Original paper Soft Computing 4 (2000) 76–80 © Springer-Verlag 2000

Adaptive mutation in genetic algorithms

S. Marsili Libelli, P. Alba

Abstract In Genetic Algorithms mutation probability is usually assigned a constant value, therefore all chromosome have the same likelihood of mutation irrespective of their fitness. It is shown in this paper that making mutation a function of fitness produces a more efficient search. This function is such that the least significant bits are more likely to be mutated in high-fitness chromosomes, thus improving their accuracy, whereas low-fitness chromosomes have an increased probability of mutation, enhancing their role in the search. In this way, the chance of disrupting a high-fitness chromosome is decreased and the exploratory role of low-fitness chromosomes is best exploited. The implications of this new mutation scheme are assessed with the aid of numerical examples.

Key words Genetic algorithms, optimization, numerical methods, search methods

1 Introduction

Genetic Algorithms (GAs) represent a highly efficient search procedure for the determination of global extrema of multivariable functions, imitating the patterns of genetic reproduction in living organisms. GAs work by successive modifications of a collection (often referred to as *population*) of parameter combinations in the search space, using a binary coded representation termed *chromosome*. The initial set of chromosomes evolves through a number of operators (*selection*, *crossover*, *mutation*) so that subsequent generations produce more and more chromosomes in the neighborhood of the optimum, defined through a given figure of merit, or *fitness*. Genetic Algorithms were first introduced by Goldberg (1989) and Holland (1992) and differ from conventional search algorithms in the following aspects:

- GAs work with a *coding* of the parameters, not the parameters themselves;
- GAs search using a population of points, not a single one, thus thoroughly cover the search region and avoid being trapped into local extrema;
- GAs use probabilistic, not deterministic, transition rules to alter the initial population through subsequent generations;
- GAs have a high implicit parallelism, making them numerically very efficient;
- Like all derivative-free search methods, GAs use only point-wise evaluation, thus are very insensitive to discontinuities and irregularities in the fitness function.

S. Marsili Libelli, P. Alba
Department of Systems and Computer Science,
University of Florence, Via S. Marta,
3-50139 Firenze, Italy
Tel./fax: ++39-55-47.96.264
e-mail: marsili@ingfi1.ing.unifi.it

1.1 Main features of the basic GA

The optimization procedure starts from a population of randomly chosen chromosomes and generates successive populations applying the operators of *reproduction, mutation* and *crossover* (Goldberg, 1989; Davis, 1990; Holland, 1992).

Reproduction promotes the generation of offsprings from the best fitting chromosomes through a "polarised roulette" where the number of replicates of each chromosome is proportional to its relative fitness f_i, expressed as the angle α_i allocated to each chromosome

$$\alpha_i = \frac{f_i}{\sum_i f_i} \quad \text{with} \quad \sum_i \alpha_i = 360^\circ \tag{1}$$

In this way the chromosomes with highest fitness are given the largest chance to reproduce.

Crossover consists of splitting a chromosome pair at a random location with reciprocal exchange of the two halves, thus causing a mutual flow of information between pairs.

Mutation introduces random binary changes in a chromosome. Usually the mutation likelihood is kept constant at a low value for all bits. In this paper, a new mechanism of mutation will be introduced and the consequences of this modification assessed.

1.2 Similarity templates or schemata

A *similarity template* or *schema* is defined as a mask, e.g.

$$H = [0, 1, 1, 0, 0, \#, \#, 0, \#, 1, 1, 0]$$

in which some bits take a fixed value and the rest are left unspecified. They are denoted by a "don't care" symbol "#", meaning that the bit may take either value (1 or 0). A *schema* is a useful notation to detect similarities among samples of a population. Denoting with d the number of "don't care" in the template, the number of chromosomes including that template is 2^d. Conversely a given chromosome having β bit length represents 2^β possible

templates. Goldberg (1989) and Holland (1992) have formulated the fundamental GA theorem to demonstrate how *schemata* evolve through successive generations. Their role as generation selectors has been investigated by Mühlenbein (1991) and Altenberg (1995). *Schemata* are briefly summarized here to introduce the new mutation strategy. Let $m(H,t)$ be the number of chromosomes with a given template H at time t, then its number at the next generation is given by

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \qquad (2)$$

where $\bar{f} = \frac{1}{N}\sum_j f_j$ denotes the mean fitness of the population at time t and $f(H)$ is the mean fitness of schema H and N represents the number of chromosomes in the population. This shows that *schemata* with fitness above the average will expand in the next generation, whereas *schemata* with fitness below the average will produce less and less samples in successive generations. The propagation equation (2) can be rewritten considering a *schema H* which is above the average by an amount $c\bar{f}$, where c is a constant. Substituting in Eq. (2) yields

$$m(H,t+1) = m(H,t)\frac{\bar{f}+c\bar{f}}{\bar{f}} = (1+c)\cdot m(H,t) \qquad (3)$$

Considering the evolution of the population from $t = 0$, yields

$$m(H,t) = m(H,0)\cdot(1+c)^t \qquad (4)$$

which means that above-the-average *schemata* increase exponentially through generations, whereas below-the-average *schemata* decrease with the same rate. The problem with long *schemata* is their possible break-down due to cross-over, thus loosing the distinctive features embedded in their pattern. For this reason, attention is focused on short templates, or *building block*, which are less likely to be disrupted by crossover. The previous fundamental theorem can be viewed in terms of building-blocks, meaning that a *schema* is a collection of many short similarity templates that propagate independently and exponentially over generations depending on their fitness.

1.3
Pitfalls of the basic GA

The weak point of "classical" GAs is the total randomness of mutation, which is applied equally to all chromosomes, irrespective of their fitness. Thus a very good chromosome is equally likely to be disrupted by mutation as a bad one. On the other hand, bad chromosomes are less likely to produce good ones through crossover, because of their lack of building blocks, until they remain unchanged. They would benefit the most from mutation and could be used to spread throughout the parameter space to increase the search thoroughness. So there are two conflicting needs in determining the best probability of mutation. Usually, a reasonable compromise in the case of a constant mutation is to keep the probability low to avoid disruption of good chromosomes, but this would prevent a high mutation rate of low-fitness chromosomes. Thus a constant probability of mutation would probably miss both goals and result in a slow improvement of the population.

2
Features of the new algorithm

The new mutation algorithm is now described in details. Each aspect is considered separately. First the parameter coding format is described, then the mutation strategy is introduced and justified.

2.1
Parameter coding

For each parameter x_k its lower bound I_k and span R_k define the range of variation $x_k \in [I_k, I_k + R_k]$. If β_k is the resolution of the binary coding in this range, then g_k is an integer defining the relative value of x_k according to

$$x_k = I_k + R_k\frac{g_k}{2^{\beta_k}-1};\ \Leftrightarrow\ g_k = \frac{x_k - I_k}{R_k}\cdot\left(2^{\beta_k}-1\right) \qquad (5)$$

The collection of integers $\{g_k,\ \ k = 1, 2, \ldots, N_{\text{par}}\}$ forms the binary coding of the N_{par} parameters in the chromosomes. Though crossover operates on this as a whole, the new mutation strategy requires that each parameter (*gene*) can be dealt with separately, hence the need for this particular representation. In fact crossover and mutation are mutually exclusive. Such a choice is motivated by the consideration that each crossover operation produces two new individuals whose fitness is to be evaluated before a possible mutation is applied.

2.2
New mutation operator

The points raised in Sect. 1.3 led to the idea of revisiting the mechanisms of mutation in order to obtain a more efficient mutation strategy. In the literature there are several attempts at improving the basic constant probability assumption. For example, Grefenstette (1986) used a GA as a meta-algorithm to optimize the search parameters of a standard GA. More recently, Gelsema (1995) proposed a "steady-state" variation, consisting in the selection of a chromosome pair, which was reinserted into the population in place of the worst ones, after crossover and mutation. Bramlette (1991) proposed a fitness scaling algorithm, in addition to the "steady-state" variation. Sendhoff et al. (1997) have expressed concern that small variations in the genotype space could result in small variations in the phenotype space.

In this paper, the assumption of a constant probability of mutation is abandoned in favour of an adaptive one, related to the chromosome fitness. If $\bar{f}$ is the average fitness of the population, the mutation probability of a chromosome with fitness f is determined as follows:

a) if $f > \bar{f}$, the mutation probability should be kept low because the chromosome is a good one and any mutation could disrupt its *schema*. In particular, a mutation among its most significant bits (MSBs) would send it away from the neighborhood of the extremum. So the shape of the mutation probability density function should be such as to encourage mutation of the least significant bits (LSBs), leaving the MSBs unchanged. This would lead to a further improvement of the chromosome.

b) if $f < \bar{f}$ there in no reasons to avoid the mutation of all bits, including MSBs, because this would favour the formation of a better *schema*. Thus the mutation probability density function will be assumed constant for all bits, as in conventional GAs.
c) in any case, a scaling of mutation probability density function by the factor $f/\bar{f}$ is performed in order to foster the mutation of low grade chromosomes.

These operations are summarized in the following Fig. 1.

The following starting value is assumed for the probability of mutation

$$P_m = \frac{N_{\text{par}}}{\sum_{k=1}^{N_{\text{par}}} \beta_k} = \frac{1}{\bar{\lambda}} \tag{6}$$

where N_{par} is the number of parameters (*genes*) of chromosomes; β_k is the number of bits of k-th parameter binary coding into chromosomes and $\bar{\lambda}$ is the mean bit-length of the genes. Assuming $P_m \in [0, 0.5]$ i.e. $\bar{\lambda} \geq 2$ the following probability of mutation, for cases (a) and (b) are introduced

$$\text{(a)} \quad f > \bar{f} \quad p_k(i_k) = \frac{P_m}{i_k + 1} \frac{\bar{f}}{f} \cdot \sigma \tag{7}$$

$$\text{(b)} \quad f < \bar{f} \quad p_k(i_k) = \frac{1}{4} \frac{\bar{f}}{f} \tag{8}$$

with

$$i_k = 0, 1, \ldots, \beta_k; \quad k = 1, \ldots, N_{\text{par}}$$

where $i_k = 0$ corresponds to the LSB and σ is a shape factor chosen so that in case (a) chromosomes with mean fitness ($\bar{f} = f$) have $p_k(0) = 0.5$. In fact for $f = \bar{f}$

$$p_k(0) = P_m \sigma = \frac{1}{2} \tag{9}$$

from which σ can be obtained

$$\sigma = \frac{1}{2P_m} = \frac{\bar{\lambda}}{2} \tag{10}$$

For the same reasons, mutation probability in case (b) is such as to obtain $p_k = 0.5$ for chromosomes with a fitness $f = \bar{f}/2$. With this definition of σ, in case (a) the transition bit i_T for which the $p_k(i_k)$ equals the uniform mutation probability is

$$p_k(i_k) = P_m \frac{\bar{f}}{f} \quad \Rightarrow \quad i_T = \sigma - 1 = \frac{\bar{\lambda}}{2} - 1 \tag{11}$$

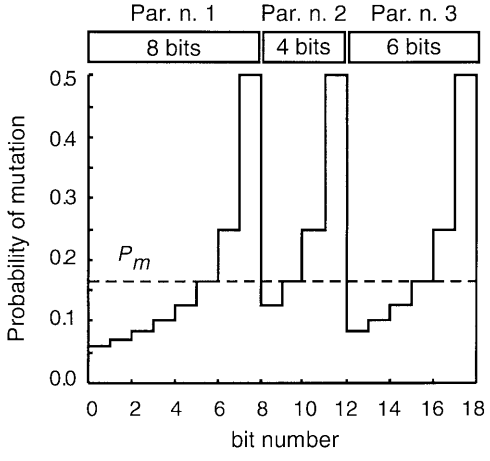

Fig. 1. Probability of mutation in case (a)

2.3 Complementary building blocks and search space exploration

The choice of $p_k = 0.5$ introduced in the previous section is now justified reminding that mutation results in the generation of chromosomes with new building blocks which are unlikely to be generated through cross-over alone. In this sense setting $p = 1$ would result in a bit-wise *NOT* operation, generating the most "distant" building blocks and thus enhancing the explorative features of the algorithm. However, applying this operation twice would re-create the original population with no obvious improvement. If, on the contrary, a lower probability of mutation $p \in (0, 1)$ is considered, new populations will be generated at each application of the mutation operator, containing unprecedented building blocks, which is precisely the reason for applying mutation. Thus the objective is to determine the best value for the probability of mutation p so that the most unlikely building blocks are generated. From the qualitative point of view, the mutation operator $M_p(\cdot)$ can be regarded as the composition of two elemental operators:

1. a $NOT(\cdot)$ operator performing a bit-wise complement of the chromosome;
2. a mutation operator $M_{p^*}(\cdot)$ where $p^* = 1 - p$

Thus the complete mutation operator can be written as

$$M_p(\cdot) \equiv \left(M_{p^*} \circ NOT\right)(\cdot) \tag{12}$$

where '$\circ$' represent the composition of the two operators. The probability of mutation through two successive applications of the operator $M_p(\cdot)$ is now computed, considering the two possible events:

event $A = \{$the bit is changed in the first application$\}$

event $B = \{$the bit is changed in the second application$\}$

The combined probability of mutation can be found as the XOR of the two events

$$\begin{aligned} \mu_2 &= \left[M_p\right]^2 = P[(A \cap \bar{B}) \cup (\bar{A} \cap B)] \\ &= P(A \cap \bar{B}) + P(\bar{A} \cap B) \\ &= P(A) \cdot P(\bar{B}) + P(B) \cdot P(\bar{A}) = 2p(1-p) \end{aligned} \tag{13}$$

Maximisation of this probability yields

$$\frac{\partial \mu_2}{\partial p} = \frac{\partial (2p - 2p^2)}{\partial p} = 0 \Rightarrow p = 1/2 \tag{14}$$

Thus the function μ_2 is parabolic with a maximum in $p = 0.5$. Therefore selecting a value of σ such that

$p_k(0) = 0.5$ produces the maximum mutation likelihood of the LSB, in the case that $f > \bar{f}$.

3 Algorithm testing

To demonstrate the algorithm capabilities, two examples of differing complexity are shown.

3.1 One-dimensional test

As a first test a discontinuous one-dimensional objective function is. This function exhibits one narrow global maximum, which coincides with the discontinuity, and two symmetrical wide local minima. Its mathematical form is the following

$$x_1 = \begin{cases} |x - x_0 - d| & x \le x_0 \\ |x - x_0 + d| & x \ge x_0 \end{cases}$$
$$\rightarrow \begin{cases} x_1 = \mathrm{abs}(x - x_0 + d \times \mathrm{sgn}(x - x_0)) \\ y = \frac{1}{x_1} + 5\left[1 + \cos\left(1.5 \times x_1 + \frac{\pi}{2}\right)\right] \end{cases} \quad (15)$$

A normally distributed noise $N(0, 0.2)$ was added to the objective function. For this example a population of 10 chromosomes over 12 generations was used. Figure 2 shows a sample run, with two successive chromosome distributions, whereas Fig. 3 summarises the algorithm performance over an average of 200 runs, performed with different noise samples.

It can be seen that the average fitness of the modified algorithm is always above that produced by a conventional GA. Thus a good approximation can be obtained in less generations.

3.2 Multimodal function with discontinuity

As a two-dimensional problem, a multimodal function $f : R^2 \rightarrow R$ with several local peaks and a discontinuity was selected. It consists of a combination of gaussian-shaped peaks with a shear in the middle of the domain, creating a step-wise discontinuity, as shown in Fig. 4.

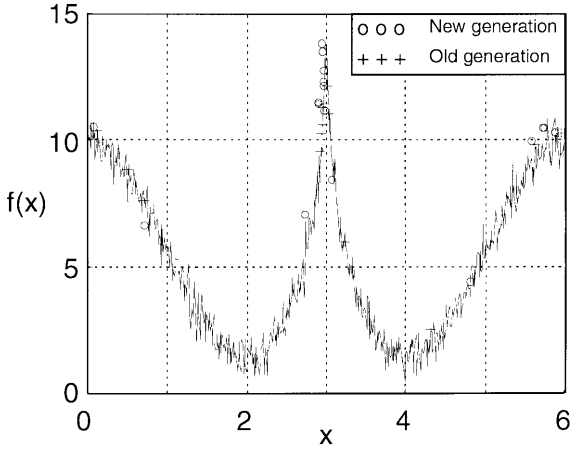

Fig. 2. Distribution of chromosomes during the search in the one-dimensional problem of Eq. (15)

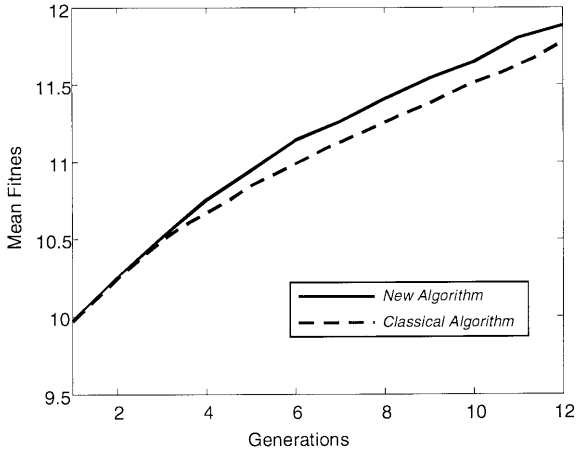

Fig. 3. Mean fitness comparison over generations for the one-dimensional case of Eq. (15)

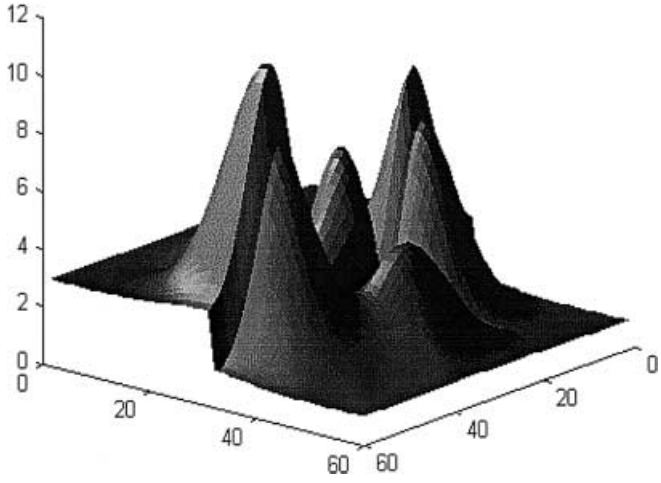

Fig. 4. Multimodal test function of Eq. (16)

The mathematical form of this function is the following

$$z = \left|3(1-x)^2 \mathrm{e}^{-x^2-(1+y)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)\mathrm{e}^{-x^2-y^2} - \frac{1}{3}\mathrm{e}^{-(1+x)^2-y^2}\right| \quad \text{for} \quad \begin{cases} -3 \le x \le 3 \\ -3 \le y \le 3 \end{cases}$$
$$\text{if} \quad x \ge 0 \Rightarrow z = z + 3 \quad (16)$$

The proposed algorithm was compared with the classical one using the following protocol: a series of 100 tests was carried out for both algorithms, using a population of 10 chromosomes over 25 generations. Each search was seeded with a new random population.

Both search parameters x_1, x_2 were coded with 8-bit words, therefore

$$N_{\mathrm{par}} = 2 \quad \beta_k = 8 \quad \text{for} \quad k = 1, 2$$
$$\bar{\lambda} = 8 \quad \sigma = \bar{\lambda}/2 = 4, \quad P_m = 1/8;$$

When $f = \bar{f}$ the probabilities of mutation are $p_k(0) = 1/2$ for the LSB and $p_k(7) = 1/16$ for the MSB. Since $i_T = \sigma - 1 = 3$, for bits $k = [0, 1, 2]$ the mutation probability is increased with respect to P_m according to Eq. (7) and decreased for the remaining ones, as already shown in Fig. 1.

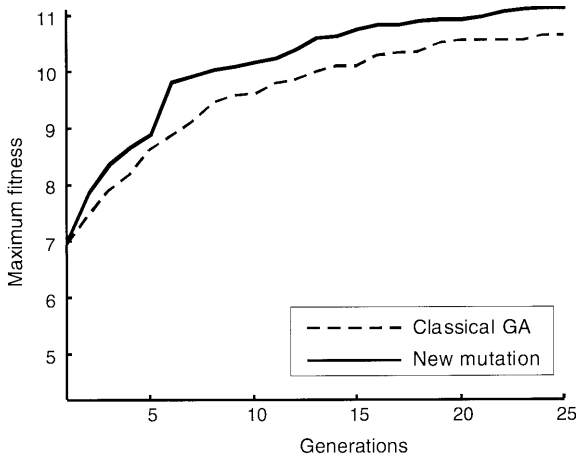

Fig. 5. Maximum fitness comparison over generations for the two dimensional case of Eq. (16)

Figure 5 compares the maximum fitness over successive generations and shows that the new algorithm yields a better performance over each generation. In fact the difference between the two algorithms is always positive in favour of the new one.

4 Conclusion

This paper has presented an improved search algorithm to reduce the chance that high-fitness chromosomes are muted during the search, thus losing their favourable *schemata*. In GAs, mutation is usually assigned a constant probability and thus all chromosome have the same likelihood of mutation irrespective of their fitness. Conversely, making mutation a function of fitness produces a more efficient search. An inverse relationship has been introduced between fitness and probability of mutation, which is a function of bit position. In this way, only the least significant bits are likely to be muted in high-fitness chromosomes, thus improving their accuracy, whereas low-fitness chromosomes are much more likely to change, enhancing their exploratory role in the search. The implications of such a new mutation scheme have been assessed. In particular, it was found that the chance of disrupting a high-fitness chromosome is decreased and the exploratory role of low-fitness chromosomes is best exploited. The algorithm was tested with two examples, a one-dimensional and a bidimensional one, each including discontinuities and multiple extrema. In all cases the new algorithm showed a faster improvement of the maximum fitness, which was always greater than the one produced by the classical GA at the same generation.

References

Altenberg L (1995) The schemata theorem and Price's theorem, In: Witley D, Vose M (Eds) Foundations of Genetic Algorithms, San Francisco: Morgan Kaufmann, pp 23–49

Bramlette MF (1991) Initialisation, mutation and selection methods in genetic algorithms for function optimisation, Proc. Fourth International Conference on Genetic Algorithms, San Diego, CA

Davis L (1990) Handbook of Genetic Algorithms, New York: Van Nostrand Reinhold

Gelsema ES (1995) Abductive reasoning in Bayesian belief networks using a genetic algorithm, Pattern Recog Lett, 16, 865–871

Goldberg DE (1989) Genetic Algorithms In Search, Optimization and Machine Learning, Addison Wesley

Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms, IEEE Trans Syst Man Cybern 16, 122–128

Holland JH (1992) Adaptation in natural and artificial systems, Cambridge, Mass.: MIT press

Mühlenbein H (1991) Evolution in time and space – the parallel genetic algorithm, In: Rawlins G (Ed) Foundation of Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, pp 316–338

Sendhoff B, Kreutz M, Seelen W (1997) Causality and the analysis of local search in evolutionary algorithms, Internal report 97-16, Institüt für Neuroinformatik, - Ruhr-Universität Bochum