The Role of Selection in Genetic Algorithms

Vladimir Estivill-Castro
School of Information Systems,
Queensland University of Technology,
Brisbane, 4000, Australia.
vlad@icis.qut.edu.au

Tech. Report ♯4/96

Abstract

Genetic Algorithms have gained wide popularity due to their conceptual simplicity and their ability to obtain approximate solutions for complex searching and optimization problems. Their wide applicability contrasts with their rather undeveloped theoretical basis. In particular, the convergence of genetic algorithms has been one of the most challenging theoretical issues in the evolutionary computation area.

We provide a careful analysis of the structure of the classical genetic algorithm. We demonstrate that, when a Genetic Algorithm is used to maximize a function, selection is the process that assures convergence to the global optimum (asymptoticly in the limit) for the fitness of the population.

Our results have several practical implications; among these, the number of generations up to which reasonable improvement could be expected may be approximated ahead or during the working of the genetic algorithm.

1 Introduction

Genetic algorithms have gained popularity due to their conceptual simplicity and their ability to obtain approximate solutions for complex searching and optimization problems at reasonable cost. The fields of engineering and optimization have benefited from many applications of this type of methods. The properties of genetic search have motivated their use as an alternative to other search strategies. We mention two examples in the field of machine learning. For a first example, consider the proposals [7, 12] to use genetic search (to induce decision trees from a set of examples) as an replacement for the hill-climbing search within the well-known inductive learning algorithm ID3 [23]. For a second example, consider the training of multilayered perceptrons. Training usually minimizes a least-squares fit error measure between desired outputs and current outputs [25]. Typically, *gradient descent* is used when adjusting the weight matrix of neural networks. Genetic search has also been proposed as an alternative for minimizing this error measure [27, 15].

The wide applicability of genetic algorithms contrast with their rather undeveloped theoretical basis. Genetic algorithms are probabilistic methods and their *Schema Theorem* [3] explains why their search can be regarded as guided by a decision theory heuristic. Similarly, *implicit parallelism* results [2, 3] give indication of the effectiveness of the computation.

The convergence of genetic algorithms is one of the most challenging theoretical issues in the evolutionary computation area [8, 18]. Several researchers have previously explored this problem mainly from two different perspectives: Analyses using Markov chains [5, 9, 14, 26] and analyses to identify characteristics of the objective functions that affect the behavior of genetic algorithms [13, 16]. Goldberg and Segrest [14] provide a finite Markov-chain analysis of a genetic algorithm (finite population, reproduction, mutation only and restricting the analysis to chromosomes consisting of binary strings of length one). Davis and Principe [5] investigated the possibility of extending the existing theoretical foundation of simulated annealing algorithms onto a model of a Markov chain for genetic algorithms, and consequently, the role of mutation in genetic algorithms parallels the role of temperature control in simulated annealing. Eiben, Aarts and Van Hee [9] propose an abstract genetic algorithm which unifies genetic algorithms and simulated annealing; a Markov chain analysis on such abstract genetic algorithm is discussed and theoretical conditions implying that the evolution process finds an optimum with probability 1 are given. Kingdom [16] investigates starting points, convergence and the class of problems genetic algorithms find hard to solve. Kingdom's generalizes the notion of competing schemata and provides estimates for the probability of convergence of such schemata. An exact mathematical analysis has been developed for a genetic algorithm with uniform crossover for the case when the objective function is the number of ones in a binary string of length $d \geq 1$ [24]. Rudolph [26] proved that a classical genetic algorithm using a finite number of steps never converges to the global optimum, but modified versions, which maintain the best solution in the population (elitist model) do. Recently, techniques from the classic science of livestock breading have been used to estimate the variance and the mean of the fitness of the population in a Breeder Genetic Algorithm [30, 21] Although these results include an analysis with combination operators (for example, uniform crossover), they usually illustrate the difficulties in obtaining predictions because some factors (like the *realized heritability*) can not be accurately estimated as a result of untractable equations. However, such results have motivated the design of new techniques to produce offspring [20]. Other results on convergence of genetic algorithms [18] show that as long as the algorithm makes an improvement, it will converge to the global optimum. This claim is immediate, given that the algorithm is exploring a finite space. The real issue, in this case, will be proving that the algorithm never gets stranded on a local optimum. Finally, other results on different coding schemes, genetic operators and robustness to the characteristics of the objective function are mainly the product of experimentation [1].

We provide a step in addressing the problem of convergence by demonstrating that, for the classical genetic algorithm, selection assures convergence (in the limit) of the expected fitness of the population. As opposed to Rudolph's [26] analysis, we do not require that the algorithm maintains copies of the best individuals from one generation to the next. As a corollary of our results, we confirm that any single run of a genetic algorithm that terminates after a finite number of operations is not expected to attain the optimum, and that predictable performance is obtained when averaging several executions of the genetic algorithm.

Our results will also provide a mechanism for addressing a practical concern; namely, the number of generations that a genetic algorithm must be allowed to run to achieve acceptable precision without wasting additional computational resources. Our analysis describes the fundamental role of selection in the behavior of a genetic algorithm and contributes to the underlaying principles that describe their operation. Thus, we extend previous theoretical work on genetic algorithms based on selection only [5, 14]. However, we do not intend to imply that, in practice, other genetic operators should not be incorporated — crossover or mutation are necessary for exploring the search space beyond the points obtained in the first generation.

Section 2 provides an introduction to the classical genetic algorithm and the nomenclature used when discussing genetic algorithms. Section 3 presents the results on the expected value of the approximation obtained by the genetic algorithm. Section 4 discusses the practical implications of the results presented in Section 3. We conclude with some final remarks in Section 5. Genetic algorithms will most likely be used when the objective function is unknown; however, through our presentation, we will illustrate our results and clarify the meaning of our notation with the maximization of the objective function

$$F_0(x) = \begin{cases} 3 & \text{if } x > 0, \\ 1 & \text{if } x \leq 0; \end{cases}$$

and with the objective function $F_1(x) = 255^2 - x^2$.

2 The classical genetic algorithm

A genetic algorithm is a probabilistic method inspired upon an abstract model of our current understanding of the natural process of evolution. Because of this, the nomenclature imports some terminology from biology and genetics. Through the process, an initial *population* of encoded individuals (or *chromosomes*) improves as parents are replaced by better offspring. Thus, two mechanisms are required to use a genetic algorithm. The first is a mechanism for awarding merit to an individual, the second is the encoding to represent individuals for their manipulation by the computer program. When using a genetic algorithm to maximize[1] a given function F, the value $F(c)$ is considered the *fitness* of the individual c. The encoding mechanism allows the representation of individuals (*phenotypes*) as chromosomes (*genotypes*).

For example, if we are maximizing the function $F_1(x) = 255^2 - x^2$ in the interval $[-255, 255]$, then the fitness for the value 6.5 is $F_1(6.5) = 64,982.75$. By representing

[1]In this paper we discuss only maximizing a positive function, since minimizing F is equivalent to maximizing $-F$; and also maximizing $F'(x) = F(x)[F(x) > 0]$ (where $[F(x) > 0]$ is 1 if the predicate is true and 0 otherwise) is equivalent to maximizing $F(x)$.

GA $(\mathcal{S}, F, n, Prob_{cross}, Prob_{mutation}, stop)$;
  Initialize population :
    $generation \leftarrow 0$;
    $P_0 \leftarrow \langle c_1, c_2, \ldots c_n \rangle$ where c_i is selected, independently and uniformly in $\mathcal{S}$.
  Repeat
    $Parents \leftarrow \langle c'_1, c'_2, \ldots, c'_n \rangle$ where c'_i is selected, independently and randomly from
      $P_{generation} = \langle c_1, c_2, \ldots, c_n \rangle$ with probability distribution
      $Prob[c'_i = c_{j0}] = F(c_{j0}) / \sum_{j=1}^{n} F(c_j)$.
    For each letter α in each chromosome $c'_i \in Parents$
      change α with probability $Prob_{mutation}$.
    For $i := 1$ to $\lfloor n/2 \rfloor$
      replace c'_{2i} and c'_{2i-1} with their crossover offspring
      with probability $Prob_{cross}$.
    $generation \leftarrow generation + 1$;
    $P_{generation} \leftarrow Parents$
  until *stop*

Figure 1: Algorithmic description of GA. The size of the population is n, the function F is the objective function and $Prob_{cross}$ is the probability of crossover while $Prob_{mutation}$ is the probability of mutation.

integers in binary with a leading bit for the sign, the integers in the interval $[-255, 255]$ could be encoded as a binary string of length 9, such that $c = b_8 b_7 \ldots b_0$ (with $b_i \in \{1, 0\}$, for $i = 0, \ldots, 8$) represents the integer $x = (-2b_8 + 1) \times \sum_{i=0}^{7} b_i 2^i$.

It is important to observe that the genetic algorithm will work in the space $\mathcal{S}$ of encodings (*chromosomes* or *genotypes*) usually given as $\mathcal{A}^k$ where $\mathcal{A}$ is an alphabet and k is the length of chromosomes; and thus, $\mathcal{S}$ is finite. In the example above, the space of encodings is the set $\mathcal{S}_1 = \{0, 1\}^9$ of all binary strings of length 9 (but other encodings are possible, even decimal notation, floating point representation or even a classifier encoded as LISP's S-expressions).

The algorithmic description of a *classical* or *traditional* genetic algorithm [4] in its generic format is presented in Figure 1. We will consider that mechanisms, like scaling the function F, are just modifying the function F that the genetic algorithm actually sees via composition with other functions. We will not elaborate on the details of crossover operators. *Selection* is the process that extracts the set of *Parents* from which the new population is produced. Note that the probability distribution from which a parent is drawn is given by the ratios of fitness values of each chromosome to the total fitness value of the current population. Termination of execution is determined by a boolean criteria (*stop*); usually, when a large number of generations is reached or when all chromosomes are extremely alike.

If the probabilities by which genetic operators work are set to zero ($Prob_{cross} = 0$ and

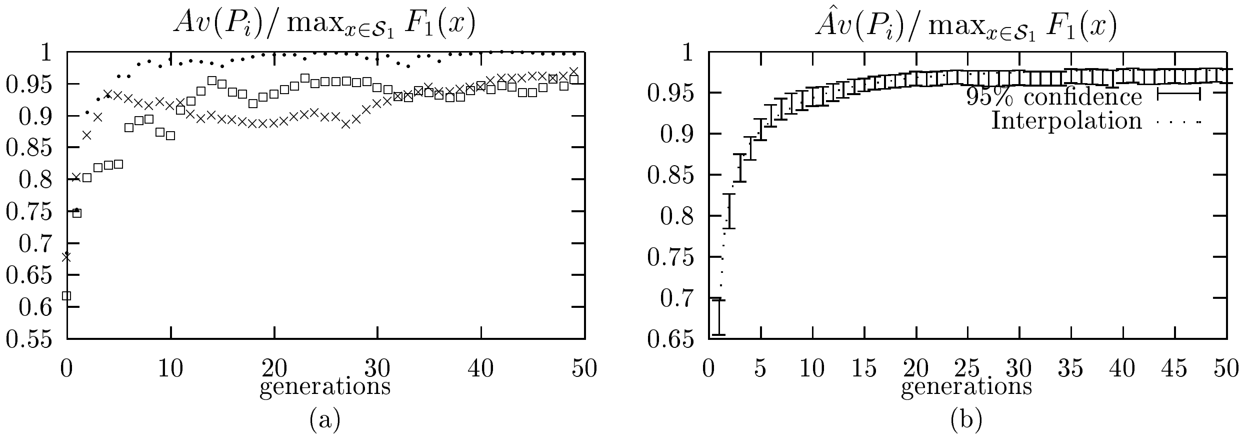

Figure 2: Execution of the GA for maximizing $F_1(x) = 255^2 - x^2$ with population size 50 chromosomes and stopping after 50 generations.

$Prob_{mutation} = 0$), then the set of parents is the new generation. That is, the set of parents prevails intact to the new generation and only the selection process is at work. Note that populations are manipulated by the genetic algorithm as sequences; that is, the relative order of chromosomes is relevant in describing a population.

Definition 2.1 *Let $P_i = \langle c_1, \ldots, c_n \rangle$ be the i-th population of a particular execution of the genetic algorithm GA. The population fitness of P_i is defined by $Av(P_i) = \sum_{i=1}^{n} F(c_i)/n$.*

Note that, because the genetic algorithm is a probabilistic algorithm, $Av(P_i)$ is a random variable depending on the random choices of the algorithm. Moreover, $Av(P_i) \leq max_{x\in\mathcal{S}} F(x)$ always. For example, Figure 2 (a) plots $(i, Av[P_i])$ for three executions of the genetic algorithm when maximizing the function $F_1(x) = 255^2 - x^2$ with $Prob_{cross} = 0.6$ and $Prob_{mutation} = 0.001$.

To define the expected value of $Av(P_i)$ we use the same approach used for expected-case analysis [17]. For example, we evaluate all possible executions of the algorithm to define the expected value of the number of comparison performed by Randomized Quicksort (where the algorithm chooses the pivot to partition a sublist, randomly and uniformly, among the elements of the sublist).

Definition 2.2 *The expected value of the population fitness $Av(P_i)$ is defined as follows*

$$E[Av(P_i)] = \sum_{\textit{all possible executions of GA}} Prob[P_i]Av(P_i).$$

The variance of $Av(P_i)$ is also the usual notion of variance, namely,

$$VAR[Av(P_i)] = \sum_{\textit{all possible executions of GA}} Prob[P_i](Av(P_i) - E[Av(P_i)])^2.$$

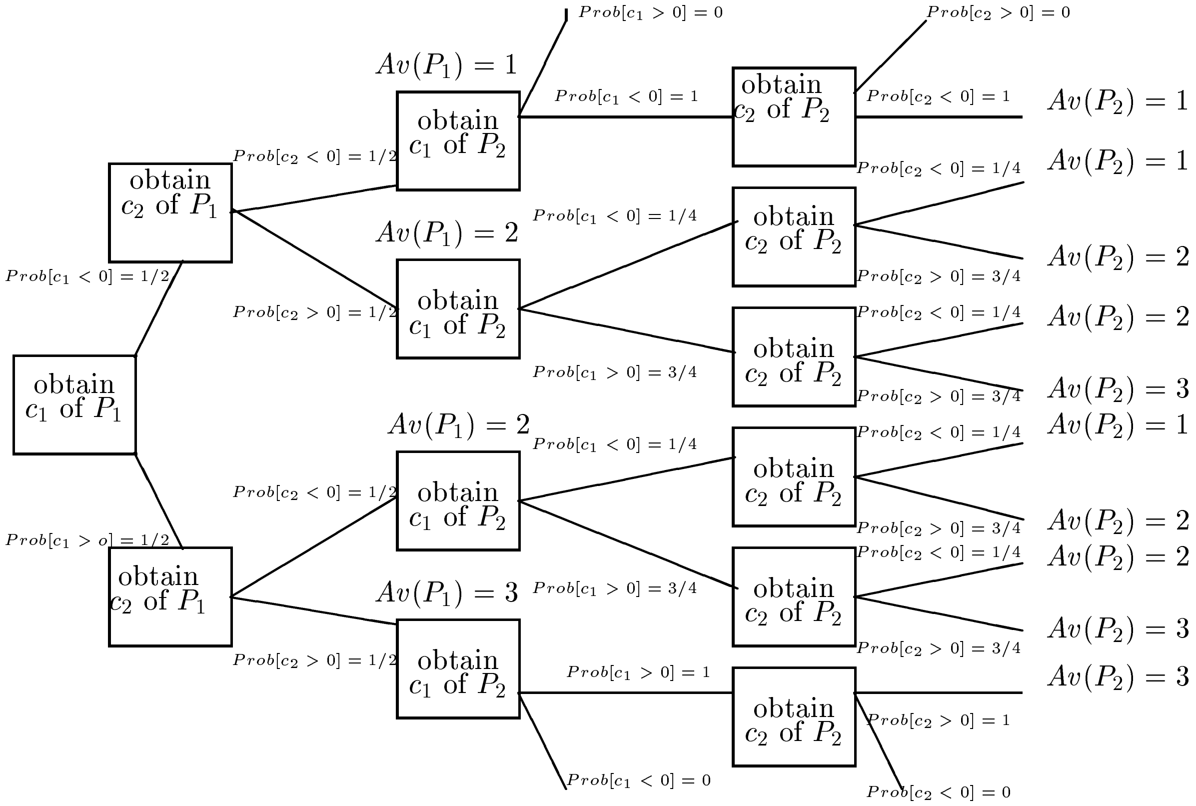

Figure 3: Randomized decision tree for maximizing $F_0(x)$ with a population size of $n = 2$. The first generation is obtained at depth two. There are four events for the first generation according to if the 2 chromosomes are greater or less than zero.

Note that $E[Av(P_i)] \leq \max_{x \in \mathcal{S}} F(x)$, for all $i \geq 0$.

We can illustrate the meaning of $E[Av(P_i)]$ with the unfolding of the randomized execution in a randomized decision tree [17]. The nodes of the tree are labeled with instructions and tests while edges are labeled with the probability that the algorithms follows the edge. An illustration of such a tree is given in Figure 3. In this figure, the objective function is $F_0(x)$, the search space is $\mathcal{S}_0 = \{-m, -m+1, \ldots, -1, 1, 2, \ldots, m\}$ and the population size is $n = 2$. The event space for P_i is $\{\langle c_1, c_2 \rangle \mid c_i \in \mathcal{S}_0\}$, and it is the disjoint union of four events, each with probability 1/4. These events are $\text{Event}_1 = \{c_1 < 0 \;\wedge\; c_2 < 0\}$, $\text{Event}_2 = \{c_1 < 0 \wedge c_2 > 0\}$, $\text{Event}_3 = \{c_1 > 0 \wedge c_2 < 0\}$, and $\text{Event}_4 = \{c_1 > 0 \wedge c_2 > 0\}$.

Because the genetic algorithm is working with $F_0(x)$, we have that $Av(P_1) = (1+1)/2 = 1$, when $P_1 \in \text{Event}_1$; while $Av(P_1) = (1+3)/2 = 2$, when $P_1 \in \text{Event}_2 \cup \text{Event}_3$, and $Av(P_1) = (3+3)/2 = 3$, when $P_1 \in \text{Event}_4$. Thus, $E[Av(P_1)] = (1+2+2+3)/4 = 2$.

However, past the first generation, the probabilities to prevail in further generations are in proportion to the fitnesses amongst other chromosomes in the current population. For example, if $P_1 \in \text{Event}_1$, then $Prob[c_1 > 0 \wedge c_1 \in P_2] = 0$, while $Prob[c_1 < 0 \wedge c_1 \in P_2] = 1$.

Similarly, if $P_1 \in \text{Event}_2$, then $Prob[c_1 < 0 \;\wedge\; c_1 \in P_2] = F(c_1)/\sum_{c_i \in P_1} F(c_i) = 1/4$, while $Prob[c_1 > 0 \;\wedge\; c_1 \in P_2] = 3/4$.

The example is summarized in Figure 3 for 2 generations. A simple calculation shows that

$$E[Av(P_2)] = \frac{1}{4}(1) + \frac{2}{4}\left\{\frac{1}{16}(1) + \frac{6}{16}(2) + \frac{9}{16}(3)\right\} + \frac{1}{4}(3) = \frac{18}{8} = 2.5.$$

Thus, $E[Av(P_2)] > E[Av(P_1)]$ for this example. We intend to show that, in general, $E[Av(P_i)]$ converges to the maximum of the objective function.

3 Selection implies convergence

Figure 2 (b) plots the estimator $\hat{Av}(P_i)$ (normalized by $\max_{x \in \mathcal{S}_1} F_1(x) = 255^2$). $\hat{Av}(P_i)$ is the average of 49 values of $Av(P_i)$ corresponding to 49 independent executions of GA when optimizing $F_1(x) = 255^2 - x^2$ with the parameters as in Figure 2 (a). The estimator $\hat{Av}(P_i)$ is shown with 95% confidence intervals and its values have been interpolated (using least squares) by the theoretical learning curve of the form $\phi(i) = 1 - (1-\rho)\theta^{i-1}/(1+\log i)$; where ρ is the initial performance and θ is the learning rate. For this experiment $\phi_1(i) = 1 - (1 - 0.661184)(0.114)^{i-1}/(1 + \log i)$.

For all functions F and all encoding schemes $\mathcal{S}$, the sampled average $\hat{Av}(P_i)$ is an estimator (obtained in practice when using a GA) of $E[Av(P_i)]$, and the fundamental question is if $E[Av(P_i)]$ converges to $\max_{x \in \mathcal{S}} F(x)$ when the number of generations goes to infinity.

We answer this question positively (for a theoretically grwoing population size), even when selection is the only genetic operator.

Theorem 3.1 *Let GA be a genetic algorithm maximizing a positive function F on an encoding scheme $\mathcal{S}$ with $Prob_{cross} = Prob_{mutation} = 0$ and a population of size $n > 1$. Then, $\lim_{i \to \infty} E[Av(P_i)] = \max_{x \in \mathcal{S}} F(x)$, as $n \to \infty$.*

The idea of the proof is as follows.
1.- Observe that the first population P_1 is made of a uniform random sample of points in $\mathcal{S}$. Thus, $Av(P_1)$ is an estimator for the expected value of the function F on the domain $\mathcal{S}$ (where each point in $\mathcal{S}$ has probability $1/\|\mathcal{S}\|$). This suggests that $E[Av(P_1)]$ is the expected value of F on $\mathcal{S}$. We will confirm this in Lemma 3.2 later on.
2.- The idea in the previous step is generalized for later generations ($i > 1$) to obtain an expression for $E[Av(P_i)]$ in terms of the expected values of powers of the objective function.
3.- We evaluate $\lim_{i \to \infty} E[Av(P_i)]$ using the expression for $E[Av(P_i)]$ in Lemma 3.6 below.

In preparation for the first lemma, we now introduce the definitions of the expected values (and the variance) of the powers of the objective function F on $\mathcal{S}$.

Definition 3.1 *Let $\mathcal{S}$ be the space of encodings, F a positive function to be maximized on $\mathcal{S}$. The expected value of F on $\mathcal{S}$ is the expected value of $F(c)$ when c is drawn from $\mathcal{S}$ with a uniform distribution; that is, $E_{\mathcal{S}}[F] = \sum_{c \in \mathcal{S}} Prob[c]F(c) = \sum_{c \in \mathcal{S}} F(c)/\|\mathcal{S}\|$. The expected* *value of the i-th power of F is the expected value of the function $F^i(x) = (F(x))^i$ when c is drawn randomly and uniformly from $\mathcal{S}$; that is,*

$$E_{\mathcal{S}}[F^i] = \sum_{c\in\mathcal{S}} Prob[c] \underbrace{F(c)F(c)\ldots F(c)}_{i\ times} = \frac{1}{\|\mathcal{S}\|}\sum_{c\in\mathcal{S}}(F(c))^i.$$

The variance of F^i on $\mathcal{S}$ is the expected value of $(F^i(c) - E_{\mathcal{S}}[F^i])^2$ when c is drawn randomly and uniformly from $\mathcal{S}$; that is,

$$VAR_{\mathcal{S}}[F^i] = \sum_{c\in\mathcal{S}} Prob[c](F^i(c) - E_{\mathcal{S}}[F^i])^2 = \frac{1}{\|\mathcal{S}\|}\sum_{c\in\mathcal{S}}(F^i(c) - E_{\mathcal{S}}[F^i])^2.$$

We will handle each step of the proof of Theorem 3.1 in one corresponding subsection below.

3.1 The sample mean

Because the genetic algorithm samples each chromosome in the population independently of the other chromosomes, the following result can be derived directly from the fact that the sample mean of a random sample equals the expected value of the distribution [19]. However, we present a direct proof here to clarify the notation.

Lemma 3.2 *Let GA be maximizing F under the encoding $\mathcal{S}$. Then, $E[Av(P_1)] = E_{\mathcal{S}}[F]$.*

Proof: Because the number of possible populations P_1 times the population size is the total of chromosomes occurrences in all possible P_1 we have the following relation.

$$n \times \sharp \text{ of possible } P_1 = \sum_{c\in\mathcal{S}} \begin{matrix}\sharp \text{ of occurrences of } c\\ \text{among all possible } P_1\end{matrix} = \|\mathcal{S}\| \begin{pmatrix}\sharp \text{ of occurrences of } c\\ \text{among all possible } P_1\end{pmatrix}.$$

Therefore, for a fixed chromosome $c \in \mathcal{S}$,

$$\begin{matrix}\sharp \text{ of occurrences of } c\\ \text{among all possible } P_1\end{matrix} = \frac{(\sharp \text{ of possible } P_1) \times n}{\|\mathcal{S}\|} = \frac{\|\mathcal{S}\|^n n}{\|\mathcal{S}\|} = \|\mathcal{S}\|^{n-1} n.$$

Since each $c \in \mathcal{S}$ has probability $1/\|\mathcal{S}\|$ of appearing as the j-th element of the population P_1, we have the following derivation.

$$\begin{aligned}
E[Av(P_1)] &= \sum_{\text{all possible } P_1} Prob[P_1 = \langle c_1, \ldots, c_n\rangle]\frac{1}{n}\sum_{j=1}^{n} F(c_j)\\
&= \frac{1}{n}\frac{1}{\|\mathcal{S}\|^n} \sum_{\text{all possible } P_1=\langle c_1,\ldots,c_n\rangle} \sum_{j=1}^{n} F(c_j)\\
&= \frac{1}{n}\frac{1}{\|\mathcal{S}\|^n} \sum_{c\in\mathcal{S}} \begin{pmatrix}\sharp \text{ of occurrences of } c \text{ in}\\ \text{all possible } P_1\end{pmatrix} F(c)\\
&= \frac{1}{\|\mathcal{S}\|}\sum_{c\in\mathcal{S}} F(c) = E_{\mathcal{S}}[F].
\end{aligned}$$

And the proof of the lemma is complete. □

For example, for the function $F_1(x) = 255^2 - x^2$ and the encoding space $\mathcal{S}_1$ of integers in $[-255, 255]$, we have that:

$$E_{\mathcal{S}_1}[F_1] = \frac{1}{511}\sum_{i=-255}^{255} 255^2 - i^2 = \frac{255^2}{511} + \frac{2}{511}\left[255^3 - \frac{255(256)(255+1/2)}{3}\right] = 43,265.$$

Therefore, $E_{\mathcal{S}_1}[F_1]/max_{x\in\mathcal{S}_1}F_1(x)$ is 0.665359. For the 49 independent executions of the GA illustrated in Figure 2, we have $\hat{Av}(P_1) = 0.661184$ with a 95% confidence interval of ± 0.0213. We can say that the initial population of the GA for F_1 is expected to be 34% away from maximizing F_1.

3.2 Later generations

We now provide the analogous result to Lemma 3.2 for the second generation.

Lemma 3.3 *Let GA be a genetic algorithm maximizing a positive function F for the encoding $\mathcal{S}$ and $n > 1$. Then, $E[Av(P_2)] \approx E_{\mathcal{S}}[F^2]/E_{\mathcal{S}}[F]$, with the approximation being exact as $n \to \infty$.*

For our example with $F_1(x) = 255^2 - x^2$, this result means that the expected fitness of the second population is given by:

$$E[Av(P_2)] = \frac{E_{\mathcal{S}}[F_1^2]}{E_{\mathcal{S}}[F_1]} = \frac{\sum_{i=-255}^{255}(255^2 - x^2)^2}{(511)(43,256)} \approx \frac{4}{5} = 0.8. \tag{1}$$

Although in this example the arithmetic manipulation can be performed here to obtain an exact value for $E[Av(P_2)]$, we will show in Section 4 how to obtain a close approximate for $E[Av(P_i)]$ when maximizing F_1, for all $i > 2$. For the experiment of Figure 2, we have $\hat{Av}(P_2) = 0.8053$ with a 95% confidence interval of ± 0.0214.

To prove Lemma 3.3, we require to estimate the probability of a chromosome occurring in the second population.

Lemma 3.4 *Let GA be a genetic algorithm maximizing a positive function F for the encoding $\mathcal{S}$ and $n > 1$. Let $c_0 \in \mathcal{S}$ be any chromosome. Then,*

$$Prob[c_0 \textit{ is } j\textit{-th chromosome in } P_2] \approx \frac{F(c_0)}{\|\mathcal{S}\|E_{\mathcal{S}}[F]} + \tag{2}$$

$$+\frac{F(c_0)}{n}\left\{\frac{-COV_{\mathcal{S}}[\sharp \textit{ of occurrences of } c_0 \textit{ in } P_1, Av(P_1)]}{(E_{\mathcal{S}}[F])^2} + \frac{VAR_{\mathcal{S}}[F]}{\|\mathcal{S}\|(E_{\mathcal{S}}[F])^3}\right\}, \tag{3}$$

where $COV[X,Y]$ is the covariance of random variables X and Y.

Proof: Let n be the size of all populations and c_0 any chromosome in $\mathcal{S}$. Since the first population is generated uniformly we have

$$Prob[c_0 \text{ is } j\text{-th chromosome in } P_1] = 1/\|\mathcal{S}\|.$$

By Lemma 3.2, we also have $E[Av(P_1)] = E_{\mathcal{S}}[F]$. Note that the genetic algorithm selects c_0 as the j-th chromosome of the second generation with probability

$$F(c_0) \times (\sharp \text{ of occurrences of } c_0 \text{ in } P_1)/\sum_{c \in P_1} F(c).$$

Since each slot in the second generation is selected independently, we have

$$\begin{aligned} & Prob[c_0 \text{ is } j\text{-th chromosome in } P_2] \\ = & \sum_{\text{possible } P_1} \frac{Prob[P_1]F(c_0)(\sharp \text{ of occurrences of } c_0 \text{ in } P_1)}{\sum_{c\in P_1} F(c)} \\ = & \frac{F(c_0)}{n} \sum_{\text{possible } P_1} Prob[P_1]\frac{(\sharp \text{ of occurrences of } c_0 \text{ in } P_1)}{Av(P_1)} \\ = & \frac{F(c_0)}{n} E\left[(\sharp \text{ of occurrences of } c_0 \text{ in } P_1)/Av(P_1)\right]. \end{aligned}$$

Now, for two random variables X and Y, the value of $E[X/Y]$ can be approximated as

$$E[X/Y] \approx \frac{E[X]}{E[Y]} - \frac{COV[X,Y]}{(E[Y])^2} + \frac{E[X]VAR[Y]}{(E[Y])^3},$$

with an error of terms of order higher that 2 [19]. Since $E[\sharp \text{ of occurrences of } c_0 \text{ in } P_1] = n/\|\mathcal{S}\|$, we have

$$\begin{aligned} & Prob[c_0 \text{ is } j\text{-th chromosome in } P_2] \\ \approx & \frac{F(c_0)}{\|\mathcal{S}\|E[Av(P_1)]} \\ & -\frac{F(c_0)}{n}\frac{COV[\sharp \text{ of occurrences of } c_0 \text{ in } P_1], Av(P_1)]}{(E[Av(P_1)])^2} \\ & +\frac{F(c_0)}{\|\mathcal{S}\|}\frac{VAR[Av(P_1)]}{(E[Av(P_1)])^3}. \end{aligned}$$

Now, using Lemma 3.2 and the fact that $VAR[Av(P_1)]$ is the sample variance, and thus $VAR[Av(P_1)] = VAR_{\mathcal{S}}[F]/n$ [19], the claim of the lemma follows. □

We illustrate this result by estimating $Prob[c_0 \text{ is } j\text{-th chromosome in } P_2]$ for a genetic algorithm working to maximize $F_0(x)$ in $\mathcal{S}_0$. Since $E_{\mathcal{S}_0}[F_0] = 2$ we have that, for a $c_0 < 0$, the first term in the right hand side of Equation (2)-(3) is

$$\frac{F_0(c_0)}{\|\mathcal{S}_0\|E_{\mathcal{S}_0}[F_0]} = \frac{F_0(c_0)}{(2m)(2)} = \frac{1}{4m}.$$

Now, from Figure 3, we see that

$$\begin{aligned}
VAR[Av(P_1)] &= \sum_{\text{possible } P_1} Prob[P_1](Av(P_1) - E[Av(P_1)])^2 \\
&= \sum_{i=1}^{4} \sum_{P_1 \in \text{Event}_i} Prob[P_1](Av(P_1) - E[Av(P_1)])^2 \\
&= \sum_{P_1 \in \text{Event}_1} \frac{1}{4}(1-2)^2 + \sum_{P_1 \in \text{Event}_2 \cup \text{Event}_3} \frac{2}{4}(2-2)^2 + \sum_{P_1 \in \text{Event}_4} \frac{1}{4}(3-2)^2 \\
&= 1/2.
\end{aligned}$$

Thus, the third term in (2)-(3) is

$$\frac{F_0(c_0)}{\|\mathcal{S}_1\|(E[F_0])^3} VAR[Av(P_1)] = \frac{1}{32m}.$$

Now,

$$\begin{aligned}
&COV[Av(P_1), \sharp \text{ of occurrences of } c_0 \text{ in } P_1] \\
&= \sum_{\text{possible } P_1} Prob[P_1](Av[P_1] - E[Av(P_1)]) \left(\sharp \text{ of occurrences of } c_0 \text{ in } P_1 - \frac{n}{\|\mathcal{S}\|}\right).
\end{aligned}$$

When $n = 2$, algebraic manipulations show that, for this example,

$$COV[Av(P_1), \sharp \text{ of occurrences of } c_0 \text{ in } P_1] = \frac{-1}{2m}.$$

Thus, the second term in the right hand side of Equation (2)-(3) is

$$-\frac{F_0(c_0)}{n} \frac{COV[Av(P_1), \sharp \text{ of occurrences of } c_0 \text{ in } P_1\ ]}{(E[F_0])^2} = \frac{1}{16m}.$$

Finally, if $c_0 < 0$ Lemma 3.4 gives

$$Prob[c_0 \text{ is the } j\text{-th chromosome of } P_2] \approx \frac{1}{4m} + \frac{1}{16m} + \frac{1}{32m} = \frac{11}{32m}.$$

In fact, from Figure 3, it is not hard to see that

$$\sum_{c_0=-m}^{-1} Prob[c_0 \text{ is the } j\text{-th chromosome of } P_2] = 3/8;$$

thus, we can compute exactly $Prob[c_0 \text{ is the } j\text{-th chromosome of } P_2] = 3/8m$, for $c_0 < 0$. That is, Lemma 3.4 is an approximation within $1/32m$ for the example function F_0 even for such a small n as 2. Therefore, for small m (say $m = 50$), this is extremely accurate.

We observe that Lemma 3.4 explains why genetic algorithms are more robust with larger population sizes and also why functions with high variance (usually regarded as difficult for genetic algorithms [6, 22]) are a challenge to the genetic algorithm's behavior. Because the covariance of only one chromosome to $Av(P_1)$ is usually small and the powers of $E_{\mathcal{S}}[F]$ are large, Lemma 3.4 can be simplified to the following result.

Lemma 3.5 *Let GA be a genetic algorithm maximizing a positive function F for the encoding $\mathcal{S}$ and $n > 2$. Let $c_0 \in \mathcal{S}$ be any chromosome. Then,*

$$Prob[c_0 \textit{ is } j\textit{-th chromosome in } P_2] \approx \frac{F(c_0)}{\|\mathcal{S}\| E_{\mathcal{S}}[F]}.$$

Here, $Prob[c_0$ is j-th chromosome in $P_2]$ means the probability of finding c_0 as the j-th chromosome of a second generation of an execution of a genetic algorithm. This should be distinguished from estimates made for the proportions of appearances of c_0 in the i-th generation *given the proportion of appearances* of c_0 in the $(i-1)$-th generation. Such estimates are conditional probabilities computed in the derivation of the Schema Theorem [3] or in the derivation of the behavior of a Markov Chain on a one-bit string (even with selection only) [14].

We now prove Lemma 3.3.

Proof: Using Lemma 3.5 we have the following derivation.

$$\begin{aligned} E[Av(P_2)] &= \frac{1}{n}\sum_{c\in\mathcal{S}} E\left[\sharp \text{ of occurrences of } c \text{ in } P_2\right] F(c) \\ &= \frac{1}{n}\sum_{c\in\mathcal{S}} Prob[c \text{ is } j\text{-th chromosome in } P_2]\; n\; F(c) \\ &\approx \sum_{c\in\mathcal{S}} \frac{F(c)}{\|\mathcal{S}\| E_{\mathcal{S}}[F]} F(c) = \frac{\sum_{c\in\mathcal{S}} F^2(c)}{\|\mathcal{S}\|}\frac{1}{E_{\mathcal{S}}[F]} = E_{\mathcal{S}}[F^2]/E_{\mathcal{S}}[F]. \end{aligned}$$

□

We are now ready to extend our results regarding the expected value of $Av(P_i)$ to generations later than 2.

Lemma 3.6 *Let GA be a genetic algorithm using only selection to maximize a positive function F for the encoding $\mathcal{S}$. Then, for all $i > 1$, $E[Av(P_i)] \approx E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$.*

Proof: Let n be the size of all populations. We prove Lemma 3.6 by proving by induction on i the following helping claim

$$Prob[c \text{ is } j\text{-th chromosome in } P_i] \approx \frac{F^{i-1}(c)}{\|\mathcal{S}\| E_{\mathcal{S}}[F^{i-1}]} \quad (\text{for } i > 1, 1 \le j \le n \text{ and all } c \in \mathcal{S}),$$

along with the claim of the lemma. Recall expectations are over all executions of the GA.

Basis of Induction: Let $i = 2$. The helping claim for $i = 2$ is Lemma 3.5. The case $i = 2$ is Lemma 3.3.

Induction step: We will almost replicate the derivations of the basis of the induction. Because the genetic algorithm selects independently each slot in the next generation we have

$$Prob[c_0 \text{ is } j\text{-th chromosome in } P_i]$$

$$
\begin{aligned}
&= \sum_{\text{possible } P_{i-1}} \frac{Prob[P_{i-1}]F(c_0)(\sharp \text{ of occurrences of } c_0 \text{ in } P_{i-1})}{\sum_{c\in P_{i-1}} F(c)} \\
&= \frac{F(c_0)}{n} \sum_{\text{possible } P_{i-1}} Prob[P_{i-1}] \frac{(\sharp \text{ of occurrences of } c_0 \text{ in } P_{i-1})}{Av(P_{i-1})} \\
&= \frac{F(c_0)}{n} E\left[(\sharp \text{ of occurrences of } c_0 \text{ in } P_{i-1})/Av(P_{i-1})\right].
\end{aligned}
$$

Now, we again approximate the expected value of a ratio of random variables by the ratio of expected values with an error of terms of order higher that 2 [19]. That is,

$$
E[X/Y] \approx \frac{E[X]}{E[Y]} - \frac{COV[X,Y]}{(E[Y])^2} + \frac{E[X]VAR[Y]}{(E[Y])^3}.
$$

Moreover, the approximation to the first term is accurate when n is of moderate size. Therefore, with the induction hypothesis for $i-1$ (for the helping claim and the lemma), we obtain:

$$
\begin{aligned}
Prob[c \text{ is } j\text{-th chromosome in } P_i] &\approx \frac{F(c)E\left[\sharp \text{ of occurrences of } c \text{ in } P_{i-1}\ \right]}{nE[Av(P_{i-1})]} \\
&= \frac{F(c)}{n} \frac{Prob[c \text{ is } j\text{-th chromosome in } P_{i-1}]\ n}{(E_{\mathcal{S}}[F^{i-1}]/E_{\mathcal{S}}[F^{i-2}])} \\
&= \frac{F(c)E_{\mathcal{S}}[F^{i-2}]}{E_{\mathcal{S}}[F^{i-1}]} \frac{F^{i-2}(c)}{\|\mathcal{S}\|E_{\mathcal{S}}[F^{i-2}]} = \frac{F^{i-1}(c)}{\|\mathcal{S}\|E_{\mathcal{S}}[F^{i-1}]}.
\end{aligned}
$$

This is the helping claim for i, which is now used to prove the lemma for i.

$$
\begin{aligned}
E[Av(P_i)] &= \frac{1}{n}\sum_{c\in\mathcal{S}} E\left[\sharp \text{ of occurrences of } c \text{ in } P_i\right]\ F(c) \\
&= \frac{1}{n}\sum_{c\in\mathcal{S}} Prob[c \text{ is } j\text{-th chromosome in } P_i]\ n\ F(c) \\
&\approx \sum_{c\in\mathcal{S}} \frac{F^{i-1}(c)}{\|\mathcal{S}\|E_{\mathcal{S}}[F^{i-1}]} F(c) = \frac{\sum_{c\in\mathcal{S}} F^i(c)}{\|\mathcal{S}\|} \frac{1}{E_{\mathcal{S}}[F^{i-1}]} = E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}].
\end{aligned}
$$

This completes the induction step and the proof. □

3.3 The proof of convergence

Proof of Theorem 3.1: Consider $E[Av(P_i)]$, with $i > 1$. By Lemma 3.6 and Definition 3.1 we have the following derivation.

$$
E[Av(P_i)] = \frac{E_{\mathcal{S}}[F^i]}{E_{\mathcal{S}}[F^{i-1}]} = \frac{\sum_{c\in\mathcal{S}} \frac{1}{\|\mathcal{S}\|} F^i(c)}{\sum_{c\in\mathcal{S}} \frac{1}{\|\mathcal{S}\|} F^{i-1}(c)} = \frac{\sum_{c\in\mathcal{S}} F^i(c)}{\sum_{c\in\mathcal{S}} F^{i-1}(c)}.
$$

Let $\mathcal{M} = \{c_i \in \mathcal{S} \mid F(c_i) \geq F(c_j),\ \forall c_j \in \mathcal{S}\}$; that is, $\mathcal{M} \subseteq \mathcal{S}$ is the set of points in the encoding space where F is globally maximized (F main have more than one global optimum or even be constant resulting in all points being global optima; however, in this last case, the theorem follows trivially). Let $c_0 \in \mathcal{M}$ be one point where F is maximized and assume $\mathcal{S}\backslash\mathcal{M}$ is not empty. We now have the following derivation.

$$\begin{aligned} E[Av(P_i)] &= \frac{\sum_{c\in\mathcal{S}} F^i(c)}{\sum_{c\in\mathcal{S}} F^{i-1}(c)} = \frac{\sum_{c\in\mathcal{M}} F^i(c) + \sum_{c\in\mathcal{S}\backslash\mathcal{M}} F^i(c)}{\sum_{c\in\mathcal{M}} F^{i-1}(c) + \sum_{c\in\mathcal{S}\backslash\mathcal{M}} F^{i-1}(c)} \\ = \frac{F^i(c_0)\|\mathcal{M}\| + \sum_{c\in\mathcal{S}\backslash\mathcal{M}} F^i(c)}{F^{i-1}(c_0)\|\mathcal{M}\| + \sum_{c\in\mathcal{S}\backslash\mathcal{M}} F^{i-1}(c)} &= \frac{F(c_0)\|\mathcal{M}\| + \sum_{c\in\mathcal{S}\backslash\mathcal{M}} F(c)[F(c)/F(c_0)]^{i-1}}{\|\mathcal{M}\| + \sum_{c\in\mathcal{S}\backslash\mathcal{M}}[F(c)/F(c_0)]^{i-1}} \end{aligned}$$

Observe that $F(c)/F(c_0) < 1$, for all $c \in \mathcal{S}\backslash\mathcal{M}$; thus, $\lim_{i\to\infty}[F(c)/F(c_0)]^{i-1} = 0$. Now, taking the limit when $i \to \infty$, we have that $\lim_{i\to\infty} E[Av(P_i)] = \|\mathcal{M}\|F(c_0)/\|\mathcal{M}\|$ which is $F(c_0)$, the maximum of the function, as required. □

4 Practical implications

Theorem 3.1 contributes towards the theoretical understanding of convergence for genetic algorithms. However, from the practitioner's point of view, perhaps Lemma 3.6 is the main result and Theorem 3.1 is a corollary. In this section we illustrate this point.

Lemma 3.6 allows us to predict $E[Av(P_i)]$ as $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$. In particular, if for some value i_0, we have that $E_{\mathcal{S}}[F^{i_0+1}]/E_{\mathcal{S}}[F^{i_0}]$ is not significantly larger than $E_{\mathcal{S}}[F^{i_0}]/E_{\mathcal{S}}[F^{i_0-1}]$, then there would be little improvement in running the genetic algorithm past i_0 generations. Thus, practitioners can search for the appropriate value of i_0 by comparing $E_{\mathcal{S}}[F^{i+1}]/E_{\mathcal{S}}[F^i]$ with $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$, for $i = 1, 2, 4, \ldots, 2^p, \ldots$ until no improvement to the desired precision occurs. Of course, $E_{\mathcal{S}}[F^i]$ can not be computed exactly in practice, but it can be sampled. In a sense, the GA is doing so every generation.

Let us illustrate this with our running example; the maximization of $F_1 = 255^2 - x^2$ in $\mathcal{S}_1$. Because we know the objective function we will not estimate $E_{\mathcal{S}_1}[F_1^i]$ by sampling. We will show an analytical and more precise approximation. Since $E_{\mathcal{S}_1}[F_1^i] = \sum_{c\in\mathcal{S}_1}(F_1(c))^i/\|\mathcal{S}_1\|$ and $\mathcal{S}_1$ is a discretization of the domain $\mathcal{D}_1 = [-255, 255]$, we have that $E_{\mathcal{S}_1}[F_1^i] \approx \int_{\mathcal{D}_1}(F_1(x))^i dx/measure(\mathcal{D}_1)$. Thus, since $measure(\mathcal{D}_1) = 2(255)$ we obtain:

$$\begin{aligned} 2(255)E_{\mathcal{S}_1}[F_1^i] &\approx \int_{-255}^{255}\left[255^2 - x^2\right]^i dx = \sum_{k=0}^{i}\binom{i}{k} 255^{2k}\int_{-255}^{255}(-1)^{i-k}x^{2(i-k)}dx \\ &= \sum_{k=0}^{i}\binom{i}{k}\frac{255^{2k}(-1)^{i-k}}{2(i-k)+1}\left[2(255)^{2(i-k)+1}\right] \\ &= 2(255)^{2i+1}\sum_{k=0}^{i}\binom{i}{k}\frac{(-1)^k}{2k+1}. \end{aligned}$$

Now, since $\max_{x\in\mathcal{D}_1} F_1(x) = 255^2$ and using identities for summations involving combinatorial coefficients, we have that $E[Av(P_i)]/\max_{x\in\mathcal{D}_1} F_1(x) = E_{\mathcal{S}_1}[F_1^i]/(E_{\mathcal{S}_1}[F_1^{i-1}]255^2) =$

	Prediction by Lemma 3.6	GA selection only (no crossover nor mutation)		Classical GA ($Prob_{cross} = 0.6$ and $Prob_{mutation} = 0.001$)	
i	$2i/(2i+1)$	observed	95% interval	observed	95% interval
1	66.66 %	66.63 %	2.18	66.11 %	2.13
2	80.00 %	78.60 %	2.00	80.52 %	2.14
3	85.70 %	84.92 %	1.64	85.80 %	1.67
4	88.88 %	88.79 %	1.22	88.19 %	1.38
5	90.90 %	90.52 %	1.17	90.48 %	1.29
8	94.11 %	94.02 %	1.12	93.67 %	1.26
9	94.73 %	94.64 %	1.22	94.05 %	1.18
10	95.23 %	95.17 %	1.13	94.49 %	1.12

Table 1: Predicted $100E[Av(P_i)]/\max_{x\in\mathcal{D}_1} F_1(x)$ and observed $100\hat{Av}(P_i)/\max_{x\in\mathcal{D}_1} F_1(x)$.

Function	Search space and encoding per variable
$f_1(x_1,x_2,x_3) = 3a^2 - \sum_{i=1}^{3} x_i^2$	$x_i \in [-a,a],\ i=1,2,3$; 10 bits
$f_2(x_1,x_2) = 1769 - 100(x_1^2 - x_2^2)^2 - (1-x_1)^2$	$x_i \in [-2.048, 2.048],\ i=1,2$; 12 bits
$f_3(\vec{x}) = 25 + \sum_{i=1}^{5} \lfloor x_i \rfloor$	$x_i \in [-5.12, 5, 12],\ i=1,\ldots,5$; 10 bits
$f_4(\vec{x}) = 1249 - \sum_{i=1}^{30} ix_i^4 + Gaus((0,1)$	$x_i \in [-1.28, 1.28],\ i=1,\ldots,30$; 8 bits
$f_5(x_1,x_2) = 0.002 + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}$ where $a_{1j} = 16[(j \bmod 5) - 3]$ and $a_{2j} = 16[\lfloor (j-1)/5 \rfloor - 3]$	$x_i \in [-65.535, 65.353],\ i=1,2$; 17 bits

Table 2: De Jong's testbed of objective functions.

$2i/(2i+1)$. Table 1 shows the comparison of the theoretically predicted accuracy (as a percentage) and 49 executions of the classical genetic algorithm (with crossover and mutation as in Figure 2) and the GA with selection only. Thus, for the expected fitness of the population to be within p percent of the optimum, we must have $2i/(2i+1) > (100-p)/100$ or $i > 50/p - 1/2$. For example, the average fitness is within 95% of the optimum for F_1 after the 10-th generation, within 96% after the 13-th generation and within 97% after the 17-th generation.

To further illustrate this point, we have carried out a set of extensive simulations on several objective functions. Here, we present results on a testbed of functions (and search spaces) proposed by De Jong [6] and extensively used among the genetic algorithms community (refer to Table 2). This testbed includes smooth functions (f_1), nonconvex functions(f_2), discontinuos functions (f_3), stochastic functions (f_4), and multimodal (f_2 and f_5). The functions in Table 2 are non-negative and the goal is to find their global maximum.

The results are presented as the ratio of $E[Av(P_i)]$ to the maximum of the objective function because:

- this measures the approximation obtained by the algorithm in relative terms and
- in light of our results, this evaluation of the search heuristic has the desirable property that is invariant to positive scaling [1]. Namely, for a constant $c > 0$,

$$\frac{E[Av(P_i)] \text{ on } cf}{\max_{\mathcal{S}} cf} = \frac{E_{\mathcal{S}}[(cf)^i]}{cE_{\mathcal{S}}[(cf)^{i-1}] \max_{\mathcal{S}} f} = \frac{E_{\mathcal{S}}[f^i]}{E_{\mathcal{S}}[f^{i-1}] \max_{\mathcal{S}} f} = \frac{E[Av(P_i)] \text{ on } f}{\max_{\mathcal{S}} f}.$$

The value $E[Av(P_i)]/\max_{\mathcal{S}} f$ was computed by running the genetic algorithm with a population size $n = 100$ and computing an estimate over 49 independent executions. This resulted in 95% confidence intervals with width approximately ± 0.002 (except for f_5 which is such a difficult function that $n = 1000$ was used and still the 95% confidence interval was approximately ± 0.04 for generations later than the first). The value $E_{\mathcal{S}}[f^i]/(E_{\mathcal{S}}[f^{i-1}] \max_{\mathcal{S}} f)$ depends on the objective function (and the search space) only. It was obtained by sampling 1,000 points uniformly in the search space, evaluating f^i and f^{i-1} in each point and computing the corresponding average over 49 samplings. This resulted in 95% confidence intervals with width approximately ± 0.002 (except, again, for f_5 which is such a difficult function that 10,000 points were sampled).

The results are presented in Table 3. We see that Lemma 3.6 is a magnificent predictor of the performance of the genetic algorithms across the testbed of functions. Thus, $E_{\mathcal{S}}[f^i]/(E_{\mathcal{S}}[f^{i-1}] \max_{\mathcal{S}} f)$ is a reliable indicator of how hard the objective functions is to optimize by the genetic algorithm. It is an indicator of a reasonable rate of convergence of any realistic genetic algorithm that involves combination operators (mutation and crossover). However, the actual rate of convergence on a practical genetic algorithm may be slower. For example, the encoding scheme may have placed building blocks in positions that are not adequately exploited by the particular crossover operator (thus, the different convergence rates for different crossover operators and/or different encoding schemes [29, 10].) Moreover, for a practical genetic algorithm, the ratio $E_{\mathcal{S}}[f^i]/(E_{\mathcal{S}}[f^{i-1}] \max_{\mathcal{S}} f)$ is an accurate indicator on relative terms, specially for early generations. For later generations ($i > 50$), it would be a poor indicator in absolute terms. This is so because when a genetic algorithm with mutation and combination operators is near convergence, it looses population diversity and parallels random search [11], However, early convergence and loss of population diversity are handled by proposals like *dynamic parameter encoding* [28] which restart the genetic algorithm with a local encoding, and thus, the indicator would be reliable again.

Function f_1 is the tri-dimensional version of our working example F_1. We have specified the bounds of the search space in terms of the constant a because the behavior is the same for different values of a. The function is optimized to 80% of the maximum value within 5 generations, but then the rate slows down as the function is so smooth near the optimum that selective pressure is subsummed by stochastic error. Function f_2 has two global optima at $(-1, 1)$ and $(1, 1)$ which are relatively not much higher than other points in the planes $x_1 = x_2$ and $x_1 = -x_2$. However, points in this regions are much higher than everywhere else

Objective function		Prediction by Lemma 3.6	GA selection only	Classical GA
	i	$E_{\mathcal{S}}[f^i]/(E_{\mathcal{S}}[f^{i-1}]\max_{\mathcal{S}} f)$	$E[Av(P_i)]/max_{\mathcal{S}}f$	$E[Av(P_i)]/max_{\mathcal{S}}f$
f_1	1	0.6669	0.6668	0.6683
	2	0.7113	0.7104	0.7114
	3	0.7463	0.7441	0.7424
	4	0.7747	0.7706	0.7678
	5	0.7984	0.7948	0.7891
f_2	1	0.8219	0.8158	0.8208
	2	0.8810	0.8811	0.8670
	3	0.9101	0.9051	0.8862
	4	0.9276	0.9250	0.8957
	5	0.9392	0.9386	0.9021
f_3	1	0.5002	0.4998	0.4991
	2	0.5262	0.5258	0.5290
	3	0.5498	0.5503	0.5497
	4	0.5711	0.5704	0.5715
	5	0.5908	0.5874	0.5907
f_4	1	0.7997	0.7993	0.8002
	2	0.8036	0.8047	0.8041
	3	0.8075	0.8072	0.8079
	4	0.8111	0.8091	0.8111
	5	0.8147	0.8134	0.8146
f_5	1	0.0036	0.0036	0.0037
	2	0.0865	0.0723	0.0725
	3	0.2899	0.2538	0.2883
	4	0.3359	0.3050	0.4132
	5	0.3512	0.3219	0.5456

Table 3: Empirical comparison of behavior from 49 executions of genetic algorithm and prediction provided by Lemma 3.6. Classical GA has $Prob_{cross} = 0.6$ and $Prob_{mutation} = 0.001$.

and it is easy to find high values of the function. Within 3 generations, a value 90% away from optimum is achieved. Function f_3 has many plateaus and the high values of the function are a small proportion of the search space. This reflects on a very slow convergence rate towards the optimum. Although f_4 has noise, this is very small and results in an easy function for genetic algorithms. Finally, f_5 is a very difficult function in that it has 25 local maxima (and only one of which is the global optimum). This local maxima form a "porcupine" [1] landscape with a very small "trap" region [1]. That is, each local maxima is a very high peak (relative to the values in the vast majority of the search space) with a very small base. Thus, local maxima are hard to find because of the size of the search space, and even if found, the algorithm may very likely find a local optimum. Thus, the rate of convergence is very slow. However, Lemma 3.6 continues to be a strong predictor. Moreover, we tested the prediction in the domain $[-32.0, -31.0] \times [-32.0, -31.0]$ where f_5 has just one peak and has a shape similar to one hill and the prediction obtained was accurate and also the rate of convergence similar to the one shown for f_1. Thus, this illustrates also the importance on the encoding and size of the search space across practical applications.

Another practical implication of Lemma 3.6 is the explanation (and possible tunning) of procedures like scaling, normalizing and windowing [4] of the fitness values of the population. The idea behind this procedures is that when all elements in the population have similar fitness values, these values should be differentiated more to have "selective pressure" and allow the GA to advance. When fitness values are far apart, they should be leveled to avoid lossing weak chromosomes totally or to avoid dominance by the strongest chromosome leading to early convergence to local optima. We have seen that, the relative approximation of the optimum is not affected by scaling. However, narrowing the search space (by adjusting the encoding accordingly) to regions detected as promising by a previous run for detailed exploration by a subsequent run is very promising. That is, $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$ can vary significantly for different $\mathcal{S}$ and same objective function. This observation supports the method of *dynamic parameter encoding* [28] (DPE) where the range of real variables in the domain is resized to a smaller interval of the real line. In this maner, DPE can zoom and adjust $\mathcal{S}$ dynamically, re-starting the genetic search with a finner encoding. When fitness values are similar, the ratio $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$ is not very large, while when fitness values are extreme the ratio $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$ is large. Thus, another alternative is to normalize the objective function to make the ratio $E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]$ a reasonable compromise.

5 Discussion and final remarks

We have demonstrated that (in the limit), in the classical genetic algorithm, the expected fitness of the population converges to the maximum value of the objective function. We have shown that the genetic operator responsible for this is selection (where the probability of survival of a chromosome is proportional to its relative fitness in the current population).

Our result is general and comprehensive. The model of GA does not need to be elitist and the objective function can be any real-valued function of one or more variables. The encoding scheme can be arbitrary too (for example, binary, decimal, or a decision-tree encoded as

LISP's S-expressions). Moreover, the encoding scheme need not be a bijection to the domain. If $c_1, c_2 \in \mathcal{S}$ are strings denoting some individual x, we just think as F defined on $\mathcal{S}$ as $F(c_1) = F(c_2) = F(x)$, and recall that $E_{\mathcal{S}}[F]$ is the expectation of F on $\mathcal{S}$.

Our results indicate that genetic algorithms converge at a rate that is the rate of convergence of the sequence $\{E_{\mathcal{S}}[F^i]/E_{\mathcal{S}}[F^{i-1}]\}_{i\in\mathcal{N}}$ to $\max_{\mathcal{S}} F$. Not much slower, neither much faster. This rate depends on the shape of F and the encoding scheme S, but the rate of convergence is independent of the size n of the population of the genetic algorithm (as long as n has moderate size). Therefore, making n very large does not accelerate the search and, in fact, slows down the process since the genetic algorithm must manage a larger population. However, this also motivates parallel genetic algorithms [22]. These genetic algorithms can be considered as concurrent explorers of the random-decision tree that exchange information regarding their findings. Thus, if computational resources are available to consider large n, we would recommend dividing the population into disjoint subpopulations where genetic algorithms work independently for some generations before migrating chromosomes across subpopulations.

As illustrated by the test function f_5, the rate of convergence depends on the encoding. This suggests another possibility for genetic algorithms, namely, a genetic algorithm that refines or modifies the encoding scheme as the search progresses. In doing so, it adjusts dynamically the rate of convergence. Note that when a search heuristic has a fast rate of convergence, it usually misses the global optimum for a local optimum. On the other hand, a slow rate is unacceptable for cases where extensive exploration is unnecessary. Thus, it seems promising to adjusting the rate as the search explores the space and the gains information on the shape of the objective function with respect to the space.

Altough our approximations benefit form the fact that $\|\mathcal{S}\|$ and powers of $E_{\mathcal{S}}[F^i]$ apears in the denominator of lower order terms and that in practice n will be very small with respect to $\|\mathcal{S}\|$, it is not hard to see that

$$\lim_{i\to\infty} E[Av(P_i)] \leq E_{\mathcal{S}}[\max_{c\in P_1} F(c)].$$

Since $E_{\mathcal{S}}[max_{c\in P_1} F(c)]$ could be much smaller than $\max_{x\in\mathcal{S}} F(x)$ for small n, our theoretical results leave open how small could n be to obtain a resonable convergence behavior.

We have illustrated our results with simulations. A surprising fact for the GA community may be that in the results of Table 1. Using selection only results in a learning rate of 0.1563 when least squares fitting of the learning model $\phi(i) = 1-(1-\rho)\theta^{i-1}/(i+\log i)$. This learning rate is larger that the learning rate when using also crossover and mutation ($\theta = -0.114$ in Figure 2). Thus, in this case, other genetic operators disrupt selection and slow down convergence of the expected fitness of the population. Since practitioners regard the expected fitness of the population as a theoretical issue over finding at least one chromosome close to the global optimum, it remains to theoretically justify the use of other genetic operators.

6 Acknowledgements

We would like to thank Peter Turney for valuable comments on an earlier version of this paper.

References

[1] D.H. Ackley. An empirical study of bit vector function optimization. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 170–204, Los Altos, Ca., 1987. Morgan Kaufmann Publishers.

[2] A. Bertoni and M. Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61:307–324, 1993.

[3] L.B. Booker, D.E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(2):235–282, 1989.

[4] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[5] T.E. Davis and J.C. Principe. A simulated annealing like convergence theory for the simple genetic algorithm. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 174–181, Los Altos, CA., 1991. Morgan Kaufmann Publishers.

[6] K.A. DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, 1975.

[7] K.A. DeJong and W.M. Spears. Learning concept classification rules using genetic algorithms. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 651–656, Sydney, Australia, 1991. Morgan Kaufmann.

[8] K.A. DeJong and W.M. Spears. On the state of evolutionary computation. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 618–623, 2929 Capus Drive, San Mateo CA 94403, 1993. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.

[9] A.E. Eiben, E.H.L. Aarts, and K.M. Van Hee. Global convergence of genetic algorithms: On infinite Markov chain analysis. In H.P. Schwefel and R. Männer, editors, *Proceedinngs of the First International Conference on Parallel Problem Solving from Nature*, pages 4–12. Springer-Verlag, Lecture Notes in Computer Science 496, 1991.

[10] L.J. Eshelman, R.A. Caruana, and J.D. Schaffer. Biases in the crossover landscape. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19, San Mateo, CA., 1989. George Mason University, Morgan Kaufmann Publishers.

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

[12] R. Forsyth. BEAGLE — a Darwinian approach to pattern recognition. *Kybernetes*, 10(3):159–166, 1981.

[13] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Co., Reading, MA, 1989.

[14] D.E. Goldberg and P. Segrest. Finite Markov chain analysis of genetic algorithms. In J.J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8, Hillsdape N.J., 1987. M.I.T., Lawrence Erlbaum Associates.

[15] A. Hojjat. *Machine Learning: neural networks, genetic algorithms and fuzzy systems.* Wiley, New York, 1995.

[16] J. Kingdom. Genetic algorithms: Deception, convergence and starting conditions. Technical report, Dept. of Computer Science, University College, London, UK, 1992.

[17] K. Mehlhorn. *Data Structures and Algorithms, Vol 1: Sorting and Searching.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin/Heidelberg, 1984.

[18] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Heidelberg, Berlin, second, extended edition, 1992.

[19] A.M. Mood, F.A. Graybill, and D.C. Boes. *Introduction to the Theory of Statistics.* Mc-Graw-Hill International Book Company, Tokyo, Japan, third edition, 1974.

[20] H. Mühlenbein, J. Bendisch, and H.M. Voigt. From recombination of genes to the estimation of distributions. ii. continuous parameters. In H.M. Voigt, W. Ebeling, I. Rechenberg, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV. International Conference on Evolutionary Computation*, pages 188–197. Springer-Verlag, Lecture Notes in Computer Science 1141, 1996.

[21] H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions. i. binary parameters. In H.M. Voigt, W. Ebeling, I. Rechenberg, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV. International Conference on Evolutionary Computation*, pages 178–187. Springer-Verlag, Lecture Notes in Computer Science 1141, 1996.

[22] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as a function optimizer. In R. Belew and L. Booker, editors, *Proceedings of the Fourth Int. Conf. on Genetic Algorithms*, pages 271–278, Los Altos, CA., 1991. Morgan Kaufmann Publishers.

[23] J.R. Quinlan. Induction of decision trees. *Machine Learning Journal*, 1:81–106, 1986.

[24] Y. Rabinovich and A. Widgerson. An anlysis of a simple genetic algorithm. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 215–221, Los Altos, CA., 1991. Morgan Kaufmann Publishers.

[25] R. Rohwer, M. Wyne-Jones, and F. Wysotzki. Neural networks. In D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors, *Machine Learning, Neural and Statistical Classification*, pages 84–106. Ellis Horwood Series in Artificial Intelligence, 1994.

[26] G. Rudolph. Convergence analysis on canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1), 1994. Special issue on evolutionary computation.

[27] J.D. Schaffer, D. Whitley, and L.J. Eschelman. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In J.D. Whitley, L.D.; Schaffer, editor, *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, pages 1–37, Baltimore, MD, 1992. IEEE Comput. Soc. Press; Los Alamitos, CA, USA.

[28] N.N. Schraudolph and R.K. Belew. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9:9–21, 1992.

[29] G. Syswerda. Uniform crossover in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA., 1989. George Mason University, Morgan Kaufmann Publishers.

[30] H.M. Voigt, H. Muhlenbein, and D. Sehlierkamp-Voosen. The response to selection equation for skew fitness distributions. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC-96)*, pages 820–825, New York, NY, USA, Nagoya, Japan; 20-22 May 1996. IEEE Neural Network Council, IEEE.