Vose. Generalising the notion of schema in genetic algorithms. *Artificial Intelligence*, 50:385–396, 1991.

[9] D.E. Goldberg. *Algorithmes Génétiques -Exploration, optimisation et apprentissage Automatique-*. Vie Artificielle. Addisson Wesley, 1994.

[10] K. Pettit and E. Swigger. An analysis of genetic based pattern tracking and cognitive based component tracking models of adaptation. In *Proceedings of National Conference on AI (AAAI-83)*. Morgan Kaufmann, 1983.

[11] F. Vavak and T.C. Fogarty. A comparative study of steady state and generational genetic algorithms for use in nonstationary environments. In *Proceedings of the Society for the Study of Artificial Intelligence and Simulation of Behavior, workshop on Evolutionnary Computation'96*, pages 301–307. University of Sussex, 1996.

[12] D.E. Goldberg and R.E. Smith. Nonstationnary function optimization using genetic algorithms with dominance and diploidy. In *Icga-2*, 1987.

[13] J. Koza. *Genetic Programming: On the Programming of Computers by the mean of Natural Selection*. MIT Press, 1992.

[14] J.J. Grefenstette. Genetic algorithms for changing environments. In R. Manner abd B. Manderick, editor, *Parallel Problem Solving from Nature 2*, pages 465–501. Elsevier Science Publishers B.V., 1992.

[15] H.C. Cobb and J.J. Grefenstette. Genetic algorithms for tracking changing environments. In *Icga-5*, 1993.

[16] L.M. Rocha. Contextual genetic algorithms: Evolving developmental rules. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, *ECAL-95, Advances in Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*. Springler, June 1995.

[17] A.S. Wu and R.K. Lindsay. A survey of intron research in genetics. In H-M Voigt, W. Ebeling, I. Rechenberg, and H-P Schwefel, editors, *International Conference on Evolutionnary Computation : Parallel Problem Solving from Nature IV*, volume 1141 of *Lecture Notes in Computer Science*, Berlin Germany, September 1996. Springer Verlag.

[18] J.R. Levenick. Inserting introns improves genetic algorithms success rate: Taking a cue from biology. In R.K. Belew and L.B. Booker, editors, *Fourth International Conference on Genetic Algorithms*, pages 123–127, San Francisco California, July 1991. Morgan Kaufmann.

[19] J.R. Levenick. Metabits: Generic endogenous crossover control. In L.J. Eschelman, editor, *Sixth International Conference on Genetic Algorithms*, pages 88–95, San Francisco, California, July 1995. Morgan Keaufmann.

[20] C. Allais, editor. *La Recherche sur la Génétique et l'Hérédité*. Point Sciences. Seuil, 1985.

[21] A. Gaspar and P. Collard. Symmetrical foldings as the royal road toward an optimal

chromosome's expressiveness. In *International Computer Science Convention, 2nd ICSC Symposium on soft computing (SoCo'97)*, 1997. To be published.

[22] M. Mitchell and S. Forrest. Genetic algorithms and artificial life. *Artificial Life*, 1(3):267–289, 1994.

[23] P. Collard and C. Escazut. Fitness distance correlation in a dual genetic algorithm. In *Ecai'96*, 1996.

[24] P. Collard and C. Escazut. Genetic operators in a dual genetic algorithm. In *ICTAI'95: Proceedings of the seventh IEEE International Conference on Tools with Artificial Intelligence*, pages 12–19. IEEE Computer Society Press, 1995.

[25] P. Collard and C. Escazut. Fitness Distance Correlation in a Dual Genetic Algorithm. In W. Wahlster, editor, *ECAI 96: 12th European Conference on Artificial Intelligence*, pages 218–222. Wiley & Son, 1996.

[26] P. Collard and A. Gaspar. Royal-road landscapes for a dual genetic algorithm. In W. Wahlster, editor, *ECAI 96: 12th European Conference on Artificial Intelligence*, pages 214–217. Wiley & Son, 1996.

[27] P. Collard, C. Escazut, and A. Gaspar. Genetic algorithms and relational landscapes. In H-M Voigt, W. Ebeling, I. Rechenberg, and H-P Schwefel, editors, *PPSN IV: The Fourth International Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science, pages 472–481, 1996.

[28] P. Collard and J.P. Aurand. Dga: An efficient genetic algorithm. In *ECAI'94*, Amsterdam, 1994.

[29] W. M. Spears. Crossover or mutation ? In L. D Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 221–233. Morgan Kaufmann, San Mateo, CA, 1993.