Control and Cybernetics

vol. 26 (1997) No. 3

Schemata and deception in binary genetic algorithms: a tutorial

by

G. Venturini* S. Rochet** M. Slimane*

* Laboratoire d'Informatique, E3i, Universite de Tours,
64, Avenue Jean Portalis, 37913 Tours, France,
e-mail: venturini,slimane@univ-tours.fr

** Neurinfo and Universite d'Aix Marseille, Europarc Bat C,
Technopole de Chateau Gombert, 13453 Marseille, France,
e-mail: rochet@gyptis.univ-mrs.fr

Abstract: In this paper, we present a survey of the theory concerning the canonical genetic algorithm (CGA). We first describe this algorithm and show that several questions should be answered about it in order to understand precisely its behavior: how does the CGA explore its search space, how does it converge, how difficult is the function it optimizes? We present the answers which are provided by the CGA theory. For the first question, this consists in the classical schema analysis, with the fundamental theorem, the implicit parallelism and the k-armed bandit analogy. For the second question, we describe several convergence theorems. Finally, to answer the last question, we review the work on GA-easy and GA-hard functions, which includes general theorems, deception theory, measures of function difficulty like epistasis measures, and we relate these definitions to other search methods.

Keywords: canonical genetic algoritm, schema analysis, convergence, deception, GA-easy, GA-hard

1. Introduction

1.1. The optimization problem

Genetic algorithms (GAs) (Holland, 1975, De Jong, 1975; 1988) are now much more popular than a few years ago, and especially so from the applications point of view. In many domains, GAs represent an important alternative to gradient or heuristic search, for instance, and positive results obtained in practice are mainly responsible for their success. However, theoretical analysis of GAs is the fundamental point to be explored if one wants to analyze, explain and forecast the behavior of GAs. Thus, this paper does not emphasize at all the practical applications of GAs, but rather aims at giving some insight about the theoretical analysis of GAs.

We will assume that the problem P to be solved has been formalized as an optimization problem. This optimization problem consists in finding in a search space S a point s^* that maximizes a function f. We will assume that the solution space S is a binary search space, i.e. $S = \{0, 1\}^l$, where l is a constant denoting the number of bits used to encode the solutions to problem P.

Many methods can be used for solving this optimization problem. Some of them can be called "direct" methods and compute s^* provided that useful information is available about f. For instance, if it is known that $f(s) = -s^2 + 2s - 1$, then it is possible to compute s^* directly. When less information is available about f, such direct methods cannot be applied. In this case, the optimization methods must enumerate points in the search space S. Such enumeration can be deterministic or probabilistic. Deterministic enumerative methods always enumerate the same points in S when the initial point is the same. This is the case for instance of a standard gradient search. Probabilistic methods may not follow the same path in S for a given initial point. GAs belong to this second category of methods.

1.2. The canonical GA

A simple genetic algorithm, referred to as the canonical GA (CGA) in the following, considers that a binary string $s \in S$ is the genotype of an individual. One bit at a given location in s can be viewed as a gene. The CGA processes a population of such individuals and applies to them the search operators called genetic operators that modify the genes. The function f is viewed as the fitness function of an individual and represents its ability to survive in the environment. The CGA simulates the natural selection and gives an opportunity to the fittest individuals to reproduce in order to possibly improve the fitness of the generated offspring. The CGA uses the following principles:

1. Generate randomly and Evaluate an initial population $P(0)$ of n individuals,
   $t \leftarrow 0$,
2. Generate $P(t+1)$:
   (a) Select n individuals from $P(t)$ to form a population $P_s(t)$ with the following probability distribution: $\forall s \in P(t)$, $P_{select}(s) = \frac{f(s)}{\sum_{s' \in P(t)} f(s')}$,
   (b) Recombine individuals of $P_s(t)$ into a population $P_r(t)$ with the one point crossover operator. Pairs of individuals are considered, and the crossover is applied to one pair with a probability P_{cross}.
   (c) Mutate individuals of $P_r(t)$ to form $P(t+1)$ by modifying bits in the individuals with a probability P_{mut},

3. Evaluate the individuals in $P(t+1)$,
4. $t \leftarrow t+1$,
5. Go to 2 or Stop.

This algorithm is a parallel one, even if you implement it on a sequential computer. If you consider that the population is a vector of n variables, then the variables at generation $t+1$ evolve in parallel and according to the variables at generation t. Truly sequential versions of the CGA exist where only one variable changes at each generation. These are the so-called steady state GAs such as the Genitor (Whitley, 1989). Parallel models have also been designed for coarse or fine grained parallelization (Whitley and Starkweather, 1990, Spiessens and Manderick, 1991, Muhlenbein, 1991). GAs may also use various representations and operators. However, most of the theoretical studies concern the CGA, so this paper concentrates on this simple algorithm.

The initial population $P(0)$ is generated randomly by choosing n points in S with a uniform distribution (step 1 in the previous algorithm). Using $P(0)$ and genetic operators, the CGA will generate the population $P(1)$. More generally, the CGA generates population $P(t+1)$ from $P(t)$: the search for new solutions is guided by the previously explored solutions.

The first intermediary population $P_s(t)$ is selected (step 2a) by performing n sampling with replacement from $P(t)$ with the probability distribution mentioned previously: an individual s is selected with probability $P_{select}(s) = \frac{f(s)}{\sum_{s' \in P(t)} f(s')}$.

The second intermediary population $P_r(t)$ is computed by recombining pairs of individuals of $P_s(t)$ with the crossover operator (step 2b). One way to select which individuals of $P_s(t)$ will recombine, is to scan the population $P_s(t)$ and select an individual for recombination with a probability P_{cross}. Selected individuals can be considered by pairs in the order of their selection, for instance. For a given pair of individuals, the one point crossover operator selects randomly and uniformly a cutting point between 2 and l, and exchanges between the two individuals the two sub-strings delimited by the cutting point. The offspring generated this way will replace their parents in the population, transforming $P_s(t)$ into $P_r(t)$.

The final population $P(t+1)$ is computed by applying the mutation operator to the population $P_r(t)$ (step 2c). This operator considers the whole population of binary strings as a single string, and modifies every bit of this string with a probability P_{mut}.

Several stopping criteria can be used (step 5). For instance, the CGA may stop when the quality of the best individual is above a given threshold, or when a given number of generations have been performed, or when the search does not improve any more.

In this paper, we will consider an example of a simple function, called the Onemax function (Ackley, 1987; Syswerda, 1989). This function simply sums the bits in a given binary string. For instance with $l = 6$ bits, f (000000) equals

0, !(000011) equals 2 and !(111111) equals 6. This function has only one optimum which in this example is the string $s^* = 111111$.

1.3. Fundamental questions about the CGA

Several fundamental questions are raised about the CGA, but the CGA algorithmic description does not provide an explicit and direct answer to them. For instance, one may ask:

- How does the CGA explore S? The CGA uses its population of points to guide the search. This strategy works in a more global way than other enumerative methods like simulated annealing, for instance. The earlier work on the CGA has concerned the characterization of this strategy (Holland, 1975; Goldberg, 1989a) by using schema analysis as described in Section 2.

- What about convergence? Another point that one should consider about search methods is the stopping criterion, and more precisely, under which conditions does the method stops, and what kind of solution s you get when it stops. For instance, a gradient search usually stops when the derivatives of f equal 0. When f is unimodal, the output solution is s^*. GAs do not guarantee in practice that you will get the optimum. Section 3 presents several convergence theorems about GAs.

- How to characterize the difficulty of f? When comparing enumerative search methods, one should consider the assumptions about f under which one method is likely to work well. Such assumptions are known for instance for a gradient search. One could be tempted to say that GAs make less assumptions about f than other methods. It is true that GAs can deal for instance with noisy, multimodal (De Jong, 1975; Goldberg and Richardson, 1987; Mahfoud, 1995), or time dependent (Cobb and Grcfenstette, 1993) functions. However, it is also true that GAs do make assumptions about f, but these assumptions are not yet well understood, and arc different from the traditional ones. This is precisely one aim of the GA theory of deceptiveness. An introduction to this part of theory is presented in Section 4.

The remaining of this paper is organized as follows. Section 2 desr;ribes the basic schema analysis, which includes the schema theorem, the implicit parallelism, and the two-armed bandit analogy. Section 3 gives some examP.les of convergence theorems about GAs. Section 4 describes how the difficulty of a problem can be characterized from the GA point of view. This concerns mainly the deceptive problems, GA-easy and GA-hard functions.

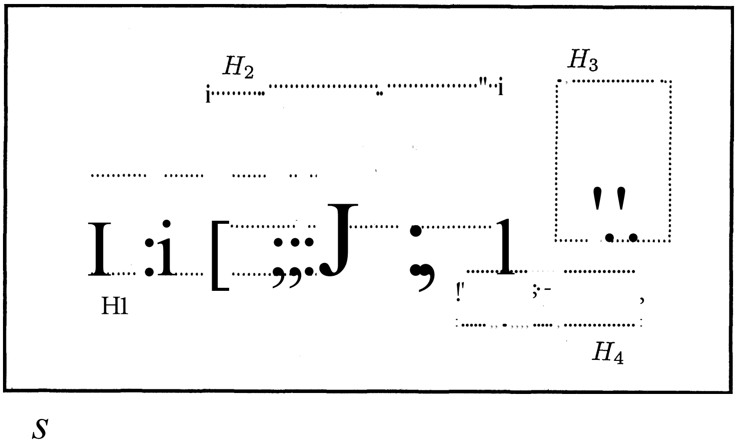

Figure 1. The CGA processes points in S (s_1 to s_5) but also regions of S (H_1 to H4).

2. Schema analysis

2.1. Basic concepts

2.1.1. Schema: a region and a building block

A schema H is a string of symbols of length l the same length as the binary strings in S, but over the alphabet of 3 symbols {O, 1, *}. "O" and "1" are used to represent fixed bits in the schema. "*" is a wildcard symbol that replaces either a "O" or a "1". For instance, with l = 6, the following schema:

H1 = 111 * **

represents the region in S that contains all strings that start with "111" and that have any other bits in the remaining positions. The following schema:

H2 = 1 * * * *O

represents the region in S where all strings start with 1 and end with 0.

A schema thus can be viewed as a region in S, or a set of strings, or a hyperplane of S or also a search direction in S. However, a schema does not only formalize the notion of region, it also formalizes the notion of a building block. For instance, the two schemata O* * * ** and * * * * *1 represent two blocks of one bit each. Such blocks are useful to explain the CGA behavior. For instance, the CGA may combine these two blocks to produce a third schema o* * * * 1.

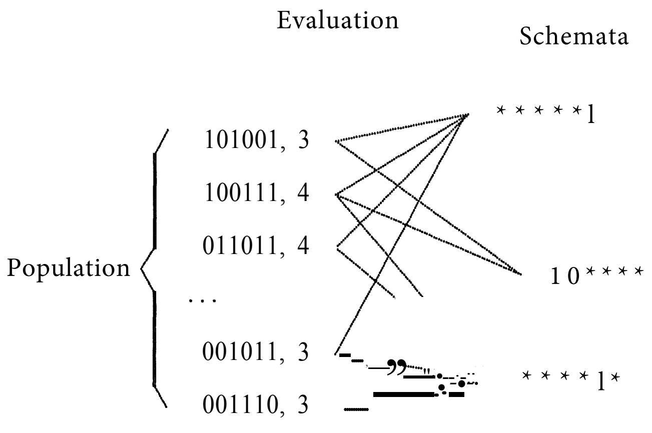

Figure 2. Strings in the population may belong to several schemata. One schema may be represented by several strings in the population. We have represented the evaluation of binary strings that would be obtained with the Onemax function.

2.1.2. Order and defining length of a schema

The order of a schema *H*, denoted by *o(H)*, is the number of fixed bits in *H*, like for instance:

$$o(H1) = o(111***) = 3, \quad o(H2) = o(1****0) = 2$$

The number of strings in H equals r $2^{o(H)}$.

The defining length of a schema *H*, denoted by *8(H)*, is the difference between the position of the last and the first fixed bits in *H* like for instance:

$$\delta(H_1) = \delta(111 * **) = 3 - 1 = 2, \quad \delta(H_2) = \delta(1 * * * *0) = 6 - 1 = 5$$

2.1.3. Schemata and the population of binary strings

As explained before, by processing strings the CGA also processes schemata. In fact, binary strings can be related to the schemata they belong to, as shown in Fig. 2. Several strings may belong to the same schema. Also, several schemata may have one or several strings in common.

For a given population of strings, one can define the set of all schemata that contain at least one string of the population. When the population is very diverse, this set may contain many schemata. However, as will be shown in the section 2.4 on implicit parallelism, not all such defined schemata are usefully processed by the CGA, mainly because the sampling of strings is generally not uniform and not large enough.

2.1.4. Static and dynamic schema fitness

In a general way, the fitness of a schema H is the mean fitness of binary strings that belong to H. More precisely, one may consider either the set of all strings that belong to H irrespective of the strings that are in the population, or only the strings in the population that belong to H. In the first case, the fitness of H does not depend on the population, but only on the fitness landscape of f and is therefore called the static fitness. It is constant for a given fitness function f and equals:

$$f(H) = \frac{1}{|H|} \sum_{s \in H} f(s)$$

In the second case, the fitness of a schema H depends on the fitness landscape but also on the current population of binary strings, and is therefore called the dynamic or observed fitness. It may change over time depending on how the CGA samples strings in S and equals:

$$f(H, t) = \frac{1}{|H \cap P(t)|} \sum_{s \in H \cap P(t)} f(s)$$

Static fitness will be used in the section on deception. It has the advantage of facilitating the analysis of the CGA, but it does not consider its dynamic behavior. Dynamic fitness will be used in the fundamental theorem in the next section but is much more difficult to analyze than static fitness.

2.2. The fundamental theorem

2.2.1. Notations

The discussion of this theorem is detailed in Goldberg (1989a), and is performed in three steps, where each step takes into account the selection alone, the selection and the crossover operator, and finally the selection, crossover and mutation simultaneously. We present here this discussion. The following notations are used:

- $m(H, t) = |H \cap P(t)|$ denotes the number of strings in the population $P(t)$ at generation t that belong to schema H. In the following, these strings will be said to "represent" H.
- $f(t) = \frac{1}{n} \sum_{s \in P(t)} f(s)$, denotes the mean fitness of strings in the population at generation t.

For a given schema H, let us try to express $E(m(H, t+1))$, the average number of strings that represent H in the next generation, as a function of $m(H, t)$.

2.2.2. Selection only

Firstly, let us consider the CGA without crossover or mutation. In this simplified algorithm, only the selection operator is used. It selects n strings in $P(t)$

with the probabilities $\frac{f(s)}{\sum_{s' \in P(t)} f(s')}$ described previously. Let us suppose that $m(H, t) = k$ and let $s_1, \ldots, s_i, \ldots, s_k$ denote all the strings in $P(t)$ that belong to H. After selection which selects randomly n strings in P(t), each string s_i will be represented on the average n_i times in $P(t+1)$ with:

$$n_i = n \frac{f(s_i)}{\sum_{s' \in P(t)} f(s')}$$

In $P(t+1)$, there will thus be $E(m(H, t+1)) = n_1 + \ldots + n_i + \ldots + n_k$ strings that belong to H. This gives:

$$E(m(H, t+1)) = n \frac{f(s_1) + \ldots + f(s_k)}{\sum_{s' \in P(t)} f(s')}$$

The above equation can be rewritten using the two following equalities:

$$\frac{n}{\sum_{s' \in P(t)} f(s')} = \frac{1}{\bar{f}(t)}$$

and:

$$f(s_1) + \ldots + f(s_k) = k f(H, t) = m(H, t) f(H, t)$$

This results in the fundamental theorem limited to the selection only:

$$E(m(H, t+1)) = m(H, t) \frac{f(H, t)}{\bar{f}(t)}$$

This equation states that schemata which have a dynamic fitness above the average fitness of the strings in the population will be given on the average more strings in the next population. Furthermore, if one assumes that $\frac{f(H,t)}{\bar{f}(t)}$ is a constant denoted by c, then the increase or decrease of $m(H, t)$ is exponential (i.e. $m(H, t) = e^t$). This property will be important when describing the k-armed bandit problem. Of course, c is unlikely to be constant all the time.

2.2.3. Selection and crossover

Let us consider now the influence of crossover. Once selection has been performed, an intermediary population $P_S(t)$ is created. Each strings in $P_S(t)$ has a probability of p_{cross} to be involved in a crossover. Let us consider a given schema H and one string s in $P_S(t)$ that belongs to H. Let us suppose that s has been selected for crossover, and let s' denote the second string involved in the crossover. We remind the reader that the offspring generated by crossover replace their parents in the population.

The probability that s still belongs to H after crossover depends on the cutting point that has been chosen and on the two parents s and s'. When the cutting point is between two bits of H, then H is unlikely to survive in s and will be disrupted, unless the two parents belong to H. For instance, let us consider the schema $H = 1 * * * 0*$ and the two strings $s = 101000$ and $s' = 010011$. If the crossover between s and s' takes place between the two bits of H, i.e. after position 1, 2, 3 or 4, then H will not survive in s. If crossover takes place after position 5, then H will survive in s. Also, if s' has a "0" in position 5, then H will always survive in s.

The probability $P_{cut}(H)$ that the cutting point takes place between two bits of a schema H depends on the defining length $\delta(H)$ of H and is equal to:

$$P_{cut}(H) = \frac{\delta(H)}{l - 1}$$

i.e., the number of cutting points that disrupt H divided by the total number of cutting points on strings of length l. As mentioned in the previous example, the probability that H does not survive in s is, however, smaller than $P_{cut}(H)$ because the two parents may be equal, for instance, or may both belong to H. So the probability that H survives in s after crossover is thus:

$$P_{survive}(H) \geq 1 - P_{cross} P_{cut}(H)$$

which can be rewritten as:

$$P_{survive}(H) \geq 1 - P_{cross} \frac{\delta(H)}{l - 1}$$

This probability must be related now to the previous equality between $m(H, t)$ and $E(m(H, t + 1))$:

$$E(m(H, t + 1)) = m(H, t) \frac{f(H, t)}{\bar{f}(t)}$$

Taking into account the influence of crossover, this equality becomes:

$$E(m(H, t + 1)) = m(H, t) \frac{f(H, t)}{\bar{f}(t)} P_{survive}(H)$$

Using the previous minoration of $P_{survive}(H)$, this results in the fundamental theorem limited to selection and crossover:

$$E(m(H, t + 1)) \geq m(H, t) \frac{f(H, t)}{\bar{f}(t)} \left[1 - P_{cross} \frac{\delta(H)}{l - 1}\right]$$

Selection and crossover increase the number of representing strings of schemata with fitness above the average and whose defined bits are compact, i.e. close to each other in the binary string representation.

2.2.4. Selection, crossover and mutation

Let us consider now the effect of mutation. This operator modifies a bit with a probability P_{mut}. A string *s* will still belong to a schema H after mutation as soon as no mutation takes place in s on the defined bits of *H*. The number of defined bits of H has been defined previously as the order o(H) of H. The probability that no defined bits of H are altered by mutation is thus equal to:

$$(1 - P_{mut})^{o(H)}$$

For instance, for $H = 1 * * * 0*$ and $P_{mut} = 0.01$, this probability is equal to $(0.99)^2 \simeq 0.98$. This schema will survive a mutation with a higher probability than the schema $10 * * * 0*$ for instance.

If one considers that $P_{mut} << 1$, then this probability can be approximated by $1 - P_{mut}\,o(H)$. Taking into account crossover and mutation, the probability that H survive application of these two operators is such that:

$$P_{survive}(H) \geq \underbrace{(1 - P_{cross} \cdots)}_{\text{crossover influence}}\underbrace{(\qquad\qquad)}_{\text{mutation influence}}$$

If one assumes that the cross product, i.e. $P_{cross} \cdot [z] \; P_{mut}\,o(H)$, is small, then this expression can be simplified into:

$$P_{survive}(H) \geq 1 - P_{cross} \cdot l - P_{mut}\,o(H)$$

The fundamental theorem can now be rewritten in its final form:

$$E(m(H, t+1)) \geq m(H,t)\frac{f(H,t)}{f(t)}\left[1 - P_{cross}\frac{o(H)}{l-1} - P_{mut}\,o(H)\right]$$

This theorem states that a schema will be given an increasing number of representing strings in *P(t)* when:

- its fitness is above the average, and
- it is short, and
- it is compact,

2.2.5. Discussion of the fundamental theorem

This theorem gives some insight about how the CGA behaves but has, however, several limitations. For instance, it only describes the mean behavior of the CGA: standard deviations may have an important role to play, especially in the way the CGA concentrates in a given area of the search space. This point is discussed for instance in Radcliffe and Surry (1994) (see section 4.3.3 in this paper). Also, the fundamental theorem does not tell anything about reaching the optimum, neither in term of convergence time nor in term of quality of the best string obtained (see section 3). It does not tell which encoding would be the best one for solving the problem. This can be seen by considering that there exists Ml different binary encodings of a space of size M, that all verify the theorem, and it would be surprising that all give the same results in practice. Finally, the role of genetic operators is not really taken into account as it should: on one hand genetic operators of crossover and mutation are considered as bad perturbations because their effects must be bounded in order to prove the theorem, but on the other hand, the recombination of building blocks through crossover and the necessary pertubations of mutatio are claimed to be the basis of GAs success.

Thus, this theorem is more useful from the point of view of adaptive systems than from the point of view of optimization.

2.3. Implicit parallelism

2.3.1. Intuitive view

Now that we know which schemata are favored by the CGA, an important issue is to know how many schemata are processed by the CGA at the same time. This number of schemata can be approximated by n^3, where n is the number of strings in the population (Goldberg, 1989a). The number of schemata processed by the CGA is thus much higher than the number of strings themselves.

2.3.2. Principle

This demonstration concerning implicit parallelism is detailed in Goldberg (1989a). Let us denote by n_s the number of schemata or search directions processed by the CGA. To evaluate the value of n_s for a population of n strings with l bits, one may consider the set of all possible schemata. First, from that set one should remove the schemata that do not resist well crossover and mutation. So we only consider the schemata that have a probability to survive the genetic operators of at least P_s. According to the probabilities of survival computed in the previous section, such schemata must have a defining length strictly smaller than $l_s = (1 - P_s)(l - 1) + 1$.

The number of schemata of the length smaller than l_s that a string of length l belongs to is $2^{l_s - 2}(l - l_s + 1)$. As there are n strings in the population, the number of schemata usefully processed in a population of size n is:

$$n_s = n 2^{(l_s - 2)} (l - l_s + 1)$$

However, some low order schema have been counted several times because they are common to several strings. So finally, only the schemata with an order greater than $l_s/2$ are considered. In addition, one assumes that the population has a size $n = 2^{l_s/2}$ in order to count each schema once on the average. So the estimation of n_8 now becomes:

$$ns \; 2 \; \frac{n^3(1 - 1_s + 1)}{4}$$

which amounts to:

$$n_s = \Omega(n^3)$$

One should notice that this demonstration has been extended to population of size n $= 2^{k\,l}$ where k is a positive parameter (Bertoni and Dorigo, 1993). The authors are also able to compute a lower bound on the average number of schemata that will be present in a uniformly distributed population for all $k > 0$. This lower bound is optimum when k 2: 1.

2.3.3. Discussion on implicit parallelism

This optimistic result is tempered however by several authors (Baker and Grefenstette, 1989; Grefenstette, 1991). Such a parallelism is possible only when the population is large enough and spread uniformly over $\{O, 1\}^1$. Those conditions are met at the beginning of the search, just after the random initialization, but after a few generations, the population has already converged and contains less schemata. This is true for Golberg's discussion on implicit parallelism but also for Bertoni and Dorigo's paper. Goldberg has already identified this problem and proposes a solution he called "messy GAs" (Goldberg et al., 1991). These new algorithms exploit directly the recombination of building blocks but the cost is as of now prohibitive even though a fast version of this algorithm has been developed (Goldberg et al., 1993; Kargupta, 1995).

2.4. The k-armed bandit analogy

2.4.1. Intuitive view

The point studied here is about how the CGA explores or exploits schemata, a problem known as the exploration versus exploitation dilemma. Exploration consists in testing new regions in S and is necessary to get away from local optima, it is more likely to be unsuccessful. Exploitation consists in concentrating the search effort in the promising regions already visited. It is necessary to do this once the right region has been found. But if the region is not really the best one, then exploitation may lead the search to a local optimum. Both actions are necessary but they work in opposite directions. Finding the right balance between exploration and exploitation is to resolve the exploration versus exploitation dilemma.

This dilemma can be formalized by considering bandit machines traditionally used in decision theory. The arms of the machine represent areas, or more precisely schemata, as represented in Fig. 3. The CGA has to select whether it explores new schemata with unknown fitness, or whether it concentrates on

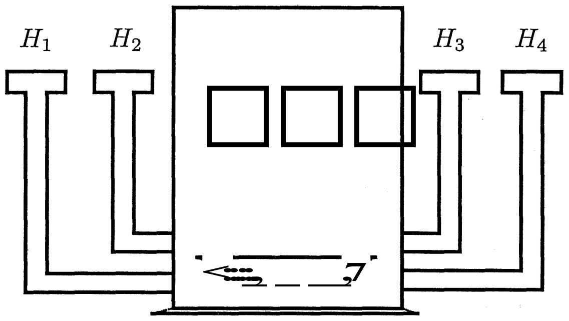

Figure 3. The k-armed bandit analogy: the CGA plays with a k-armed bandit machine where arms represent schemata. A trial of a given arm represent an additional string sampled in this schema.

existing schemata with relatively high fitness. Exploration is important in order to avoid local minima. Exploitation is important in order to converge to a good region. The answer provided by Holland on a two-armed bandit machine is that the optimal strategy is to allocate an exponentially increasing number of trials to the observed best arm. This is precisely what the CGA does, as shown in Section 2.2.2. It increases exponentially the number of strings that belong to schemata with more than average fitness.

2.4.2. Principle

Holland's demonstration is based on a two-armed bandit machine as represented in Fig. 4. When activated, one arm of the machine generates a payoff that is computed according to a probability distribution. The payoff associated to A1 (respectively A2) is generated according to a probability distribution of mean μ1 (resp. μ2) and standard deviation a1 (resp. ir2). These two distributions are supposed to be overlapping and are such that μ1 > μ2. One should assume that (μ1, a1) and (μ2,a2) are known values but one does not know to which arm they correspond. Thus, it is not possible to decide definitely that one arm is better than the other in a fixed amount of trials. There will always be a small probability of error.

The aim is to find a strategy that minimizes the losses when playing N trials on this machine. After N trials, one of the two arms will have received on the average more payoff than the other. This arm is named the observed best arm, but it may not correspond necessarily to the real best arm (μ_1, a_1) because of sampling errors. The other arm is named the observed worst arm. n denotes the number of trials allocated to the observed worst arm, and N - n thus represents the number of trials allocated to the observed best arm.

The first part of Holland demonstration consists in showing that there is an

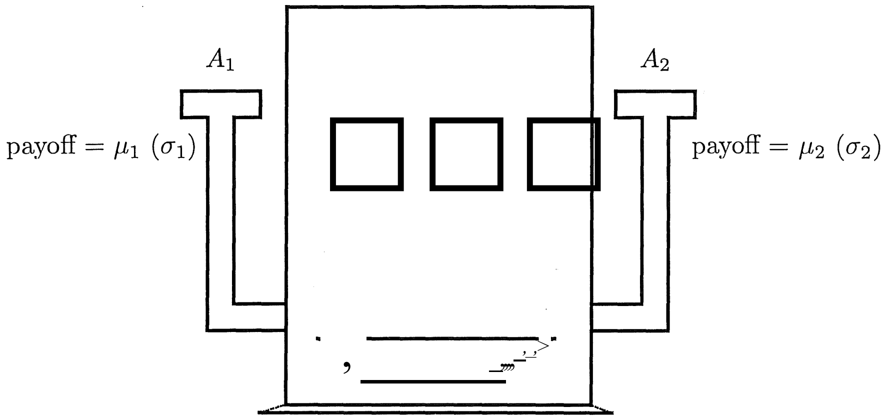

Figure 4. The two-armed bandit machine used in Holland's demonstration.

optimal value of n, denoted by n^*, which minimizes the losses in N trials. Two sources of losses exist in this problem. The first one is due to allocating n trials to the observed worst arm when this arm is really the worst one (A_2). This is the correct behavior: the observed best arm is really the best arm A_1, and the observed worst arm is A2, These n trials are, however, necessary because the worst arm must be sampled anyway in order to determine what payoff it brings. The second source of loss is due to allocating N - n trials to the best observed arm when this arm is in fact the worst arm (A_2). In this case, sampling errors have been such that A1 has been worse on the average than A_2, while $\mu 1 > \mu_2$. These two sources of loss are weighted by their probability of occurrence which depends on n. Holland demonstrates that the optimal value of n^* is such that the number of trials allocated to the observed best arm (N-n^*) is an exponential of the number of trials allocated to the observed worst arm (n^*).

However, this strategy is not useful in practice, because in the demonstration it is assumed that one knows which arm is the observed best arm even before you start playing. ·This strategy would thus require an oracle that would tell which arm will be the observed best one, and this is not realizable in practice. So, the second step of Holland's demonstration is to show that there exists a realizable strategy that asymptotically approximates the optimal but unrealizable strategy. This strategy can be stated as follows: (1) compute the value of n^*, (2) allocate n^* trials to each arm (i.e. a total of $2n^*$ trials), and (3) allocate the remaining trials (N - $2n^*$) to the observed best arm in the previous step (2).

Once these theoretical points are established, they can be related to schemata. The previous exponential ratio is used as a guideline for designing good strategies. When several schemata are in competition, a strategy that is the closest to the optimal but unrealizable one is to allocate an exponential number of trials to the observed best schemata with respect to the others. From one generation to the other, is has been shown that the number of strings in a schema increases exponentially when this schema has a fitness above the average (see Section 2.2.2). This is precisely the behavior suggested by Holland's demonstration.

In the second version of his book (Holland, 1992), Holland provides a different version of this demonstration.

2.4.3. Discussion on the bandit analogy

The optimal strategy is hence the one applied by the CGA. One should notice that this strategy docs not imply that the CGA will reach the optimum (De Jong, 1992): the CGA will only look for an optimal strategy for allocating trials.

This analogy also has limitations. The way the CGA solves competitions between schemata is not as simple as in the bandit machine. For instance, the schemata usefully processed may vary dynamically during the search. In practice, this strategy implies a too fast convergence, usually towards a local optimum that contains some building blocks. The sampling errors related to the small size of the population compared to 2^l, create this early convergence and reinforce the genetic drift phenomenon (Goldberg and Segrest, 1987): when two chromosomes have very close fitness but are very different f 'om each other in the bit string space, the stochastic behavior of selection will make the CGA converge only towards one of those two strings. This implies a loss of diversity in the population that is not taken into account in the determination of the optimal strategy.

2.5. Other work on schema analysis

Schema analysis is the basis for analyzing the CGA and is used in many theoretical works as can be seen in the remainder of this paper. Hence, the corresponding list of references would be too long and too general. However, concerning the fundamental theorem and other results presented in this section, an important part of the other work deals with the generalization to non-binary representations. This may concern alphabets with higher cardinality (Antonisse, 1989; Radcliffe, 1991; Radcliffe and Surry, 1994), set representations (Radcliffe, 1992), sequence representations (Fox and McMahon, 1991) or real-coded representations (Wright, 1991; Eshelman and Schaffer, 1992). All these representations are usually more efficient than binary ones, and it is important to generalize the theory ofthe binary CGA to other GAs.

3 Convergence theorems

Several standard methods in optimization are such that when they stop, the output solution is the optimum. This is the case for instance for branch and bound algorithms. The CGA does not provide such information. However, several theorems characterize how the convergence of the CGA takes place.

3.1. Convergence without mutation

This first theorem described in Ankenbrandt (1991), states that the CGA without mutation will converge to a uniform population in a given time. This population contains only one individual. However, this theorem does not tell whether this individual is the optimum or not.

Let us consider a population of randomly generated strings. The "fitness ratio" r is defined as follows: for a given position i in the string (i E [LZ]), let ri denote the ratio of the mean fitness of strings in a population with a "1" at position i over the mean fitness of strings with a "O" at this position. That is:

$$ri = \frac{f(*1 .. *i-1\ 1i\ *i+1\ .. *z, t)}{f(*1 .. *i-1\ Qi\ *i+1\ .. *z, t)}$$

Here, dynamic fitness is used and is considered at time $t = 0$. One may always assume that the binary encoding is such that ri 1. Let r denote the smallest ratio ri among $r_1, \ldots, r_l$. If no mutation is used in the CGA, the number of generations needed to obtain a uniform population is, in the average case, in the order of:

$$\frac{Order(!)\ nln(n)}{ln(r)}$$

where n is the population size and where *Order(!)* is the time complexity of the algorithm that computes the fitness function f. The complexity of f is taken into account. This theorem does not take into account the mutation operator because this operator makes the population diverge. This theorem can also be generalized for non binary strings.

3.2. Syntactic convergence

This theorem takes also into account only crossover and selection (Louis and Rawlins, 1992). It uses the mean of Hamming distances among the strings of the population as a measure of convergence of the population. Initially, this measure equals ½ on the average for a randomly generated population. When the population has converged to a single individual, then this measure equals 0. The authors show that standard crossover operators usually do not change the average Hamming distance of the strings. Mutation maintains a minimal value of this Hamming distance by changing randomly some bits and thus introducing different bit values at the same location. Selection makes this distance converge. The authors give an upper bound on the probability that all bits have converged at generation t This upper bound equals:

$$[1 - \frac{6po(1 - Po)}{n} (1 - \frac{}{n} /]z$$

where *p0* is the initial proportion of 0's in the population (which equals ½ in a randomly generated population).

3.3. Markov chain analysis of the CGA convergence

The CGA can be modeled by a Markov chain (Goldberg and Segrest, 1987; Eiben et al., 1991, a survey in De Jong et al., 1994), because this mathematical model allows to formalize stochastic processes where the probability that the process goes into a particular state depends only on the previous state. GAs are such that the new population is constructed only from the previous one. Markov chains are a very powerful tool to analyze their behavior. Finite chains can be specified using a transition matrix $M=$ [pi.ikJEG that contains the probability that the system jumps from state i to state j, where 8 denotes the set of all possible states of the system. For the CGA, 8 represents the set of all possible populations, and Pi.i the probability that population i becomes population j after one generation. Some theoretical works use the properties of this matrix M to obtain information about the behavior of the stochastic process.

In Goldberg and Segrest (1987), the authors studied the genetic drift, computing the expected time of convergence and comparing it for different models. This unexpected convergence, even without any selection pressure implies a loss of diversity in the population and some techniques have been created against it. For example, the creation of niches (subpopulations that evolve in parallel, De Jong, 1975; Goldberg and Richardson, 1987; Mahfoud, 1995; Jelasity and Dombi, 1995), is a good way to preserve the population from an early convergence. This technique has been studied also with Markov chains (Horn, 1993), and it has been proven that the expected time of convergence computed from M increases with the creation of niches, because this preserves the diversity in the population. Another way to slow convergence is to temper the selection pressure, using another selection operator. In Mahfoud (1993) the tournament selection is shown to reduce the convergence speed compared to the classic roulette wheel selection.

Another analysis using the transition matrix of a Markov chain model of GAs can be found in Rudolph (1994). The study of the limit distribution shows that elitism (keeping the best individual found so far through generations) is necessary to ensure convergence: the probability that the global optimum is in the population goes to 1. Along the same lines, the eigenvalues of M can be analyzed (Suzuki, 1993) and it can be shown that the probability that the populatation contains the optimum is bounded by 1-O(I>.(), ⊳l < 1, where t is the number of the generation and where>. is a specific eigenvalue of M. Suzuki also indicates how to choose the mutation rate in order to minimize >.

Results presented here come from the study of the properties of the transition matrix. In each model, simplifications were needed to extract usable results. This kind of approach allows mainly to compare the convergence speed of models of GAs using different operators, and using a simplified model of the fitness function. Another way to exploit the power of Markov chains model is presented now. Davis and Principe (1991) and Cerf (1995) are using similar principles to prove the convergence of GAs: modeling this algorithm with a simulated-annealing-like theory. The paper of Cerf goes further into the model: the CGA is viewed as a simple determinist selection process, perturbed by a Brownian movement that goes to zero with time. The theory of Freidlin and Wentzell (Freidlin and Wentzell, 1984) is then used to prove that, if the population is large enough and if the intensity of perturbations goes to zero slowly enough, the algorithm converges to the global optimum. This requires, among other things, that the mutation rate be decreasing in conjunction with higher and higher selective pressure. Several questions arise from the results obtained here. For instance, crossover is apprently not essential to obtain convergence but has a role to play in the speed of convergence. Also, in practice, conditions that ensure convergence are impossible to establish (the same problem as for simulated annealing), but this model gives a very good insight of the CGA convergence.

Finally, using infinite population model one can extract information about the role of crossover and mutation dispersion (Qi and Palmieri, 1994a; Qi and Palmieri, 1994b; Srinivas and Patnaik, 1993). It can be also proven that using this model in the specific case of a quadratic fitness function, the density of population is concentrating on the optimum. Some necessary conditions on the increase of mean evaluation are given, too.

All these models bring an insight into the way GAs converge, or diverge. But this depends strongly on the chosen definition of convergence, and also on the fitness landscape imposed by the encoding, the genetic operators as well as the fitness function itself. Even in practice, stopping criteria are very difficult to define in a satisfactory way. Models presented here allow to better understand the needed conditions to ensure convergence, and are promising. However, those conditions are as yet generally never realizable in practice, except for very simple evaluation functions that do not really need a GA to be optimized.

4. GA-easy or GA-hard?

4.1. General theorems

Some general theorems concern the optimization problem that the CGA is trying to solve. A summary of the consequences of the two general theorems presented in the following is that it is essential to characterize which functions are easy or hard to optimize with the CGA (as with any other search algorithm). This helps in understanding how the CGA really works, and it helps the engineer or user to aecide which search algorithm will best fit his problem.

4.1.1. An NP-hard problem

The optimization problem described in the introduction of this paper can be formalized into the DGA-max problem where f takes as input a binary string of length l and can give a positive or negative integer as output. f also has to be computed in polynomial time. DGA-max consists in finding s^* as defined previously. It is shown in Hart and Belew (1991), that DGA-Max is an NP-hard problem. This means that there are no polynomial time algorithms, either deterministic or probabilistic, that can find the maximum of f without any further hypothesis on f, unless P=NP. In particular, this is the case for the CGA. Furthermore, Hart and Belew show that no polynomial time algorithms, either deterministic or probabilistic, can approximate the maximum $f(s^*)$ within a given percentage of approximation.

The conclusion of this theorem is that it is useless to claim that an algorithm performs well on arbitrary functions, unless it explores the whole search space. This suggests, as will be highlighted by the next theorem, that characterizing which functions fit the CGA is very important.

4.1.2. No free lunch theorems

We will not go into philosophical discussion about the no free lunch theorems concerning search (Wolpert and Macready, 1995). The optimization problem the CGA deals with is also similar to the one studied by Wolpert and Macready. The no free lunch theorem mainly states that the performances of two search algorithms are equivalent when averaged on the set of all possible functions f, under a distribution $P(f)$ of functions which is uniform. We will not discuss here whether this theorem exactly applies to the CGA or not. However, it is interesting to notice that the consequence of this theorem is in a way similar to the one of the previous theorem: it is essential to know which distribution of functions $P(f)$ one is dealing with in a given problem, and it is essential to know how fit the CGA is with respect to these functions (Radcliffe and Surry, 1995).

4.2. Deception

We recall here several definitions and concepts introduced in Goldberg (1987; 1989a), Whitley (1991), Liepins and Vose (1991), Das and Whitley (1991).

4.2.1. Intuitive view

The behavior of the CGA can be modeled in a simple way as follows: initially, in a randomly generated population, many schemata of low order are represented in the population. Then, the order of frequently sampled schemata increases. For instance, if sampling errors arc negligible and if for instance $f(0**\ldots**) > f(1**\ldots**)$, then the CGA will concentrate in the first region $0**\ldots**$

rather than 1 * * ... * *. These two schemata can be viewed as competing schemata, because each of them would like to drive the search in the opposite direction of the other: the first schema assigns a Oto the first location, and the second schema assigns a 1 to the same location. This intuitive notion has been formalized as a competition between schemata that have the same locations instanciated with different bits. The winner of such a competition, that is the schema with the highest fitness among the competing schemata, will drive the search of the CGA in its direction. This seems reasonable if the winning schema contains the optimum: the CGA is driven towards a region of S that contains s*. However, if the winning schema does not contains*, then the CGA is driven away from the optimum. In this case, f is said to be deceptive. The work on deception is centered around this notion, as described in the following.

4.2.2. Competitions between schemata

Let us define first more precisely the notion of competition between schemata. A primary competition of order N is defined by comparing the fitness of schemata of order N which have * characters at the same locations and different instanciated bits. For instance, one may define a competition cl of order 1 between the 2 following schemata:

$$H_1^1 = 0 * * * **$$

Hi= 1 * * * **

One may also define a competition c_2 of order 2 between the 4 following schemata:

HJ;= 0 * 0 * **

HJ= 0 * 1 * **

HJ= 1 * 0 * **

Hi,= 1 * 1 * **

The schema that wins a competition c is the schema which has the highest static fitness among the schemata involved in c. For instance, if the two schemata involved in c_1 are such that *f(Hf)* > *f(Hi)*, which would be the case for the Onemax function, then *Hf* is the winner of competition cl.

A competition *CK* of order K is relevant to a competition *CN* of order N with N < K if every schema of cK is included in a schema of cN. For instance, the previous competition c_2 is relevant to c_1. The schemata *HJ,* and *H#* are included in *Hf,* and *HJ* and *Hi* are included in *Hf.* The following competition c; of order 2:

H' = *00 * *

H' = *01 * *

H' = * 1 0 * *

H' = *11 * *

is not relevant to c_1, because, for instance, H' is neither included in H l nor in *Hf,*

4.2.3. Deceptive functions

A function is (partially) deceptive of order *N* when there exists at least one competition *cN* of order N such that the winner of this competition has different bits than some winners of competitions of order less than *N* relevant to *cN,* More precisely, it is not necessary for all competitions that are relevant to *cN* to be misleading. Only one "path" of nested competitions relevant to *cN* has to be misleading. Also, one will be especially interested in the case where *cN* leads to *f's* optimum *s** while the misleading competitions do not. For instance, let us consider the two previous competitions cl and c2 on again. Let us suppose that the fitness function *f* is such that:

f (Hi) > f (Hf)

Then the winner of c_1 is the schema 0 * * * **. It assigns a "O" to the first location. Furthermore, if the function *f* is such that:

f(Hi) > f(HJ), f(H?), f(H?)

then the winner of c2 is the schema 1 * 1 * ** which assigns a "1" to the first location. The two competitions give two contradictory search directions. If the optimum *s** really belonged to *H#* = 1 * 1 * **, then solving the first order competition cl would lead away from that optimum. cl is a deceptive competition.

However, in partially deceptive functions, there may exist competitions which are not deceptive. For instance, another competition d_ of order 1, like for instance:

Hi = * * * * *0

H' = * * * * *1

may not be deceptive and may lead to the an optimum. In a partially deceptive function, there may exist a "path" of nested relevant competitions from order 1 to l which are not deceptive and which lead to the optimum.

4.2.4. Non deceptive functions

A function is non deceptive if all competitions of any order N lead to the optimum s^*. In those competitions, all schemata that win contain s^*. The Onemax function is a standard example of a non deceptive function. Let us consider Onemax for $l = 3$:

$$f(000) = 0,\ f(001) = 1,\ f(010) = 1,\ f(011) = 2$$

$$f(100) = 1,\ f(101) = 2,\ f(110) = 2,\ f(111) = 3$$

One can check that any competition is won by a schema that contains the optimum $s^* = 111$. For instance, the following inequalities involving first order competitions hold:

$$f(1**) > f(0**)$$

$$f(*1*) > f(*0*)$$

$$f(**1) > f(**0)$$

Any schema that contains $s^* = 111$ wins the competition it is involved in. In the same way, the following inequalities hold for the second order:

$$f(11*) > f(00*),\ !(01*),\ f(10*)$$

$$f(1*1) > f(0*0),\ f(0*1),\ f(1*0)$$

$$f(*11) > f(*00),\ f(*01),\ f(*10)$$

Finally, the order 3 competition is won by the optimum:

$$f(111) > f(000),\ f(001),\ \ldots\ f(100),\ f(101)$$

4.2.5. Fully deceptive functions

A function is fully deceptive of order N whenever there exists a competition CN of order N where all competitions relevant to CN of order less than N lead to a deceptive attractor $H-$, which is different from the winning schema H^* of cN. It is possible to show that this deceptive attractor is the binary complement of H^*, and that it is not necessarily a local optimum in the Hamming space (Whitley, 1991).

Let us detail a fully deceptive function of order 3 described in Whitley and Starkweather (1990). So let us suppose that $l = 3$ and that the global maximum of the function is $s^* = 111$. This implies that the deceptive attractor is $s- = 000$. All competitions of order 1 must be won by the schemata which contain 000. The following inequalities must hold:

$$f(0**) > f(1**)$$

$$f(*0*) > f(*h)$$

$$f(**0) > f(**1)$$

In the same way, all competitions of order 2 must be won by the schemata which contain s-, which implies:

$$f(00*) > f(0h),\ f(10*),\ f(1h)$$

$$f(0*0) > f(0*1),\ f(1*0),\ f(1*1)$$

$$f(*00) > f(*01),\ f(*10),\ f(*11)$$

Finally, only the third order competition is won by s*:

$$f(111) > f(000),\ f(001),\ \ldots\ f(100),\ f(101)$$

An example of such a function is:

$$f(000) = 28, f(001) = 26, !(010) = 22, f(011) = 0$$

$$f(100) = 14, f(101) = 0, f(110) = 0, f(111) = 30$$

4.2.6. The CGA and deception

As mentioned in the introduction of this section, many low order schemata are present in the first generation of individuals. The CGA solves competitions between schemata because it selects more often individuals with high fitness. If a competition is misleading, the CGA will get away from the optimum.

One would naturally be tempted to say that non deceptive functions are the GA-easy functions and that the fully deceptive functions are the GA-hard functions. However, this definition would be partially false, as explained in Grefenstette (1992). For instance, a "needle in a haystack" function with only one point with a fitness higher than Owould be GA-easy: any schema containing s* would win the competition it is involved in because the other schema would have a fitness of 0. This function, while being non deceptive, is hard to optimize for any robust search method including the CGA. Another example of non deceptive functions which gives troubles to the CGA are the royal road functions (Mitchell et al., 1991; Forrest and Mitchell, 1992; Mitchell and Holland, 1993). The current studies of deception fail to catch the dynamic properties of the CGA as they only consider static fitness, for instance, instead of dynamic fitness.

Experimental results on fully deceptive functions show that the CGA or other genetic algorithms usually do not find the optimum (Das and Whitley, 1991; see Grefenstette, 1992, for an exception). Experiments with non deceptive functions generally show that the CGA is able to find the optimum (Wilson, 1991). So deception has certainly got something to do with the CGA.

Several authors have proposed specific solutions for dealing with non deceptive and fully deceptive functions (Grefenstette, 1992; Louis and Rawlins, 1992). These solutions involve generally the use also of the complement of the strings, as will be explained in the following in the section on global search techniques.

4.3. Measures of function difficulty

The difficulty of functions can also be characterized by several measures that can be computed on a sample of strings.

4.3.1. Epistasis measures

An epistasis measure is useful to determine the correlations that may exist between binary genes (Davidor, 1991; Manela and Campbell, 1992; Reeves and Wright, 1995) or real-coded genes (Rochet et al., 1996). If no such correlations exist, then the influence of each bit value on the fitness function is independent of the values of the other bits. With this kind of measures, it is possible to define classes of functions which can be easy or hard for the CGA. Let us give an example of how a simple measure of epistasis can be computed with $l = 3$.

The aim of this measure is thus to tell how independent the bits are in a representation with respect to the fitness function f. For instance, if you can compute f(000) with knowledge of only f(0 * *), f(*0*) and f(* * 0), then the epistasis for these three bits will be low. In that case, you can determine $f(s)$ only by computing the interest of each bit of s independently from each other. This estimated value of $f(s)$, denoted by $A(s)$, is equal to (for $s = 000$):

$$A(000) = (f(0**) - f(***)) + (f(*0*) - f(***)) +$$

$$(f(**0) - f(***)) + f(***)$$

For instance, f(0 * *) - f(* * *) measures the interest of strings starting with 0 relative to all strings. If we consider the Onemax function for $l = 3$, then f (0 * *) = f (*0*) = f (* * 0) = 1 and f (* * *) = 1.5. Thus, A(000) equals:

$$(1 - 1.5) + (1 - 1.5) + (1 - 1.5) + 1.5 = 0$$

In this case, the value of f (000) is correctly predicted, and the epistasis of the string 000, measured by f(000) - A(000), equals 0.

One may compute for every strings from 000 to 111 the value of $A(s)$. Then an epistasis measure can be defined by computing the variance of the variable (f(s)-A(s)) for every strings. For Onemax on three bits, this measure equals 0. For the fully deceptive function of order 3 shown in Table 1 the epistasis measure equals 0.92.

In these two examples, we have implicitly assumed that the static fitness of schemata was used. The epistasis measure can be computed also only on the basis of the strings which are present in the population, i.e. with dynamic fitness of schemata. Davidor provides useful insight about the effect of sampling errors, which can change greatly the observed epistasis.

Functions with 0 static epistasis measure are non deceptive functions, and are always computed as linear weighted sum of the bits. Functions with high epistasis are generally deceptive, but counterexamples can be found where a

s	J(s)	A(s)
000	0.2	1.45
001	1.4	1.525
010	1.6	1.525
011	2.9	1.6
100	3	1.275
101	1	1.35
110	0.8	1.35
111	0.6	1.425

Table 1.

highly epistatic function is easy for the CGA (Manela and Campbell, 1992). If one increases the order of schemata which are used to compute the epistasis, then the epistasis may decrease, and this is the case for the function described in Manela and Campbell (1992) where a second order epistasis measure equals 0 (Rochet, 1996). This measure used second order schemata to compute $A(s)$.

4.3.2. Fitness/distance correlation

The fitness/distance correlation measures the correlation that may exist between the fitness value of a string s and its Hamming distance to the optimum s^* or the closest optimum when this function is multimodal (Jones and Forrest, 1995). This correlation is computed with a sample of points in S but requires that the global optima of f be known: one must collect n strings $s_1, \dots, s_n$ and evaluate their fitness $f(s_1), \dots, f(s_n)$ as well as their distance to the closest global optimum $d1, \dots, d_n$. Then, the fitness/distance correlation is the correlation of the couple $(f(s_i), d_i)$. When this correlation is close to -1, then the closer you get to one global optimums* the higher the fitness is. This is the case, for instance, of the Onemax function. This moans that f is easy to optimize by a GA, and most certainly easy for other methods, too. When this correlation is close to 1, the fitness function f is misleading, which is the case of fully deceptive functions, for instance. When this correlation is close to 0, no indication is really given about f's difficulty: f can be a "needle in a haystack" function, or a function with many high peaks located all over the search space.

4.3.3. Variance of schemata

Another way to obtain information about the expected behavior of the CGA is to consider the variance of fitness of schemata (Radcliffe, 1992). The schema theorem underlines the importance of building block recombination. Schemata with more than average dynamic evaluation, short length and small order are given an increasing number of copies in the subsequent generations. Depending on the binary strings of a schema that are present in a population, this schema will receive a growing or a decreasing number of strings. However, if the fitness values of strings that belong to a schema are very different from each other, i.e. have a high variance, then the CGA will consider this schema as a good or a bad one depending on the strings that are present in the current population. On the other hand, if the fitness values of strings that belong to this schema are close to each other, i.e. have low variance, then whichever strings are in the current population, the schema will be correctly evaluated.

As a consequence, studying the fitness variance in each schema can give an indication on the CGA behavior. In Radcliffe and Surry (1994) this tool is used to compare the performance of many encoding schemes on ordered chromosomes for the travelling salesman problem, with very good predictive results that allow to choose the right encoding for the problem to be solved.

4.4. Relations to other search methods

The definitions of easy and hard functions introduced in the previous sections can be related to each other. In addition, several authors have compared those definitions to other search methods like global search or hill climbing. It is interesting to notice, for instance, that non deceptive functions can be hard for other methods, or that fully deceptive functions can be easy for some other methods.

4.4.1. Global search

Non deceptive functions can be optimized by a simple and straightforward method introduced in Das and Whitley (1991). Since non deceptive functions are such that the optimum always wins the competition it is involved in, then this optimum can be reliably found by looking only at the first order schemata, which is a much simpler method than the CGA. This method is a global search algorithm which uses the following principle:

1. Generate a set S_l of strings in $\{0, 1\}^l$,
2. Evaluate each string of S_l,
3. Compute the fitness of all the first order schemata,
4. Generate the output st by solving the competitions of order 1:
    - the first bit of st equals 1 if $f(1 * \ldots *) > f(0 * \ldots *)$, else 0,
    - the second bit of st equals 1 if $f(*1 * \ldots *) > f(*0 * \ldots *)$, else 0,
    - and so on until the last bit,

More precisely, a statistical test, e.g. the T-test, can be used to determine whether more points should be generated or not. If the competitions are solved independently, then this algorithm can only deal with non deceptive functions. It may solve the partially deceptive functions in the following way: solve first the first order competition with the greatest fitness ratio to determine one bit of s1, and apply the same algorithm recursively to the remaining bits to be instanciated.

This global search can be simply extended to solve also fully deceptive functions (Venturini, 1995). The extended algorithm outputs the best of the two strings sj' and $\overline{\text{sj'}}$, where $\overline{\text{sj'}}$ denotes the binary complement of s1. Fully deceptive functions are such that competitions of low order arc won by schemata which contain the binary complement of the optimum. For such a function, solving the first order competitions leads to $\overline{s^*}$. This simple extension of the global search can deal efficiently with non deceptive or fully deceptive functions.

One should not conclude from the above that such algorithms are useful in practice. The set of functions that they can solved efficiently is reduced, and is certainly different from the set of real world problems. Functions can be found which drive these algorithms directly to the worst string of S However, it is clear that non deceptive or fully deceptive functions are not difficult problems for global search methods.

4.4.2. Hill climbing

We consider a simple hill climbing algorithm which can be stated as follows:

1. Choose randomly a starting point s,
2. Generate the points $s_1, \ldots, s_l$ in the neighbourhood of s by changing one bit of s,
3. Let s_i, $i \in [1, l]$, be the best point in the neighbourhood of s,
4. If $f(s_i) > f(s)$ then $s \leftarrow s_i$ and go to 2, else Stop.

This simple algorithm uses a Hamming distance to determine the neighbourhood of s. For instance with $l = 3$, the neighbours of 000 are 001, 010 and 100. In steps 3 and 4, the algorithm usually chooses the steepest ascent. We consider that a function f is easy for such a hill climber when it can get to the optimum whatever the starting point is. Otherwise, f is hard for hill climbing.

Wilson has shown that non deceptive functions arc not necessarily easy for hill climbers (Wilson, 1991) by giving a non deceptive function which contains local optima in the Hamming space. This function can be generalized to any dimension $l = 3k$

Whitley has shown that fully deceptive functions necessarily contain a local optimum in the Hamming space (Whitley, 1991). These functions are thus hard for hill climbers.

One should not conclude from this that hill climbers should not be used in cooperation with evolutionary techniques (Mulhenbein, 1992). By "hill climbing hard", one usually means that there may exist at least one local optimum in the Hamming space. However there may exist another path that leads to the optimum without getting trapped in a local optimum.

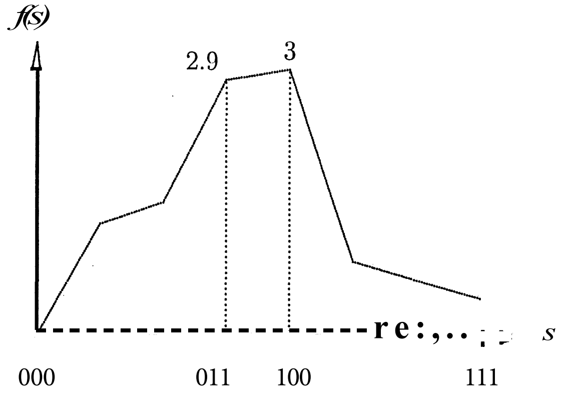

Figure 5. A fully deceptive function of order 3 which is gradient-easy.

4.4.3. Gradient search

Let us consider that the binary space is used to encode a parameter x. A gradient search could thus be used to optimize f. Here is an example of a unimodal, and thus easy for gradient search, but fully deceptive function for $l = 3$ (Venturini, 1995). The function's values are the following:

$$f(000) = 0.2, f(001) = 1.4, f(010) = 1.6, f(011) = 2.9$$

$$f(100) = 3, !(101) = 1, f(110) = 0.8, !(111) = 0.6$$

The optimum of this function is $s^* = 100$, and the deceptive attractor is thus s - = 011. This function is represented in Fig. 5 and it is obviously unimodal and easy for a gradient search.

4.5. Other work on GA-easy and GA-hard functions

Several other works have not been presented here. Among them, let us cite first the earlier work on deception which concerned the Walsh transform (Bethke, 1980; Goldberg, 1989b; 1989c; Bridges and Goldberg, 1991) or hyperplane transform (Holland, 1989).

Muhlenbein has computed for simple evolutionary algorithms the expected time to convergence to the optimum for three different functions (Muhlenbein, 1992). It is also possible to characterize the CGA behavior for several functions using entropy measures (Davidor and Ben-Kiki, 1992).

The study of royal road functions is also interesting as showing that a non deceptive function can be difficult to optimize because of the hitch-hiking phenomenon (Mitchell et al., 1991; Forrest and Mitchell, 1992; Mitchell and Holland, 1993). In these functions, chromosomes that contain good building blocks are reproduced with the bad schemata they may contain too. This phenomenon, called hitch-hiking, prevents the CGA from converging even if the fitness function seems easy at a first sight. Deception can also be analyzed for a given class of functions, like trap functions (Deb and Goldberg, 1992). For this class of functions, the proportion of fully deceptive, non deceptive or partially deceptive functions can be computed.

Messy GAs have also been designed to solve some of the problems inherent to the CGA and deception (Goldberg et al., 1991; Goldberg et al., 1993; Kargupta, 1995). One should also cite the works on the correlation of operators and on virtual GAs (Manderick et al., 1991; Grefenstette, 1995), which also characterize the difficulty of a function with respect to the genetic operators.

Finally, the CGA is not initially an optimization algorithm, and this explains some of its limitation from the optimization point of view (De Jong, 1992). As can be seen, it would be hazardous to fix definitely the definition of GA-easy and GA-hard functions, because additional understanding of the CGA is necessary.

5. Conclusion

In this paper, we have reviewed several important aspects of the theory of the simple GA. Many different mathematical tools are involved and many aspects of the CGA are concerned. They all contribute to the understanding of the CGA. The reader may have now a better idea of the important gap that exists between the easiness of description of the CGA and the difficulty of its analysis. We have only dealt here with the CGA, but several other evolutionary algorithms have been analyzecl theoretically (Spears et al., 1993).

However, the current theory of the CGA provides promising, though still partial answers to the fundamental questions stated in the introduction to this paper. Therefore, theoretical analysis of the CGA may look like a puzzle with different pieces that do not fit well altogether yet. This theory is not yet unified, and this is clue to the difficulty of the analysis. There are still many things to discover.

So perhaps the most fundamental question for the reader is about the usefulness of this theory in practical applications? The difficulty of the GA analysis is such that the theory is mainly dealing with the understanding, explanation or characterization of the GA behavior rather than with providing guidelines for empirical work. This will probably come later on, when the first aspects of the GA theory will have progressed.

From this point of view, it is possible to sketch what makes theoretical studies useful for practice. Thus, such studies should be:

- realistic: this involves probably dynamic versus static fitness, since dynamic fitness is the relevant fitness in a GA run. It should also involve all aspects of the GA, and not simply the operators alone, or the selection scheme alone, etc.

- realizable: this involves probably a sample of points rather than the whole search space. It involves also reasonable computational costs for modeling the GA.
- relevant: it should help practitioners to get the most from GAs for solving their current problems.

References

ACKLEY, D.H. (1987) *A connectionist machine for genetic hillclimbing.* Boston, MA, Kluwer Academic.

ANKENBRANDT, C. (1991) An extension to the theory of convergence and a proof of the time complexity of genetic algorithms. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 53-58.

ANTONISSE, J. (1989) A new interpretation of schema notation that overturns the binary encoding constraint. *Proceedings of the third International Conference on Genetic Algorithms,* J.D. Schaffer, ed., Morgan Kaufmann, 86-91.

BAKER, J. and GREFENSTETTE, J. (1989) How genetic algorithms work: a critical look at implicit parallelism. *Proceedings of the third International Conference on Genetic Algorithms,* J.D. Schaffer, ed., Morgan Kaufmann, 20-27.

BETHKE, A. (1980) *Genetic algorithms as function optimizers.* PhD Dissertation, Computer and Communication Sciences, University of Michigan, Ann Arbor.

BERTONI, A. and DORIGO, M. (1993) Implicit parallelism in genetic algorithms. *Artificial Intelligence,* 61, 2, 307-314.

BRIDGES, C.L. and GOLDBERG, D.E. (1991) The non uniform Walsh-schema transform. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 13-22.

CERF, R. (1995) An Asymptotic Theory of Genetic Algorithms. *Proceedings of Artificial Evolution 95,* Alliot J.-M., Lutton E., Ronald E., Schoenauer M. and Snyers D., eds., Lecture Notes in Computer Science 1063, Springer Verlag, 37-53.

COBB, H.G. and GREFENSTETTE, J.J. (1993) Genetic algorithms for tracking changing environments. *Proceedings of the Fifth International Conference on Genetic Algorithms,* S. Forrest, ed., Morgan Kaufmann, 523-530.

DAS, R. and WHITLEY, D. (1991) The only challenging problems are deceptive: global search by solving order-1 hyperplane, *Proceedings of the Fourth International Conference on Genetic Algorithms,* R.K. Belew and L.B. Booker, eds., Morgan Kaufmann, 166-173.

DAVIDOR, Y. (1991) Epistasis variance: a viewpoint on GA-hardness. *Proceedings of the .first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 23-35.

DAVIDOR, Y. and BEN-KIKI, O. (1992) The interplay among the genetic algorithm operators: information theory tools used in a holistic way. *Proceedings of the Second Conference on Parallel Problem Solving from Nature,* R. Manner and B. Manderick, eds., Elsevier, 75-84.

DAVIS, T.E. and PRINCIPE, J.C. (1991) A Simulated Annealing like Convergence Theory for the Simple Genetic Algorithm. *Proceedings of the fourth International Conference on Genetic Algorithms.* R. K Belew, L.B. Booker, ed., Morgan Kaufmann, 174-181.

DE JONG, K. (1975) *An analysis of the behavior of a class of genetic adaptive systems.* Doctoral Dissertation, University of Michigan.

DE JONG, K (1988) Learning with genetic algorithms: an overview. *Machine Learning,* 3, 121-138.

DE JONG, K (1992) Are genetic algorithms function optimizers? *Proceedings of the Second Conference on Parallel Problem Solving from Nature,* R. Maenner and B. Manderick, eds., Elsevier, 3-13.

DE JONG, K., SPEARS, W.M. and GORDON, D.F. (1994) Using Markov chains to analyze GAFOs. *Proceedings of the third Workshop on Foundations of Genetic Algorithms.*

DEB, K. and GOLDBERG, D.E. (1992) Analyzing deception trap functions. *Proceedings of the second Workshop on Foundations of Genetic Algorithms,* D. Whitley, ed., Morgan Kaufmann, 93-108.

EIBEN, A.E., AARTS, E.H.L. and VAN HEE, K.M. (1991) Global convergence of a genetic algorithm: a Markov Chain analysis. In: H.P. Schwefel and R. Manner, eds., *Parallel Problem Solving from Nature,* LNCS 496, Springer Verlag, 4-12.

ESHELMAN, L.J. and SCHAFFER, J.D. (1992) Real-coded genetic algorithms and interval schemata. *Proceedings of the second Workshop on Foundations of Genetic Algorithms,* D. Whitley, ed., Morgan Kaufmann, 187-202.

FORREST, S. and MITCHELL, M. (1992) Relative building block fitness and the building block hypothesis. *Proceedings of the second Workshop on Foundations of Genetic Algorithms,* D. Whitley, ed., Morgan Kaufmann, 109-126.

Fox, B.R. and McMAHON, M.B. (1991) Genetic operators for sequencing problems. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 284-300.

FREIDLIN, M.I. and WENTZELL, A.D. (1984) *Random perforbations of dynamical systems.* Springer-Verlag, New York.

GOLDBERG, D.E. (1987) Simple genetic algorithms and the minimal deceptive problem. *Genetic Algorithms and Simulated Annealing,* L. Davis, ed., Morgan Kaufmann, 74-88.

GOLDBERG, D.E. and RICHARDSON, J. (1987) Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms,* J.J. Grefenstette, ed., LEA Pub, 41-49.

GOLDBERG, D.E. and SEGREST, P. (1987) Finite Markov Chain Analysis of Genetic Algorithms. *Proceedings of the second International Conference on Genetic Algorithms,* J.J. Grefenstette, ed., LEA Pub.

GOLDBERG, D.E. (1989A) *Genetic Algorithms in Search, Optimization and Machine Learning:* Addison Wesley.

GOLDBERG, D.E. (1989B) Genetic algorithms and Walsh functions: part I, a gentle introduction. *Complex systems,* 3, 129-152.

GOLDBERG, D.E. (1989c) Genetic algorithms and Walsh functions: part II, deception and its analysis. *Complex systems,* 3, 153-171.

GOLDBERG, D.E., DEB, K. and KORB, B. (1991) Don't worry, be messy. *Proceedings of the Fourth International Conference on Genetic Algorithms,* R.K. Belew and L.B. Booker, eds., Morgan Kaufmann, 24-30.

GOLDBERG, D.E., DEB, K., KARGUPTA, H. and HARIK, G. (1993) Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. *Proceedings of the 5th International Conference on Genetic Algorithms.* Stephanie Forrest, ed., Morgan Kaufmann, 56-64.

GREFENSTETTE, J.J. (1991) Conditions for implicit parallelism. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 252-261.

GREFENSTETTE, J.J. (1992) Deception considered harmful. *Proceedings of the second workshop on Foundations of Genetic Algorithms,* 1992, D. Whitley (Ed), Morgan Kaufmann, 75-91.

GREFENSTETTE, J.J. (1995) *Virtv.al genetic algorithms: .first resv,lts.* Nayy Center for Applied Research in Artificial Intelligence, Technical Report AIC-95-013.

HART, W.E. and BELEW, R.K. (1991) Optimizing an arbitrary function is hard for the genetic algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithms,* R.K. Belew and L.B. Booker, eds., Morgan Kaufmann, 190-195.

HOLLAND, J.H. (1975) *Adaptation in natv,ral and artificial systems.* Ann Arbor: University of Michigan Press.

HOLLAND, J.H. (1989) Searching nonlinear functions for high values. *Applied Mathematics and Computation,* 32, 255-274.

HOLLAND, J.H. (1992) *Adaptation in natv,ral and artficial systems: an introdv.ctory analysis with applications to biology, control, and artificial intelligence,* MIT Press.

HORN, J. (1993) Finite Markov Chain Analysis of Genetic Algorithms with Niching. *Proceedings of the 5th International Conference on Genetic Algorithms.* Stephanie Forrest, ed., Morgan Kaufmann, 110-117.

JELASITY, M. and DOMBI, J. (1995) GAs, a Concept on Modeling Species in Genetic Algorithms. *Proceedings of Artificial Evolution 95,* Alliot J.-M., Lutton E., Ronald E., Schoenauer M. and Snyers D., eds., Lecture Notes in Computer Science 1063, Springer Verlag, 69-85.

JONES, T. and FORREST, S. (1995) Fitness distance correlation as a measure of problem difficulty for genetic algorithms. *Proceedings of the Sixth International Conference on Genetic Algorithms,* L.J. Eshelman, ed., Morgan Kaufmann, 184-192.

KARGUPTA, H. (1995) *Search, Polynomial complexity and the Fast Messy Genetic Algorithm.* PhD thesis, Dept of Computer Science, Univ. of Illinois at Urbana Champaign.

LIEPINS, G.E. and VOSE, M.D. (1991) Deceptiveness and genetic algorithm dynamics. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., 36-50.

Lours, J.L. and RAWLINS, J.E. (1992) Syntactic analysis of convergence in genetic algorithm. *Proceedings of the second workshop on Foundations of Genetic Algorithms,* D. Whitley, ed., Morgan Kaufmann, 141-151.

MAHFOUD, S.W. (1993) Finite Markov chain models of an alternative selection strategy for the genetic algorithm. *Complex Systems,* 7, 2, April 1993, 155-170.

MAHFOUD, S.W. (1995) A comparison of parallel and sequential niching methods. *Proceedings of the Sixth International Conference on Genetic Algorithms,* L.J. Eshelman, ed., Morgan Kaufmann, 136-143.

MANDERICK, B., DE WEGER, M. and SPIESSENS, P. (1991) The genetic algorithm and the structure of the fitness landscape. *Proceedings of the Fo11,rth International Conference on Genetic Algorithms,* R.K. Belew and L.B. Booker, eds., Morgan Kaufmann, 143-150.

MANELA, M. and CAMPBELL, J.A. (1992) Harmonic analysis, epistasis and genetic algorithms. *Proceedings of the Second Conference on Parallel Problem Solving from Nafore,* R. Manner and B. Manderick, eds., Elsevier, 57-64.

MITCHELL, M., FORREST, S. and HOLLAND, J.H. (1991) The royal road for genetic algorithms: fitness landscapes and GA performance. *Proceedings of the .first European Conference on Artificial Life,* F.J. Varela and P. Bourgine, eds., MIT Press/Bradford Books, 245-254.

MITCHELL, M. and HOLLAND, H. (1993) When will a genetic algorithm outperform hill climbing? *Proceedings of the Fifth International Conference on Genetic Algorithms,* S. Forrest, ed., Morgan Kaufmann, 647-647.

MUHLENBEIN, H. (1990) Evolution in time and space - The parallel genetic algorithm. *Proceedings of the .first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 316-337.

Qr, X. and PALMIERI, F. (1994A) Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part I: basic properties of selectiop and mutation. *IEEE Transactions on Ne11,ral Networks,* 5, 1, January 1994, 102-119.

Qr, X. and PALMIERI, F. (1994B) Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part II: analysis of the diversification role of crossover. *IEEE Transactions on Neural Networks,* 5, 1, January 1994, 120-128.

RADCLIFFE, N.J. (1991) Equivalenceclassanalysisofgeneticalgorithms. *Complex Systems,* 5, 2, 183-205.

RADCLIFFE, N.J. (1992) Genetic set recombination. *Proceedings of the second Workshop on FoV,ndations of Genetic Algorithms,* D. Whitley, ed., Morgan Kaufmann, 203-219.

RADCLIFFE, N.J. and SURRY, P.D. (1994) Fitness variance of formae and performance prediction. *Proceedings of the third Workshop on Fov,ndations of Genetic Algorithms.*

RADCLIFFE, N.J. and SURRY, P.D. (1995) Fundamental Limitations on Search Algorithms: Evolutionary Computating in perspective. In: *Comptder Science Today: Recent Trends and Developements,* J. van Leeuwen, ed., LNCS 1000, Springer Verlag, 275-291.

REEVES, C.R. and WRIGHT, C.C. (1995) Epistasis in genetic algorithms: an experimental design perspective. *Proceedings of the Sixth International Conference on Genetic Algorithms,* L.J. Eshelman, ed., Morgan Kaufmann, 217-224.

RoCHET, S. (1996) *Epistasis in genetic algorithms revisited.* To appear in *Information Sciences.*

RoCHET, S., SLIMANE, M. and VENTURINI, G. (1996) Epistasis for real encoding in genetic algorithms. *IEEE ANZIIS'96,* V. L. Narasimhan and L. C. Jain, eds., Australia, 268-271.

RUDOLPH, G. (1994) Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks,* 5, 1, 96-101.

SCHAFFER, J.D., ESHELMAN, L.J. and OFFUTT, D. (1991) Spurious correlation and premature convergence in genetic algorithms. *Proceedings of the first Workshop on Foll,ndations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 102-112.

SPEARS, W.M., DE JONG, K.A., BAECK, T., FOGEL, D.B. and DE GARIS, H. (1993) An overview of evolutionary computation. *Proceedings of the Ell,ropean Conference on Machine Learning,* P. Brazdil, ed., Lecture notes in artificial intelligence 667, Springer-Verlag, 442-459.

SPIESSENS, P. and MAND ERICK, B. (1991) A massively parallel genetic algorithm implementation and first analysis. *Proceedings of the Fov,rth International Conference on Genetic Algorithms,* R.K. Belew and L.B. Booker, eels., Morgan Kaufmann, 279-286.

SRINIVAS, M. and PATNAIK, L.M. (1993) Binomially Distributed Population for Modelling Genetic Algorithms. *Proceedings of the 5th International Conference on Genetic Algorithms,* S. Forrest, ed., Morgan Kaufmann, San Mateo, CA, 138-143.

Suzu.n, J. (1993) A Markov Chain Analysis on a Genetic Algorithm. *Proceedings of the Fifth International Conference on Genetic Algorithms,* S. Forrest, ed., Morgan Kaufmann, 146-153.

SYSWERDA, G. (1989) Uniform crossover in genetic algorithms. *Proceedings of the third International Conference on Genetic Algorithms,* J.D. Schaffer, ed., Morgan Kaufmann, 2-10.

VENTURINI, G. (1995) Towards a genetic theory of easy and hard functions. *Proceedings of Artficial Evolution 95,* Alliot J.-M., Lutton E., Ronald E., Schoenauer M. and Snyers D. eds., Lecture Notes in Computer Science 1063, Springer Verlag, 54-66.

WHITLEY, D. (1989) The genitor algorithm and selective pressure: why rank-based allocation of reproductive trials is best. *Proceedings of the third International Conference on Genetic Algorithms,* J.D. Schaffer, ed., Morgan Kaufmann, 116-124.

WHITLEY, D. and STARKWEATHER, T. (1990) Genitor II: a distributed genetic algorithm, *J Expt. Theor. Artif. Intell.,* 2, 189-214.

WHITLEY, D. (1991) Fundamental principles of deception in genetic search. *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 221-241.

WILSON, S.W. (1991) GA-easy does not imply steepest-ascent optimizable. *Proceedings of the Fourth International Conference on Genetic Algorithms,* R.K Belew and L.B. Booker, eds., Morgan Kaufmann, 85-89.

WOLPERT, D.H. and MACREADY, W.G. (1995) *No free lunch theorems for search.* Technical report SFI-TR-95-02-010, The Santa Fe Institute, 1995,

WRIGHT, D, (1991) Genetic algorithms for real parameter optimization, *Proceedings of the first Workshop on Foundations of Genetic Algorithms,* G.J.E. Rawlins, ed., Morgan Kaufmann, 205-218.

CALL FOR PAPERS & ANNOUNCEMENT

THE ELEVENTH INTERNATIONAL CONFERENCE ON INDUSTRIAL AND ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IEA/AIE-98)

Submission Deadline: November 7, 1997

Conference Location and Date: Hotel Intur Orange Benicassim, Castellon, Spain, June 1-4, 1998.
Sponsored by the International Society of Applit:d Intelligence and organized in cooperation with major international organizations, including ACM/SIGART; AAAI; INNS; IEE; ECCAI; CSCSI; JSAI; Southwest Texas State University; the Universitat Jaume-I de Castellon; and the Universidad Nacional de Education a Distancia, Madrid.
Submit five copies of long papers written in English (up to 10 single-spaced pages) by November 7, 1997, to Dr Angel P. del Pobil, Program Co-Chair, IEA/AIE-98 Conference, Department of Informatics, Jaume-I University, Campus de Penyeta Roja, E-12071 Castellon, Spain. Fax (+34) 64 345.848; E-mail: iea98©titan.inf.uji.es; www: http://titan.inf.uji.es/IEA98/