Self-Adaptation in Evolving Systems

C. R. Stephens
Instituto de Ciencias Nucleares, UNAM
Circuito Exterior
A. Postal 70-543
México D.F. 04510
stephens@nuclecu.unam.mx

I. García Olmedo
Lab. de Visualización
DGSCA, UNAM
Circuito Exterior, C.U.
México D.F. 04510

J. Mora Vargas
Facultad de Ingeniería, UNAM
Circuito Exterior, C.U.
México D.F. 04510

H. Waelbroeck
Instituto de Ciencias Nucleares, UNAM
Circuito Exterior
A. Postal 70-543
México D.F. 04510
hwael@nuclecu.unam.mx

Abstract A theoretical and experimental analysis is made of the effects of self-adaptation in a simple evolving system. Specifically, we consider the effects of coding the mutation and crossover probabilities of a genetic algorithm evolving in certain model fitness landscapes. The resultant genotype-phenotype mapping is degenerate in fitness space, there being no direct selective advantage for one probability versus another. Thus there is a "symmetry" between various genotypes that all correspond to the same phenotype. We show that the action of mutation and crossover lifts this degeneracy, that is, the genetic operators induce a breaking of the genotype-phenotype symmetry, thus leading to a preference for those genotypes that propagate most successfully into future generations. We demonstrate that this induced symmetry breaking allows the system to self-adapt in a time-dependent environment.

Keywords
self-adaptation, genetic algorithm, parameter codification, effective fitness, induced symmetry breaking

1 Introduction

One of the most striking features of complex systems, especially in the biological realm, is the ability to adapt. Loosely speaking, this means "optimization" in a time- and/or position-dependent "environment." The Darwinian paradigm of natural selection offers an intuitive framework within which one may understand such adaptive behavior. Recently, a complementary paradigm has been suggested [13] that takes as its principal theme "spontaneous ordering." In this case advantageous or disadvantageous characteristics may appear via a "spontaneous" symmetry breaking. The principal theme of this article is to show that adaptation can also occur due to an "induced" symmetry breaking—induced by the action of the genetic operators other than selection. We will demonstrate the above in the context of the evolution of a genetic algorithm (GA) seen as a simple, artificial model of an evolving, adaptive, complex system.

Evolutionary algorithms, and in particular GAs, have played an increasingly important role in a wide variety of problems (see, e.g., [2] for a recent review). Two crucial ingredients of such algorithms are a) the existence of a population of bit strings/chromosomes, each one of which codifies directly or indirectly a possible solution to a problem; and b) a set of genetic operators that act on the strings. In the case of GAs the most popular and most studied operators are selection, mutuation, and crossover, though of course in principle the list is infinite. Many other evolutionary algorithms, such as evolutionary programming [6], rely solely on selection and mutation. It has been asserted that GAs are better at function optimization than evoluationary algorithms that rely solely on mutation and that this advantage is due to crossover [10].

© 1998 Massachusetts Institute of Technology

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

Much effort has been spent trying to understand the differing roles of selection, mutation, and crossover in the context of different fitness landscapes, and in particular on discovering what constitutes optimal values for the exogenous parameters, for example, mutation probability, crossover probability, and so forth [5, 8, 15]. The chief motivation behind this work is to find parameter values that maximize performance over a wide class of test functions. However, it has been shown [9] that the most efficient parameter settings depend on the specific function. There is also a clear difference between "offline" and "online" performance with regards to what constitutes optimal parameter settings [5, 8, 15], where offline emphasizes the best solutions found by the GA in each generation and online puts more emphasis on average performance.

Intuitively it is clear that time-dependent parameters should be utilized, as will be illustrated clearly in this article. In this regard it is possible to specify a priori a schedule for the parameter changes, as is done in simulated annealing with a cooling schedule. A more attractive possibility, however, is to code the parameters within the chromosomes themselves and allow the system to "self-optimize" via a genetic search through various possible parameter values. We will refer to the continual adjustment of these parameters according to changes in the landscape as *self-adaptation*. Such self-adaption first appeared in the context of evolution strategies [17] and has been studied by Bäck and collaborators [1] in the context of both evolution strategies and GAs. One of the prime points of interest was the derivation of optimal mutation rates on a string-by-string basis in the case of certain simple landscapes. Crossover has been considered in [16] where the actual crossing points were codified in the strings. Codification of the crossover probability itself was not considered (though see [3] for an early attempt in a nonepistatic landscape).

In this article we will consider the effects of parameter codification of a GA in a set of model fitness landscapes. The motivation behind choosing a set of simple landscapes is the same as that of De Jong [5], that is, to study GA behavior in a set of easily controlled settings that present canonical GA problems such as, for example, deception or premature convergence. Our principal aim here is not just to show that parameter codification can lead to better function optimization capabilities in GAs but rather to study how self-adaptation occurs in a simple complex system. This self-adaptation will be shown to come about not via the mechanism of Darwinian natural selection but via an induced symmetry breaking.

The format of the article will be as follows: In Section 2 we give a theoretical motivation for the use of self-adaptation by showing that optimum parameter values must be both landscape and time dependent and therefore extremely difficult to determine. In Section 3 we introduce the key concepts of the article: induced symmetry breaking and effective fitness, which offer a framework within which one may understand how self-adaptation works. In Section 4 we present the methodology behind the experiments we used to compare self-adapting and fixed parameter GAs. In Section 5 we present some of the explicit experimental results, while in Section 6 we draw some conclusions from the present work.

2 Optimal Mutation and Crossover Rates

In this section we make some observations about optimal parameter settings based on an analysis of the evolution equation for GAs derived in [18, 19]. The point of this section is to show that theoretical analysis, except for the most trivial of landscapes, cannot provide optimal parameter settings, due to the fact that the latter are population dependent and hence strongly time dependent. This fact will motivate the use of self-adaptation.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

The equation we use takes the following form:

$$P(c_i, t+1) = \mathcal{P}(c_i)P_c(c_i, t) + \sum_{c_j \neq c_i} \mathcal{P}(c_j \to c_i)P_c(c_j, t) \tag{1}$$

where

$$P_c(c_i, t) = P'(c_i, t) - \frac{p_c}{N-1}\sum_{k=1}^{N-1}(P'(c_i, t) - P'(c_i^L, t)P'(c_i^R, t)) \tag{2}$$

and

$$P'(c_i^L, t) = \sum_{c_j \supset c_i^L} P'(c_j, t) \tag{3}$$

involves a sum over all strings that contain the left half of the string c_i, and similarly for $P'(c_i^R, t)$. p_c is the probability to implement crossover and k is the crossover point. $P'(c_i, t) = (f(c_i, t)/\bar{f}(t))P(c_i, t)$ with $f(c_i, t)$ being the fitness of string c_i at time t, and $\bar{f}(t) = \sum_i f(c_i, t)P(c_i, t)$ is the average string fitness. The effective mutation coefficients are $\mathcal{P}(c_i) = \prod_{k=1}^{N}(1 - p(k))$, which is the probability that string i remains unmutated, $p(k)$ being the probability of mutation of bit k, which we assume to be a constant, though the equations are essentially unchanged if we also include a dependence on time; and

$$\mathcal{P}(c_j \to c_i) = \prod_{k \in \{c_j - c_i\}} p(k) \prod_{k \in \{c_j - c_i\}_c} (1 - p(k)) \tag{4}$$

where $\{c_j - c_i\}$ is the set of bits that differ between c_j and c_i and $\{c_j - c_i\}_c$, the complement of this set, is the set of bits that are the same. $\mathcal{P}(c_j \to c_i)$ is the probability that string j is mutated into string i. In the limit where the mutation rate p is uniform, $\mathcal{P}(c_i) = (1-p)^N$ and $\mathcal{P}(c_j \to c_i) = p^{d^H(i,j)}(1-p)^{N-d^H(i,j)}$, where $d^H(i, j)$ is the Hamming distance between the strings c_i and c_j. This evolution equation takes into account exactly the evolution of the mean number of strings and in the limit of a large population gives the string probability distribution.

We may now use the above equation to investigate the evolution of any function. One of particular interest is the increment in average population fitness per generation

$$\delta\bar{f} = \bar{f}(t+1) - \bar{f}(t) \tag{5}$$

which using the evolution equation (1) we can write as

$$\begin{aligned}\delta\bar{f} = &\sum_{c_i} f(c_i, t)\mathcal{P}(c_i)P_c(c_i, t)\\ &+ \sum_{c_i}\sum_{c_j \neq c_i} f(c_i, t)\mathcal{P}(c_j \to c_i)P_c(c_j, t) - \sum_{c_i} f(c_i, t)P(c_i, t)\end{aligned} \tag{6}$$

Note that this equation is highly nonlinear in the mutation rate p, but linear in the crossover probability p_c.

The values of p and p_c play a very important role in determining the success of a GA. The specific values required depend on what we mean by success. One way

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

to gauge success is by maximizing the average population fitness, $\bar{f}$, generation by generation. As $\delta\bar{f}$ depends only linearly on p_c, the value of p_c that optimizes fitness growth over one time step, if we neglect finite size effects, will be either 0 or 1. This is somewhat deceptive, however, in that the integrated fitness gain, $\bar{f}(t) - \bar{f}(0)$, will be highly nonlinear in p_c except when $t = 1$. For a given time step, however, from (2) one can see that whether or not p_c should be minimized or maximized depends on whether parts of highly fit strings are positively or negatively correlated in the selected parent population. If the correlation is negative, that is, $P'(c_i, t) < P'(c_i^L, t)P'(c_i^R, t)$, then the reconstruction of highly fit strings dominates over destruction and p_c should be maximized. To illustrate this we restrict attention for the moment to the case $p = 0$. One finds

$$\delta\bar{f} = \Delta\bar{f} - \frac{p_c}{N-1}\sum_{k=1}^{N-1}\sum_{c_i} f(c_i, t)(P'(c_i, t) - P'(c_i^L, t)P'(c_i^R, t)) \tag{7}$$

where $\Delta\bar{f}(t) = (\bar{f^2} - (\bar{f})^2)/\bar{f}$ is the variance in the population fitness. With the further restriction to the simpler case of a population of two bit strings (one can also think of this in terms of schemata of order two), one finds

$$\delta\bar{f} = \Delta\bar{f} - \frac{p_c}{\bar{f}^2}(f_{00} + f_{11} - f_{10} - f_{01})(f_{10}P_{10}f_{01}P_{01} - f_{00}P_{00}f_{11}P_{11}) \tag{8}$$

For crossover to be a net positive force we require that the second term in (8) be positive. For a linear fitness landscape, as $f_{00} + f_{11} = f_{10} + f_{01}$, crossover in this 2-bit problem is neutral, although we once again emphasize that this is at the level of one time step. An interesting corollary of this is that even in the presence of crossover, $\bar{f}$ is a Lyapunov function. However, for a deceptive landscape of type I or type II [7] one finds that the effect of crossover is destructive. In this case having low values of p_c would be beneficial. Of course, if finite size effects are important, having p_c too low may inhibit genetic diversity.

We can try to generalize this lesson beyond the case of 2-bit strings. In (7) one may see that in the sum over strings, if the landscape is deceptive in the sense that $P'(c_i^L, t)P'(c_i^R, t) < P'(c_i, t)$, then crossover will inhibit fitness growth; while if $P'(c_i^L, t)$ $P'(c_i^R, t) > P'(c_i, t)$ it will enhance fitness growth. In the former case a low p_c would be beneficial while in the latter a high value. Generally, the more deceptive the landscape the lower the optimum value of p_c. In this sense GA hard problems may be made GA easier by a reduction in the crossover probability. Note also that we may have deception with respect to one crossover point but not with respect to another; this would imply that a crossover probability, $p_c(k)$, that depends on the crossover point would be useful. The above shows that what constitutes an optimal value for crossover is very much landscape dependent; moreover it will generally be time dependent.

We now turn our attention to mutations. Once again for simplicity we will turn off the other genetic operator. We will also assume that p is constant. In this case

$$\delta\bar{f} = \sum_{c_i}\sum_{c_j} f(c_i, t)P'(c_j, t)p^{d^H(i,j)}(1-p)^{N-d^H(i,j)} - \sum_{c_i} f(c_i, t)P(c_i, t) \tag{9}$$

thus

$$\frac{d\delta\bar{f}}{dp} = \sum_{c_i}\sum_{c_j} f(c_i, t)P'(c_j, t)p^{d^H(i,j)-1}(1-p)^{N-d^H(i,j)-1}(d^H(i, j) - pN) \tag{10}$$

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

We now wish to find the extrema. Doing the explicit sums over c_i is of course very difficult. One solution is to choose $p^* = d^H(i, j)/N$; however, this requires making p string specific. One can get an estimate of an optimum p by solving $d\delta\bar{f}/dp = 0$ in a mean field approximation where we replace $d^H(i, j)$ by $\langle d^H(i, j)\rangle$, its expectation value over the population. This results in $p^* = \langle d^H(i, j)\rangle/N$. For a random population, where $\langle d^H(i, j)\rangle \sim N/2$, $p^* \sim 1/2$, which accords with intuition. Near the ordered limit where $\langle d^H(i, j)\rangle \sim 0$, $p^* \sim 0$, which once again accords with intuition. Clearly, however, p^* will be time dependent. Note that whether or not zero mutation rate in the ordered limit is optimal will depend on whether the population has ordered about the global optimum or a local optimum. In the case of the latter, $p = 0$ will hinder rather than help.

We see then that theoretically there is no golden rule for parameter optimization. Optimal parameters' values are both landscape and population, that is, time dependent. As empirical optimization of parameter values is both difficult and time consuming one is naturally led to the idea of allowing the GA to optimize itself via a codification of the parameters.

3 Parameter Codification and Symmetry Breaking

Parameter codification can be done in various ways. Here we choose the simple route of extending the size of the chromosome such that a part of the chromosome now carries information on the mutation and/or crossover probabilities. We assume that the fitness of a chromosome is not affected directly by the values of the parameters. This means that the genotype–phenotype mapping is now noninjective (many-to-one).

Consider the different classes of maps that may be defined: first, $f_G\colon G \to R^+$ where G denotes the space of genotypes and f_G is the fitness function that assigns a number to a given genotype; second, $f_Q\colon Q \to R^+$, where Q is the space of phenotypes. It should be emphasized that these mappings may also be explicitly time dependent. In fact this will normally be the case when the "environment" is time dependent. The maps may be injective (one-to-one) or surjective (for every genotype there exists a phenotype). If they are noninjective then there exist "synonymous" genotypes or phenotypes, that is, there is "redundancy" in the mapping. If we assume there exists a map $\phi\colon G \to Q$ between genotype and phenotype then we have $f_G = f_Q \circ \phi$, that is, the composite map induces a fitness function on G. The map ϕ we may fruitfully think of as being an "interpreter," in that the map translates the genotypic information into something we call the phenotype, where generally the fitness function will have a more intuitive interpretation.

One of the principal reasons for using an interpreter in GAs is that the original genotypic coding may not be the most efficient. This is the case for a binary coding in the standard scenario where selection, mutation, and simple crossover are the preferred operators. It has been found that a Gray coding [21] that maps Euclidean neighborhoods into Hamming neighborhoods is more effective [4].

In the problem at hand G is simply the set of chromosomes, including the parts that code for the parameter values. Q is now the space of truncated chromosomes where the genes that code for the different parameter values have been removed. The interpreter here is clearly noninjective. If we use an n_c-bit binary codification of the parameters, then the interpreter, and hence the genotypic fitness landscape, will have a 2^{n_c}-fold degeneracy; that is, for every phenotype there will be 2^{n_c} corresponding genotypes.

This genotypic synonym symmetry would be broken spontaneously in a finite breeding pool, by the theory of branching processes, this observation being the backbone of the Neutral Theory of molecular evolution [14]. However, there will also be an *induced*

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

symmetry breaking from the violation of the synonym symmetry by the genetic operators themselves.

If one considers the growth of a particular string over time, selection forces will take into account not only the selective advantage of this string but also its ability to produce well-adapted offspring, which can themselves produce well-adapted offspring, and so on. Because mutation and crossover act differently on synonymous strings, the synonyms will differ in their descendence, both in the passive sense of surviving mutations, and in the active sense of generating new genetic solutions. This implies that the time-averaged *effective fitness* function, defined as the growth rate of a string over many generations, does not respect the synonym symmetry. Thus, the effective fitness function provides a selective pressure that enhances the production of potentially successful mutants by selecting, among the synonyms, those that have a higher probability to generate well-adapted offspring.

We now discuss in the context of a very simple model the phenonmenon of induced symmetry breaking. The model we consider consists of three possible genotypes, A, B, and C, where each genotype can mutate to the two adjacent genotypes. A and C are synonyms in that they encode the same phenotype a, that is, $\phi(A) = \phi(C) = a$, while B encodes the phenotype b. In a random population, $p(A) = p(B) = p(C) = \frac{1}{3}$. If there is probability $\mu_1/2$ for A to mutate to B or C and probability $\mu_2/2$ for all other possible mutations then the evolution equation that describes this system in the large population limit is

$$P(c_i, t+1) = (1-\mu_i)P'(c_i, t) + (\mu_{i-1}P'(c_{i-1}, t) + \mu_{i+1}P'(c_{i+1}, t)) \tag{11}$$

$P(c_i, t)$ being the population fraction of genotype i. The probability of mutation from genotype i to any other genotype is μ_i, while μ_{i-1} and μ_{i+1} are the mutation rates from genotypes $(i-1)$ and $(i+1)$ to genotype i, respectively. If we assume a simple fitness landscape, $f_a = 2$, $f_b = 1$, then for $\mu_i = 0$ the steady state population is $P(A) = P(C) = 1/2$ and $P(B) = 0$. Thus we see the synonym symmetry is unbroken. However, for $\mu_i > 0$, the genotype distribution at $t = 1$ starting from a random distribution at $t = 0$ is $P(A) = (4 - 4\mu_1 + 3\mu_2)/10$, $P(B) = (1 + \mu_1)/5$, $P(C) = 4 + 2\mu_1 - 3\mu_2)/10$. Thus we see that there is an induced breaking of the synonym symmetry due to the effects of mutation. The full effect can be dramatically seen in Figure 1 where we have $\mu_1 = 0.1$, $\mu_2 = 0.01$. Clearly, the less mutable string C is strongly favored over the other synonym A.

The effective fitness [18, 19] defined via

$$P(c_i, t+1) = \frac{f_{\text{eff}}(c_i, t)}{\bar{f}(t)} P(c_i, t) \tag{12}$$

is from the evolution equation (1) given by

$$f_{\text{eff}}(c_i, t) = \frac{\bar{f}}{P(c_i, t)} \left(\mathcal{P}(c_i) P_c(c_i, t) + \sum_{c_j \neq c_i} \mathcal{P}(c_j \to c_i) P_c(c_j, t) \right) \tag{13}$$

Explicitly in this model

$$f_{\text{eff}}(c_i, t) = f_i + \frac{1}{P(c_i, t)} (\mu_{i-1} f_{i-1} P(c_{i-1}, t) + \mu_{i+1} f_{i+1} P(c_{i+1}, t) - \mu_i f_i P(c_i, t)) \tag{14}$$

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

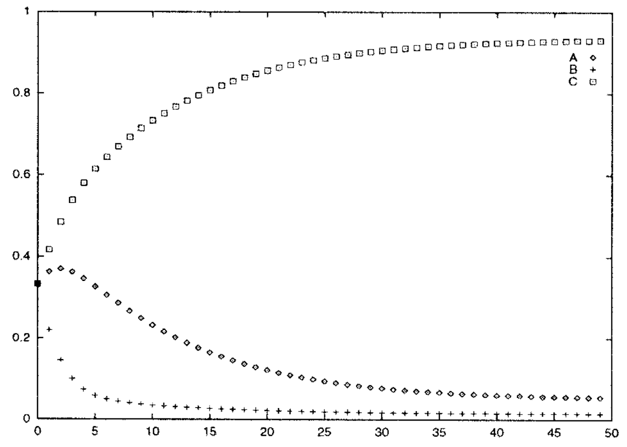

Figure 1. Graph of relative frequency as a function of time from the solution of Equation 11. The squares represent genotype *C*, the diamonds genotype *A*, and the crosses genotype *B*.

At $t = 0$, $f_{\rm eff}(A, 0) = (4 - 4\mu_1 + 3\mu_2)/2$, $f_{\rm eff}(B, 0) = (2 + 2\mu_1 - \mu_2)/2$, and $f_{\rm eff}(C, 0) = (4+4\mu_1-3\mu_2)/2$. For the case $\mu_1 = 0.1$, $\mu_2 = 0.01$; $f_{\rm eff}(A, 0) = 1.815$, $f_{\rm eff}(B, 0) = 1.095$, and $f_{\rm eff}(C, 0) = 2.195$. Thus as mentioned above we see that the effective fitness function provides a selective pressure by selecting among the synonyms those that have a higher probability to produce fit descendents.

4 Methodology

To illustrate the phenomenon of induced symmetry breaking and show that parameter codification yields significantly different results to those of a fixed parameter value GA, we chose several model fitness landscapes: a generic nondeceptive multi-modal function; a deceptive function; a time-dependent function; a traveling salesman problem of 33 cities, and finally a time-dependent traveling salesman problem. Once again we emphasize that we chose these functions as a "stripped-down" test-suite that in simplified settings manifest many of the most interesting landscape features that confront GA practitioners, for example, multi-modality, deception, premature convergence, time-dependence, and so forth. Throughout we used roulette wheel selection except at one point, which will be explicitly mentioned, where we used tournament selection of size 5.

The simple multi-modal function is seen in Figure 2. Rather than worry about how to design a real-valued, continuous, double-peaked function and then approximate it by binary numbers we simply assigned a fitness value to every integer between 0 and 63. This function is not deceptive in that crossover between optimal or near optimal strings does not produce very unfit strings, that is, crossover of strings near 10 and near 40 does not tend to produce strings between 20 and 30, or between 50 and 63. We considered a fixed population of 500 individuals. The basic chromosome was a 6-bit binary string representing the integers between 0 and 63. In coding the mutation and/or crossover probabilities we ranged between 3- and 16-bit additions to the basic chromosome depending on whether one or both parameters were coded and whether or not they were coded using 3-bit or 8-bit binary representations, the latter obviously giving a finer representation.

The second landscape we chose to investigate is shown in Figure 3. The landscape is deceptive in the sense that crossover between fit strings associated with the optima

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

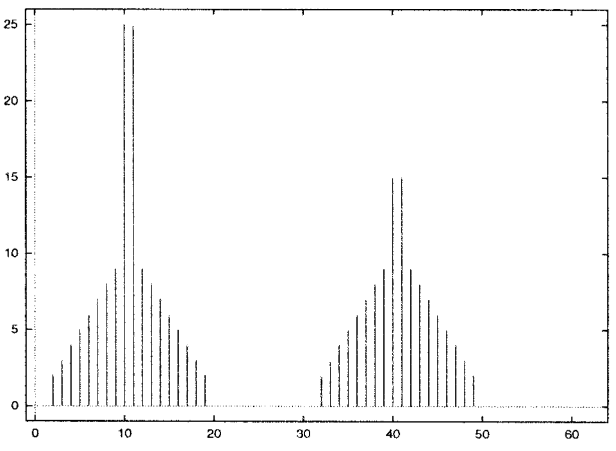

Figure 2. Graph of simple multimodal fitness landscape versus genotype. For a 6-bit binary string there are 63 possible genotypes (x axis). The y axis represents the fitness of each genotype.

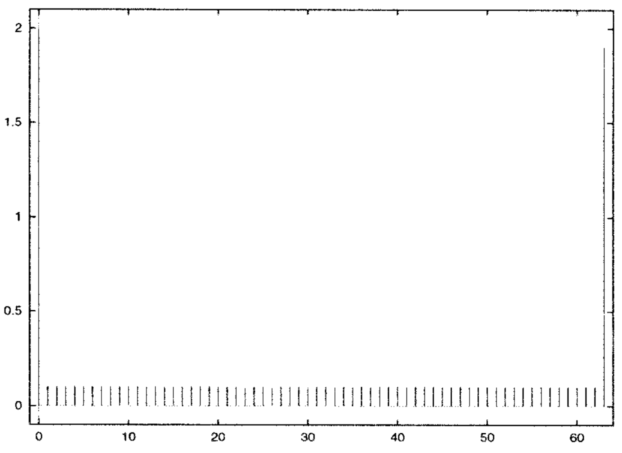

Figure 3. Graph of fitness versus genotype for a "deceptive" landscape. It is deceptive in that optimum strings (0 and 63) when crossed yield strings of very low fitness. The axes are the same as for Figure 2.

$000000 = 0$ and $111111 = 63$ produces very unfit strings. The same tests were carried out as with the previous landscape.

The third landscape chosen included time dependence. The initial landscape is that shown in Figure 2, which has the global optimum 10 and 11. However, after 60% of the population reach the global optimum the landscape is suddenly changed to that of Figure 4 wherein the original global optimum is now only a local optimum and a new, very narrow global optimum appears at 62. We call this a "jumper" landscape. In tests with the jumper landscape we used tournament selection of size 5 and also imposed a lower bound of 0.005 for mutation.

To compare fixed parameter results with self-adaptive ones we did 50 runs for each pair of fixed values of p_c and p. The crossover probability p_c was varied between 0 and 1 in increments of 0.1 while p was varied between 0 and 0.1 in increments of 0.01. Results were also obtained for $p = 0.003$. Thus, for each landscape 121 points

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

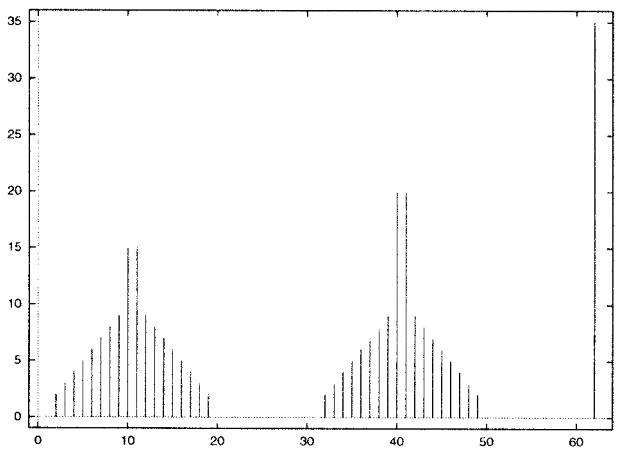

Figure 4. Graph of fitness versus genotype after "jump" from Figure 2. After 60% of the population reach the global optimum of the landscape in Figure 2 the landscape is changed to the above. The axes are the same as for Figure 2.

in parameter space were tested. The performances of the fixed parameter GAs were then compared with those of a self-adapting GA. In the latter we considered both 3-bit and 8-bit codifications. We concentrated mainly on online performance in these simple landscapes.

The above landscapes were all chosen to illustrate clearly various advantages of parameter codification in a context where the optima are all explicitly known. As an example of a system where the optima are not a priori known and where the size of the state space is very large we considered a 33-city traveling salesman problem (TSP). The TSP is a prototypical NP-complete problem that is easy to state but difficult to solve. Here our aim was to compare the performance of a GA with fixed probabilites for the genetic operators with the same GA but with coded probabilities, not to find the best codification and set of genetic operators for applying GAs to the TSP. It is known that GAs are quite competitive with other optimization techniques [11] when coded appropriately.

With that in mind we chose the most simple-minded codification via a path representation wherein the cities are listed in the order they are visited. The genetic operators used were "mutation" (permutation of two randomly selected cities) and "crossover" (inversion of the cities between two randomly selected points on the chromosome). For example, for a six-city problem mutation at 0 and 3 of the possible route 134520 leads to **5**34**1**20. Inversion between points 1 and 5 of the same route leads to 13**254**0. We did not bother to avoid cyclic permutations of a given route as once again our principal aim was to compare the performance of a given GA with and without parameter codification. Once again to compare fixed parameter results with self-adaptive ones we did various runs varying the "mutation" and "crossover" rates between 0 and 1 in increments of 0.1, thus resulting in a test of 100 points in parameter space. The operator probabilities were coded using 8 bits for each one resulting in a 49-bit chromosome. Although two-city interchange and inversion were used on the 33 bits that code for the route, standard mutation and crossover were used on the remaining 16 bits. For mutations we set a lower bound of 0.001 rather than zero. As both online and offline performance are of relevance in optimization problems such as the TSP, we compared fixed parameter and codified parameter GAs with respect to both.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

The fitness function for the n-city TSP, as is well known, is just the total distance of a route

$$R = \sum_{i=0}^{n-1} (d(c_i, c_{i+1})) + d(c_0, c_{n-1}) \tag{15}$$

with $d(x, y)$ the distance between cities x and y, subject to the restrictions that all cities must be visited and no city can be visited more than once.

Finally, we considered the above problem in a time-dependent context where starting with a TSP of 23 cities at generation 3,000, 10 more cities were added so that a new optimum route completely different to the first had to be found. To compare the different performances of the GAs in the TSP we used the following function

$$\mathcal{F} = \sum_{i=j}^{j+100} \frac{R_{\min}(i) + R_{\mathrm{av}}(i)}{R_{\min}(i) - R_{\mathrm{av}}(i)} \tag{16}$$

where the sum is over an interval of 100 generations and j represents the discrete time at which we are evaluating $\mathcal{F}$. Thus over a period of 6,000 generations we will have a set of 60 points to evaluate $\mathcal{F}$. $R_{\min}(i)$ is the shortest route length found at generation i and $R_{\mathrm{av}}(i)$ is the average route length for generation i. Note that this function emphasizes more the notion of population fitness than the best individual fitness, that is, it is a truer measure of online than offline performance. In terms of pure combinatoric optimization knowing that there exists a unique optimum state, it is of course finding the fittest individual in the shortest amount of time that is of interest. However, in many other problems, for instance in evolution and in other problems with a time-dependent environment, what is important is fitness of the population. The above fitness function takes into account both convergence speed and average population fitness. It also emphasizes both online and offline fitness and favors GAs wherein both types of fitness converge in value.

5 Results

In comparing with the performance of a standard GA we varied the mutation and crossover probabilities as mentioned in the previous section. In the case of the simple landscapes of Figures 1–3 in terms of offline performance high mutation rates had some preference relative to low ones; however, the consequent online performance was very low. We chose to emphasize more online performance without prejudicing too much offline performance and found that, averaging over many trials, mutation and crossover probabilities of 0.01 and 0.8 gave the best results, these also being the generally recommended values in the literature [5, 8, 15]. It is with these "optimized" values that we compare the self-adapting GA in the following. In comparing the fixed parameter and coded parameter behavior for the landscapes of Figures 2–4 we averaged the results of 30 different runs.

In Figure 5 we see the results of various GAs in the landscape of Figure 2. The plot shows in the upper half relative frequency of the optimum string as a function of time and in the lower half the average values of the codified GA parameters as a function of time. The initial population was chosen at random. Note that the population size here is large compared to the size of the state space as our intention with this landscape was to investigate the effects of parameter codification without strong finite size effects

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

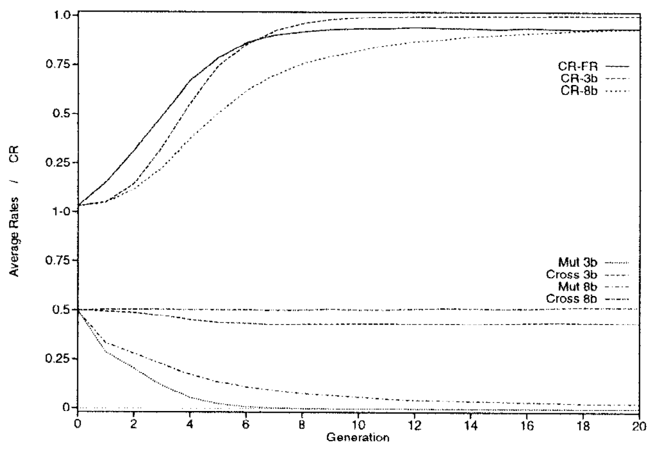

Figure 5. Results of fixed parameter and self-adapting GAs in the landscape of Figure 2. The upper graph shows the relative concentration (CR) of optimal strings as a function of time. CR-FR is the curve for a fixed rate GA, CR-3b for coded 3-bit probabilities, and CR-8b for coded 8-bit probabilities. In the lower graph one sees the average mutation and crossover possibilities as a function of time for the self-adapting GA. Mut 3b, Mut 8b, Cross 3b, and Cross 8b are the average mutation and crossover probabilities in 3-bit and 8-bit representations. The initial population was chosen randomly.

complicating the picture due to sampling errors. We will consider the latter thoroughly in a future publication.

The most notable feature of Figure 5 is the behavior of the coded parameters. For mutations we see that the system begins to “cool” itself down as the population begins to order. It is clear that there is no direct selective benefit in a given generation for one mutation rate versus another. However, we do see quite clearly the effects of symmetry breaking in that strings with low mutation rates are likely to have fitter offspring. This symmetry breaking becomes more pronounced as a function of time. With respect to crossover we see that there is rather a tendency to “genetic drift” as a function of time. That is, there does not seem to be a selective benefit for one crossover rate versus another. Comparing with the optimal fixed parameter GA we see that an 8-bit encoded GA spends more time trying to find the optimum mutation and crossover rates as it has to search through more possibilities. Note that the steady state population for a codified mutation rate will always be superior to that of a fixed rate simply because the population can only be strictly ordered when $p = 0$, that is, a codified GA always eventually leads to superior online performance.

In Figure 6 we show what happens with the landscape in the upper graph of Figure 2 but now in the case where the initial population is totally concentrated at one point, 49. One might ask why we would want to consider such a case. The reason why will become more obvious after seeing the results from the jumper landscape. Suffice it to say here that in a time-dependent landscape it can occur that the population has converged to an optimum but that at a certain point in time the landscape changes such that the original global optimum is now only a local optimum whereupon the system must seek the new global optimum. If the landscape changes only after the population has converged to the original optimum then in terms of evolution in the new landscape the system is starting from a very special initial condition.

Considering the explicit results we see that the optimal fixed parameter GA performs particularly badly. The reason why is simple: Crossover does not act very efficiently in encouraging diversification when one starts with an ordered population. This role

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

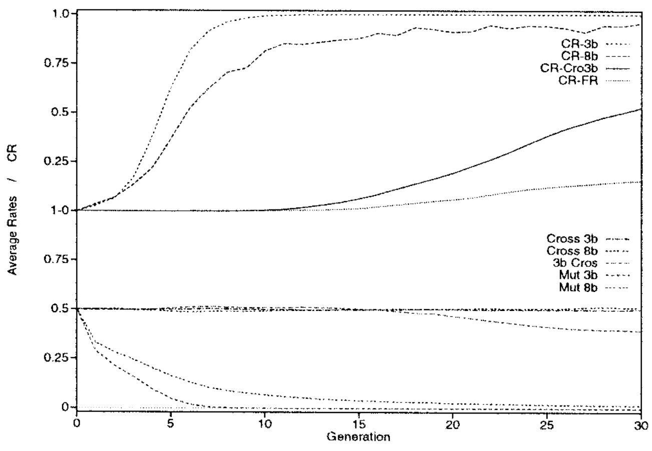

Figure 6. The same as in Figure 5, except now the initial population is centered at the point 49 in the landscape of Figure 2. CR-Cro 3b represents a GA where p is fixed at 0.01 and crossover is 3-bit coded. 3b Cros is the corresponding average crossover rate.

has to be played by mutation. If the mutation rate is low then the search time to find the optimum is large. It is clear that codifying crossover and not mutation does not help. When mutation is codified the results are clearly far superior. Once again we see how the system cools itself down after the GA begins to find the optimum. The behavior of the crossover probability as a function of time once again seems to exhibit a "genetic drift." Despite its inefficiency crossover is still a positive operator in that it aids the search for the new optimum. Only in the case of zero mutation rate and a totally ordered population is it strictly neutral.

The next figure, Figure 7, shows the results associated with the deceptive landscape of Figure 3. The initial population was biased with 100 individuals being placed at each optimum to make the problem even more "deceptive." In several runs the optimal fixed parameter GA was incapable of maintaining the population at the global optimum. The 8-bit coded GA by contrast increased the relative concentration of the optimum without any problems. The 3-bit coded GA sometimes converged to the global optimum and sometimes to the local one. An average over 30 runs led to the results seen. Once again we see that in terms of mutations the system cools itself down. The most interesting result here, however, is what happens to the crossover rate. In the results from the nondeceptive landscape the crossover probability drifted a small amount. Here, however, we see that initially there is a sharp drop in the rate. This is a direct response to the deception. Crossover of 0 strings with 63 strings produces very unfit results. As the system starts with a large number of them, those that are coded with low crossover rates will be preferred. Eventually as the system begins to order around the global optimum crossover loses its destructive nature and so the net rate increases.

Figure 8 contains the results for the deceptive landscape when $p = 0$. In this case we started with an initial population of 80 individuals, 20 of which were located at one optimum, 20 at the other and the remaining 40 distributed randomly between them. Once again, initially, due to the large populations associated with the two optima, crossover is very destructive, as any crossover that includes strings from both optima will result in very unfit offspring.

In Figure 9 we see what happens for the jumper landscape. Note that this figure shows the results of one run rather than an average over 30. The upper curves show

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

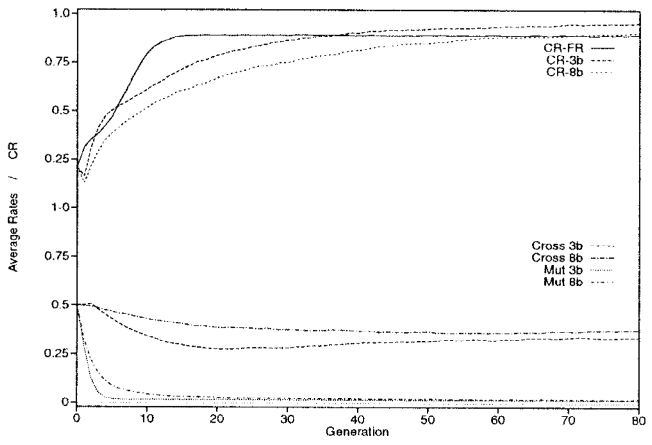

Figure 7. Same curves as in Figure 5 but now for the "deceptive" landscape of Figure 3. The initial population is chosen with a bias (see text) so as to emphasize the degree of deception.

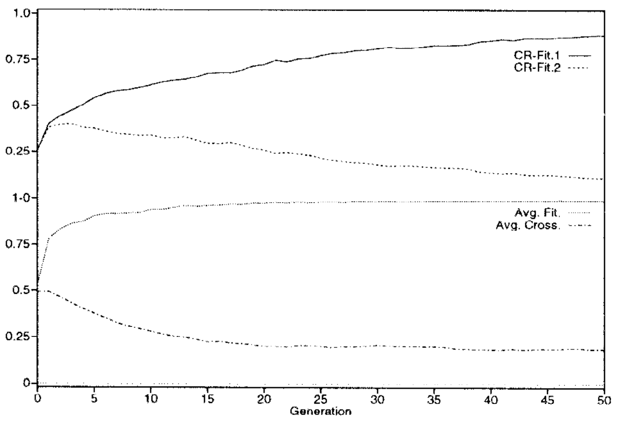

Figure 8. Graph of relative concentration (upper graph) and average crossover probabilities (lower graph) as a function of time for the landscape of Figure 3 with initial population size of 80 and $p = 0$. CR-Fit.1 is the relative concentration of the global optimum at point 0; CR-Fit.2 is the relative concentration of the local optimum at point 63; Avg. Cross. is the average crossover rate and Avg. Fit. the average population fitness.

the relative frequencies of the optima using 8-bit and 3-bit codification and also what happens when $p_c = 0$ and only the mutation rate is coded. There are several notable features: First of all, we see that the optimal fixed parameter GA was incapable of finding the new optimum, whereas the coded GAs had no problem whatsoever. For the case $p_c = 0$ the curve 40, 41 shows the relative frequency of the strings associated with the optimum at 40 and 41. Before the landscape "jump" this optimum is local, being less fit than the global optimum at 10 and 11. After the "jump" it is fitter, but less fit than the new global optimum 63, which is an isolated point.

One thus sees that the optimum was found in a two-step process after the landscape change. First, the strings started finding the optima 40, 41 before moving onto the true global optimum, 63. Immediately after the jump the effective population of the new global optimum is essentially zero. The number of strings associated with 40 and 41 first starts to grow substantially at the expense of 10 and 11 strings. At its maximum the number of optimum strings is still very low; however, very soon thereafter the

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

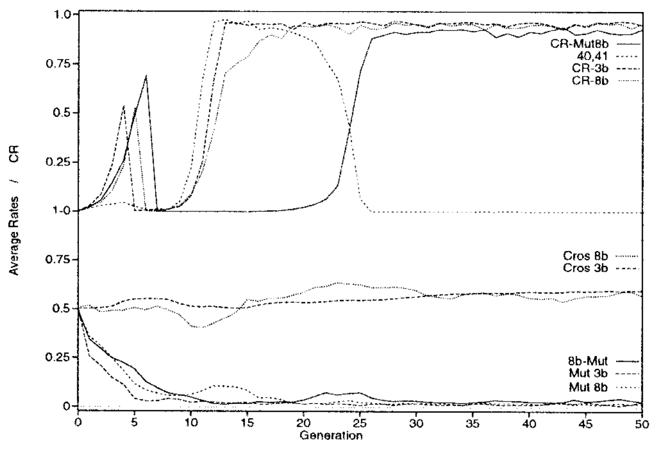

Figure 9. Graph of relative concentration of the optimum (CR) (upper graph) and average crossover and mutation probabilities (lower graph) as a function of time for the "jumper" landscape that changes from the landscape of Figure 2 to that of Figure 4 after 60% of the population has accumulated at the optimum of Figure 2. CR-3b and CR-8b are the results for 3-bit and 8-bit encoded algorithms. CR-Mut8b is the result for coded mutation with $p_c = 0$, with 40, 41 being the relative concentration of strings associated with the local optimum at 40 and 41. Mut 3b, Mut 8b, Cros 3b, and Cros 8b are the average mutation and crossover probabilities in 3-bit and 8-bit representations. The solid line for 8b-Mut is the average mutation rate in the case $p_c = 0$.

algorithm manages to find the optimum string, which then increases very rapidly at the expense of the rest. The striking result here can be seen by comparing the changes in the relative frequencies with the changes in the average mutation rate, especially in the case $p_c = 0$. Clearly they are highly correlated. First, while the population is ordering itself around the original optimum, there is an effective selection against high mutation rates as one can see by the steady decay of the average mutation rate. After the jump there is a noticeable increase in the mutation probability in the case of the 3-bit codification as the system now has to try to find fitter strings. As the global optimum is an isolated state it is much easier to find fit strings associated with 41 and 40. The population is now concentrated on this local optimum and starts to cool down again only to find that this is not the global optimum, whereupon the system heats up again to aid the removal of the population to the true global optimum. This latter reheating is most noticeable in the case where $p_c = 0$. It is clear that there is a small delay between the population changes and changes in the mutation rate. This is only to be expected given that there is no direct selective advantage in a given generation for a particular mutation rate. The selective advantage of a more mutable genotype over a less mutable one can only come about via a feedback mechanism. It is precisely this feedback process that is described and measured by the effective fitness function.

The average mutation rate also grows due to another effect, which is that the new optimum is more likely to be reached by strings with high mutation rates that then grow strongly due to their selective advantage. Thus high p strings will naturally dominate the early evolution of the global optimum. After finding the optimum, however, it will become disadvantageous to have a high mutation rate; hence low mutation strings will begin to dominate.

We now turn to the case of the TSP. In this case we tested various fixed rates, finding that the optimum compromise between offline and online performance was found for $p = 0.1$, $p_c = 0.2$. For high values we found that both offline and online performance were severely degraded. It is with the optimum fixed parameter results that we compared the self-adapting GA. In Figure 10 we see a comparison. One can

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

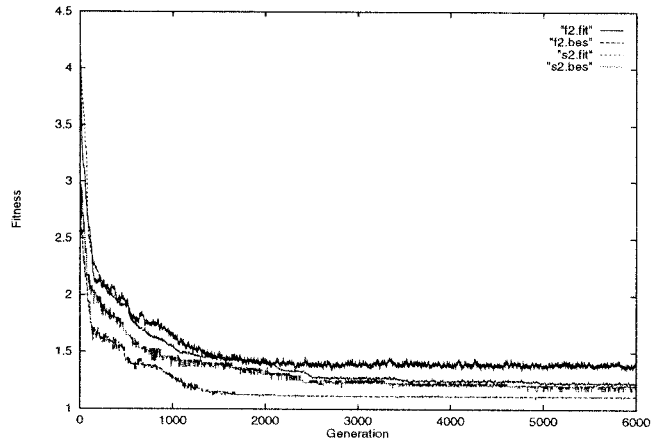

Figure 10. Graph of fitness (route length) versus time for 33-city TSP. The average and best fitnesses for a fixed parameter GA are represented by f2.fit and f2.bes while s2.fit and s2.bes are for a coded 8-bit GA.

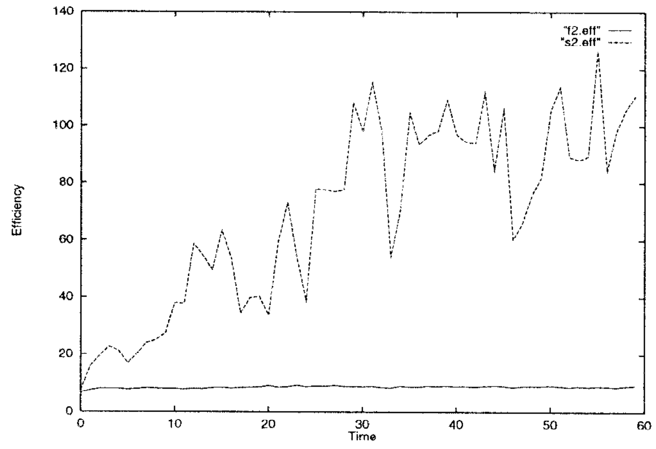

Figure 11. Graph of efficiency versus time for fixed parameter and coded parameter GAs in the 33-city TSP. Efficiency is represented by f2.eff for the fixed GA and s2.eff for the coded GA.

see that in terms of convergence velocity the two are roughly comparable; however, in terms of the algorithm fitness function, (16), the coded algorithm performs much better due to the fact that the average member of the population is almost as good as the best. This convergence between online and offline performance is an important feature of self-adapting GAs, implying that one does not have to sacrifice online performance to achieve a good offline performance. The relative performances are shown in Figure 11. The shortest route for this particular problem reported in [12] is 10,930. The best result from our GA is 11,040 for fixed parameters and 11,662 for coded parameters. We found that inversion affects the GA's performance more than mutation due to its more global scope.

In Figure 12 we see the results for the jumper TSP landscape and in Figure 13 the efficiencies of the two GAs are compared. Finally in Figure 14 we see the evolution of the coded crossover rate in the nonjumper and jumper landscapes. Note that very soon after the landscape change at t =3,000 the crossover rate begins to rise, reaching

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

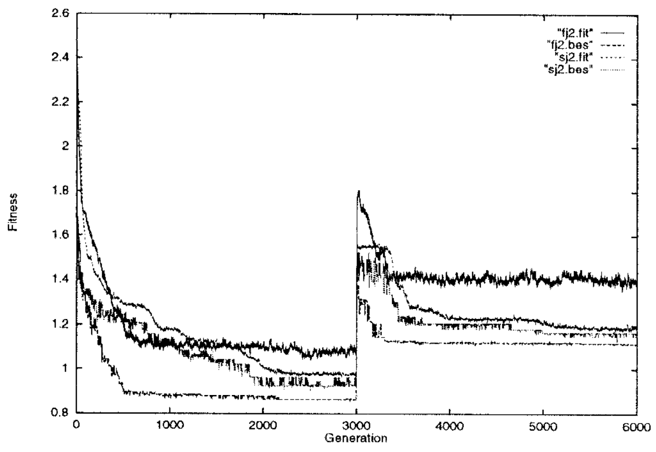

Figure 12. Graph of fitness (route length) versus time for "jumper" TSP wherein after 3,000 generations 10 more cities are added to the initial number of 23. The average and best fitnesses for a fixed parameter GA are represented by f2.fit and f2.bes while s2.fit and s2.bes are for a coded 8-bit GA.

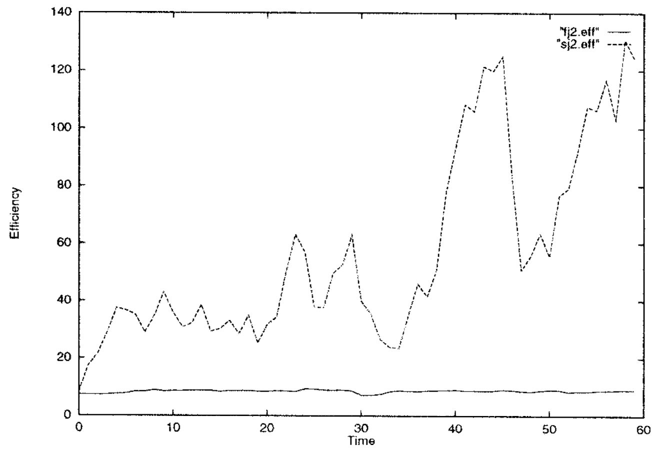

Figure 13. Graph of efficiency versus time for fixed parameter and coded parameter GAs in the "jumper" TSP. Efficiency is represented by fj2.eff for the fixed and sj2.eff for the coded GA.

a peak at t =3,500, where it is almost double its prejump value. Thus as in the case of the previous jumper landscape, one sees that the parameters respond to changes in the landscape. In this case it is the inversion rate that rises after the landscape change to aid the algorithm in the search for the new optimum.

6 Discussion and Conclusions

In this article we have investigated the effects of coding the probabilities that govern the action of genetic operators in a simple GA based on a select set of model fitness landscapes. Two basic aims of the article were to investigate how parameter codification may help in adaptive search problems and to illuminate some of the consequent theoretical issues.

Optimizing the exogenous parameters of a GA is a difficult task for various reasons.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

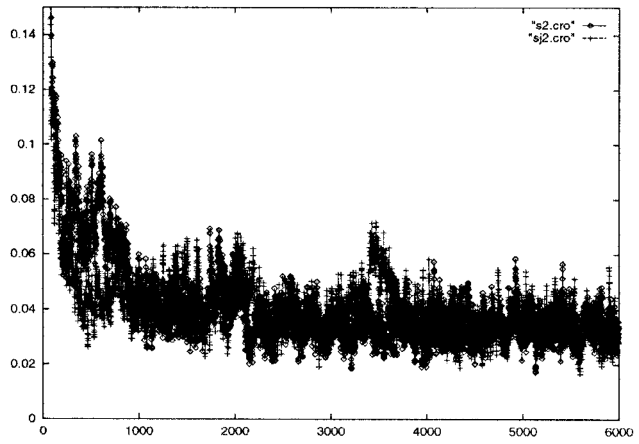

Figure 14. Graph of crossover rate versus time for the 33-city TSP and the "jumper" TSP. The "jumper" landscape is represented by sj2.cro and s2.cro is for the plain 33-city TSP.

First, the optimum values are both landscape and time dependent. Second, one must either use a meta-GA to do the optimization or run the GA for various combinations of mutation and crossover probabilities until sufficient statistics are obtained to allow one to determine what are the optimum values. We have therefore tried to show what one might gain via a codification of these parameters within the chromosomes themselves, thus leading to an essentially totally autonomous system. We saw that in a time-dependent landscape certain parameter values will be favored over others to facilitate the search for new optima, and that the values themselves will change over time. We saw as well that self-adapting GAs offer a very good compromise between offline and online performance. In normal fixed parameter GAs one almost always has to sacrifice one to optimize the other. However, the chief lessons we wish to point out from our work here have much wider implications than those to be drawn from a simple comparison of coded versus noncoded GA performance.

What are these lessons? There are two: the lesson of induced symmetry breaking, and the lesson of effective fitness. In the former, if we have a degenerate genotype-phenotype mapping, that is, there exist synonymous genotypes, then there exists a symmetry, or perhaps better to say an equivalence relation on G. By degenerate we mean explicitly that the different genotypes all correspond to exactly the same phenotypic fitness value. The fitness landscape as a function of genotype thus has neutral directions. In the presence of pure selection the system will be unable to distinguish between these directions. However, on including mutation and/or crossover this degeneracy will be lifted, certain directions now being preferred over others even though there is no direct selective advantage for them. Thus we may say there is an induced symmetry breaking. The toy model of Section 3 and the later numerical results all confirm this clearly and explicitly. In understanding this phenomenon the normal concept of fitness is of little use, which brings us to the second important lesson, that of effective fitness. Even though certain directions in G may be neutral in terms of fitness they most definitely are not in terms of the effective fitness. We therefore claim that effective fitness, and the consequent fitness landscape in terms of it, are much more relevant and meaningful concepts in the presence of other genetic operators such as mutation and crossover. The effective landscape will be time dependent even though the original landscape was not. We will give a more thorough analysis of the effective fitness in a future publication.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

Also of interest is the relation to the theory of natural evolving systems. There the bare mutation rates have a biophysical origin that cannot be coded directly; however, it is also true that different synonymous codons can have varying mutabilities. The most obvious reason for this is that point mutation itself does not respect the equivalence between synonyms. For example, in the case of the six leucine codons, *CTPu* have four possible silent point mutations, *CTPy* have three, and *TTPu* have two. Therefore, if missense mutations are selected negatively then one will find that *CTPu* codons have an effective selective advantage due to their higher resistance to mutation. Vice versa, in the recognition regions of the enveloping protein of a lentivirus there is a preference for codons that mutate nonsynonymously, to escape detection by the immune system [20]. One can view this effect as an example of self-adaption where the choice of mutation rate is realized indirectly through the codon bias.

Acknowledgments
This work was partially supported by a CONACYT scholarship to J. Mora-Vargas and by DGAPA grant IN105197.

References
1. Bäck, T. (1992). Self-adaptation in genetic algorithms. In F. J. Varela & P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life* (pp. 263–271). Cambridge, MA: MIT Press.
2. Bäck, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford: Oxford University Press.
3. Bowen, D. (1986). A study of the effects of internally determined crossover and mutation rates on genetic algorithm optimization. Unpublished manuscript, University of Alabama, Tuscaloosa.
4. Caruana, R. A., & Schaffer, J. D. (1988). Representation and hidden bias: Gray versus binary coding for genetic algorithms. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 153–161). San Mateo, CA: Morgan Kaufmann.
5. De Jong, K. (1975). *An analysis of the behaviour of a class of genetic adaptive systems*. Unpublished doctoral dissertation, University of Michigan.
6. Fogel, D. B. (1991). *System identification through simulated evolution: A machine learning approach to modeling*. Needham, MA: Ginn Press.
7. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addision-Wesley.
8. Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, SMC-16* (1), 122–128.
9. Hart, W. E., & Belew, R. K. (1991). Optimizing an arbitrary function is hard for the genetic algorithm. In R. K. Belew & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms and Their Applications* (pp. 190–195). San Mateo, CA: Morgan Kaufmann.
10. Holland, J. H. (1992). Genetic algorithms. *Scientific American*, July, 66–72.
11. Homaifar, A., Guan, S., & Liepins, G. E. (1993). A new approach on the travelling salesman problem by genetic algorithms. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms and Their Applications* (pp. 460–466). San Mateo, CA: Morgan Kaufmann.
12. Karg, R. L., & Thompson, G. L. (1964). A heuristic approach to solving travelling salesman problems. *Management Science, 10*, 225–248.
13. Kaufmann, S. A. (1993). *The origins of order*. Oxford: Oxford University Press.
14. Kimura, M. (1983). *The neutral theory of molecular evolution*. Cambridge, U.K.: Cambridge University Press.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021

15. Schaffer, J. D., Carauna, R. A., Eshelman, L. J., & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications* (pp. 51–60). San Mateo, CA: Morgan Kaufmann.

16. Schaffer, J. D., & Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications* (pp. 36–40). Hillsdale, NJ: Erlbaum.

17. Schwefel, H. P. (1981). *Numerical optimization of computer models*. New York: Wiley.

18. Stephens, C. R., & Waelbroeck, H. (1997). Effective degrees of freedom in genetic algorithms and the block hypothesis. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms and Their Applications* (pp. 34–41). San Mateo, CA: Morgan Kaufmann.

19. Stephens, C. R., & Waelbroeck, H. (1998). Analysis of the effective degrees of freedom in genetic algorithms. *Physical Review E57*, 3251–3264.

20. Waelbroeck, H., & Stephens, C. R. (in press). Codon bias and mutability in HIV sequences. *Journal of Molecular Evolution*, National University of Mexico Preprint ICN-UNAM-97-09 (adap-org/9707).

21. Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In G. Rawlins (Ed.), *Foundations of genetic algorithms 1* (pp. 205–221). San Mateo, CA: Morgan Kaufmann.

Downloaded from http://direct.mit.edu/artl/article-pdf/4/2/183/1661604/106454698568512.pdf by guest on 21 September 2021