Intelligent Systems

NORTH-HOLLAND

Incorporating Chromosome Differentiation in Genetic Algorithms

S. BANDYOPADHYAY*
S. K. PAL

Machine Intelligence Unit, Indian Statistical Institute, 203, Barrackpore Trunk Road, Calcutta—700 035, India

and

U. MAULIK

Department of Computer Science, Government Engineering College, Kalyani, Nadia, India

ABSTRACT

A genetic algorithmic methodology, termed a genetic algorithm with chromosome differentiation (GACD), is described which incorporates chromosome differentiation for evolutionary process. Chromosomes are distinguished into two categories of population over the generations based on the value contained in the two class bits. These are initially generated based on maximum hamming distance between them. Crossover (mating) is allowed only between individuals belonging to these categories. Theoretical analysis shows that the basic tenet of genetic algorithms holds for GACD as well; above average, short, low order schema will receive increasing number of trials in subsequent generations. It is also shown that in certain situations, the lower bound of the number of instances of a schema sampled by GACD is greater than or equal to that of the conventional genetic algorithm. Experimental results on a large number of function optimization and pattern classification problems demonstrate the significantly better performance of GACD over the conventional ones.

1. INTRODUCTION

Genetic algorithms (GAs) [1, 2] belong to a class of evolutionary search and optimization techniques and are modeled on the principles of natural genetics. These algorithms are randomized in nature, using probabilistic transition rules to change from one state to another. GAs incorporate

*Corresponding author. E-mail: sanghami@xdiv.lanl.gov

INFORMATION SCIENCES **104**, 293–319 (1998)

655 Avenue of the Americas, New York, NY 10010
0020-0255/98/$19.00
PII S0020-0255(97)00069-8

domain specific knowledge while performing a search to yield near optimal solutions in highly complex, large and multimodal search spaces. The utility of GAs in areas as diverse as image processing [3], function optimization [4], pattern classification [5], neural network design, and optimization [6], job shop scheduling [7], classifier systems [8] etc. was studied extensively.

The power of GAs lies in their ability to encode complex information and parameters of the search space in simple structures called *chromosomes* or *genotypes*, which are usually of a fixed length. An *objective function* is associated with each string which provides a mapping from the chromosomal space to the solution space. GA starts from a collection of chromosomes (called *population*), which is initially created randomly. Various biologically inspired operators like *selection*, *crossover*, and *mutation*, based on the Darwinian principles of survival of the fittest and evolution are applied on these strings over a number of generations to yield the solution of the problem. Figure 1 depicts the different steps of GA. Details can be found in [1, 2, 9].

Since in GAs the chromosomes are treated as individuals of the same type, unrestricted mating is allowed during crossover. However, nature generally differentiates the individuals of a species into more than one type or class (typical example being sexual differentiation). Cross breeding is preferred to inbreeding because of the various advantages it offers, e.g., healthier offspring, introduction of greater variety, etc. The widespread existence of this sort of differentiation and breeding styles in almost all living beings indicates the need for investigating a corresponding concept

```
Begin
        t=0
        initialize population P(t)
        compute fitness P(t)
        repeat
             t = t+1
            select P(t) from P(t-1)
            crossover P(t)
            mutate P(t)
            compute fitness P(t)
        until termination criterion is achieved
End
```

Fig. 1. Different steps of GA.

in artificial GAs. Motivated by this, an attempt is made in this article to differentiate the chromosomes into two distinct classes, M and F, respectively. The details of the methodology, subsequently referred to as GACD (GA with chromosome differentiation), are described in Section 2.2.

In addition to developing the methodology of GACD, a modified schema theorem is also presented here. It shows that the basic tenet of GAs holds for GACD also; short, low order, above average schemata will receive increasing number of trials in subsequent generations. Extensive empirical investigation was also made for a variety of function optimization and pattern classification problems. These show an overall better performance of GACD both in terms of the best value obtained and the number of generations required to attain this value.

2. GENETIC ALGORITHM WITH CHROMOSOME DIFFERENTIATION: INCORPORATING DIFFERENTIATION IN GENETIC ALGORITHM

2.1. CRITERION

Previously mentioned, nature generally differentiates the individuals of a species into more than one class. Sexual differentiation is a typical example, where the individuals of a species generally belong to either male or female class. The prevalence of this form of differentiation indicates an associated advantage which appears to be in terms of cooperation between two dissimilar individuals, who can at the same time specialize in their own fields. This cooperation and specialization should give rise to healthier and more fit offspring [1]. The appendix provides an analysis in this regard.

These observations led to the investigation into the effects of differentiating the chromosomes of a population into two different classes, namely M and F, respectively, thereby giving rise to two separate populations. Since, in addition, we would like to make the two populations most dissimilar, (this requirement is artificially imposed), these are initially generated in such a way that the hamming distance between them is maximized. Crossover is allowed between individuals belonging to the two distinct populations only. Note that the concept of restricted mating through hamming distance was also used in [10, 11]. The other genetic operators are applied classically.

As crossover is allowed between these two dissimilar groups only, a greater degree of diversity is introduced in the population leading to greater exploration in the search. At the same time conventional selection

is performed over the entire population which serves to exploit the information gained so far. Thus it appears that GACD attains a greater balance between exploration and exploitation which is crucial for any adaptive system; thereby making GACD superior to conventional GA (CGA).

2.2. *DESCRIPTION OF GENETIC ALGORITHM WITH CHROMOSOME DIFFERENTIATION*

The basic steps of GA as shown in Figure 1 are followed in GACD as well. However, the individual processes are modified. These are now discussed in detail.

Population Initialization: The structure of a chromosome of GACD is shown in Figure 2. Here the l bits, termed *data bits* encode the parameters of the problem. The initial two bits, termed the *class bits* indicate the class (M or F) of the chromosome.

Two separate populations, one containing the M chromosomes (M population) and the other containing the F chromosomes (F population), are maintained over the generations. The sizes of these two populations, p_m and p_f, respectively, may vary. Let $p_m + p_f = p$, where p is fixed (equivalent to the population size of CGA). Initially $p_m = p_f = p/2$. The data bits for each M chromosome are first generated randomly. One of the two *class bits*, chosen randomly, is initialized to 0 and the other to 1. The data bits of the F chromosomes are initially generated in such a way that the hamming distance between the two populations (in terms of the data bits) is maximum. The hamming distance between two chromosomes c_1 and c_2, $c_1, c_2 \in \mathscr{C}$, denoted by $h(c_1, c_2)$, is defined as the number of bit positions in which the two chromosomes differ. Hamming distance between two populations, P_1 and P_2, denoted by $h(P_1, P_2)$, is defined as follows,

$$h(P_1, P_2) = \sum_i \sum_j h(c_i, c_j), \qquad \forall c_i \in P_1, \forall c_j \in P_2.$$

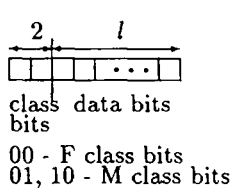

Fig. 2. Structure of a chromosome in GACD.

A method of generating p_f number of F chromosomes such that the previously mentioned restriction is satisfied, while allowing a certain amount of randomness, is developed whose pseudo code is shown in Figure 3. Here $M(i,j)$ and $F(i,j)$ indicate the jth bit of the ith M and F chromosomes in the populations, respectively. $check(i,j)$ is an auxiliary data structure used to keep track of the bits of the M chromosomes that were chosen for complementation. The *class bits* of each F chromosome are initialized to 0s.

Fitness Computation: Only the l data bits are used to compute the fitness for the chromosomes in a problem specific manner.

Selection: Selection is performed over all the $p(=p_m+p_f)$ chromosomes, (i.e., disregarding the class information) using their fitness values. In other words, all the chromosomes compete with one another for survival. The selected chromosomes are placed in the mating pool.

Crossover: Crossover is applied with probability μ_c between an M and an F parent chromosome. Each parent contributes one class bit to the offspring. Since the F parent can only contribute a 0 (its class bits being 00), the class of the child is primarily determined by the M parent which can contribute a 1 (yielding an M child) or a 0 (yielding an F child)

```
Begin
        i=0 to p_m
            j=0 to l-1
                check(i,j) = 0  /* Initialization */
        i=0 to p_f
            j=0 to l-1
                repeat
                    k = random(p_m)  /* returns an integer in
                                        the range of 0 to p_m - 1 */
                until (check(k,j)=0)  /* M(k,j) not chosen
                                         before */
                check(k,j)=1  /* M(k,j) now chosen.
                                 Not to be chosen again */
                F(i,j)= complement(M(k,j))
end
```

Fig. 3. Algorithm for initializing the F population from the initial M population in GACD.

depending upon the bit position (among the two class bits) of the M parent chosen. This process is performed for both the offspring whereby either two M or two F or one M and one F offspring will be generated.

Crossover is carried on until (a) there are no chromosomes in the mating pool, or (b) there are no M (or F) chromosomes in the mating pool. In the former case the crossover process terminates. In the latter case, the remaining F (or M) chromosomes are mated with the best M (or F) chromosome. Note that if at the start of the crossover process, it is found that the mating pool contains chromosomes of only one class, then the crossover process is discontinued.

Mutation: Bit by bit mutation is performed over the data bits only with probability μ_m. The *class bits* are not mutated.

Note: Elitism is incorporated by preserving the best chromosome, among both the M and F chromosomes, seen until the current generation, in a location outside the population.

3. SCHEMA THEOREM FOR GENETIC ALGORITHM WITH CHROMOSOME DIFFERENTIATION

In this section the schema theorem [1] is modified appropriately to incorporate the ideas of the GACD algorithm. Some definitions related to schema are first provided which is followed by an enumeration of the different terms and terminologies used. In general the M and F parameters are denoted by subscripts m and f, respectively, while parameters with no subscript denote that these are applicable over both the M and F populations. Finally an analysis of GACD with respect to schema sampling is presented where it is shown that in most situations the lower bound of the number of instances of a schema sampled by GACD is better than that of CGA.

3.1. DEFINITIONS

In this article binary strings over the alphabet $\{0, 1\}$ are considered. A *schema* h over a string of length l is defined to be a string composed of 0, 1 or the # (do not care) symbols. For e.g., # 1 # 1 # # 0 # # # is a schema of length 10. This schema will be subsequently referred to as h'. A schema indicates the set of all strings that match the schema in the positions where it has either a 0 or a 1.

The *defining position* of a schema is a position in the schema which has either a 1 or a 0. *Defining length* of a schema h, denoted by $\delta(h)$, is defined as the distance between the last defining position and the first defining

position of the schema and is obtained by subtracting the first defining position from the last defining position. For the schema h' given previously, the first defining position (counting from the left) is 2 and the last defining position is 7. Hence $\delta(h') = 7 - 2 = 5$. The *order* of a schema h, denoted by $O(h)$, is the number of defining positions in the schema. For the schema h', $O(h') = 3$.

A schema h_1 is said to be contained in another schema h_2 if for each defining position in h_2, the position is defined in h_1, the defining bit being the same. For example, let $h_1 = \#111010\#\#\#$, then h_1 is contained in h'. Note that if h_1 is contained in h_2, then $m(h_2, t) \geq m(h_1, t)$ where $m(h, t)$ represents the number of instances of h in the population at time t.

3.2. *TERMINOLOGY*

p: the total population size which is assumed to be constant. $p_m(t)$: the M population size at time t. $p_f(t)$: the F population size at time t. $\bar{f}$: the average fitness of the entire population. h: a schema. $\overline{f_h}$: the average fitness of instances of schema h over the entire population. $\overline{f_m}$: the average fitness of the M population. $\overline{f_f}$: the average fitness of the F population. l: the length of a string. $m(h, t)$: no. of instances of schema h in the population at time t. $m_m(h, t)$: no. of instances of schema h in the M population at time t. $m_f(h, t)$: no. of instances of schema h in the F population at time t. $\delta(h)$: the defining length of the schema h. $O(h)$: the order of the schema h. μ_c: the probability of crossover. μ_m: the probability of mutation.

Superscripts s and c with any of the previously mentioned symbols indicate the corresponding values after selection and crossover, respectively. It is noted that the following equalities will hold for GACD for any value of t,

$$p = p_m(t) + p_f(t), \tag{1}$$

$$\bar{f} = \frac{\overline{f_m} * p_m + \overline{f_f} * p_f}{p}, \tag{2}$$

$$m(h, t) = m_m(h, t) + m_f(h, t). \tag{3}$$

3.3. *ANALYSIS OF GENETIC ALGORITHM WITH CHROMOSOME DIFFERENTIATION*

Let us consider the effects of each operation, *selection*, *crossover*, and *mutation* separately.

Selection: Proportional selection is performed over the entire population. Hence, similar to the treatment provided in [1], the number of instances of the schema h after selection will be given by

$$m^s(h,t+1) = m(h,t) * \frac{\overline{f_h}}{\bar{f}}. \tag{4}$$

The number of instances of the schema h that will be present in the M and F populations, respectively, must obviously be proportional to the fraction present in the two populations before selection takes place. In other words,

$$m_m^s(h,t+1) = m^s(h,t+1) * \frac{m_m(h,t)}{m(h,t)}. \tag{5}$$

Similarly,

$$m_f^s(h,t+1) = m^s(h,t+1) * \frac{m_f(h,t)}{m(h,t)}. \tag{6}$$

Crossover: To analyze the effect of crossover (assuming single point crossover) on the instances of the schema h, its probability of disruption is first calculated. Instances of the schema that are members of the M population are considered first. The analysis for the F population is analogous. For the present, let us assume that an instance of the schema from the M population, if not disrupted by crossover, is placed in the male population again.

Schema h will most likely be disrupted due to crossover if all the following conditions hold.

1. Crossover occurs (with probability μ_c).
2. Crossover site falls within the first and the last defining positions (with probability $\delta(h)/(l-1)$).
3. Crossover occurs with an instance of some schema h^* in the female population such that h^* is not contained in h [with probability $1-(m_f^s(h,t+1))/(p_f^s(t+1))$].

(Note that if h^* is contained in h, then crossover can never disrupt h, i.e., schema h will survive in both the offspring. Schema h^*, on the other hand, may not survive crossover at all.)

Taking the previously mentioned three conditions into account, the probability of disruption of h in one instance of the schema may be written

as

$$\mu_c * \frac{\delta(h)}{l-1} * \left(1 - \frac{m_f^s(h,t+1)}{p_f^s(t+1)}\right). \tag{7}$$

Hence the probability of survival of one instance of the schema in the M population is given by

$$1 - \mu_c * \frac{\delta(h)}{l-1} * \left(1 - \frac{m_f^s(h,t+1)}{p_f^s(t+1)}\right). \tag{8}$$

Consequently, considering $m_m^s(h,t+1)$ instances (after selection), after crossover we get

$$m_m^c(h,t+1) \geqslant m_m^s(h,t+1)\left(1 - \mu_c * \frac{\delta(h)}{l-1} * \left[1 - \frac{m_f^s(h,t+1)}{p_f^s(t+1)}\right]\right). \tag{9}$$

The greater than sign comes because even after disruptive crossover, the schema h may survive. For example, let h and h^* be as

$$\begin{array}{r|l} h = \#1\#1\# & \#0\#\#\#, \\ h^* = \#0\#0\# & 00\#\#\#. \end{array}$$

Let the crossover site be as shown. Then after crossover the offspring are

$$\textit{child } 1 = \#1\#1\#00\#\#\#,$$

$$\textit{child } 2 = \#0\#0\#\#0\#\#\#.$$

Here *child* 1 is an instance of h, i.e., h survives possibly disruptive crossover. Other than this, the schema h may be generated due to crossover between two other strings.

Similarly the number of instances of h that will survive crossover in the F population is given by the relation,

$$m_f^c(h,t+1) \geqslant m_f^s(h,t+1)\left(1 - \mu_c * \frac{\delta(h)}{l-1} * \left[1 - \frac{m_m^s(h,t+1)}{p_m^s(t+1)}\right]\right). \tag{10}$$

It was previously assumed that if an instance of h is present in the M (or F) population, and if h is not disrupted due to crossover, then it survives in the M (or F) population. In reality the situation may not be so. Let P_1 be the probability that h survives in the M population, when it is originally present in the M population. Hence $(1-P_1)$ is the probability that h goes to the F population after crossover. Similarly let P_2 and $(1-P_2)$ be the probabilities that h survives in the M and F populations, respectively, when it is originally present in the F population.

Thus the modified equation for schema survival due to crossover is

$$m_m^{*c}(h,t+1) = P_1 \times m_m^c(h,t+1) + P_2 \times m_f^c(h,t+1).$$

The second term is introduced on considering the instances of h that are present in the F population, which survive crossover but are placed in the M population. Similarly,

$$m_f^{*c}(h,t+1) = (1-P_2) \times m_f^c(h,t+1) + (1-P_1) \times m_m^c(h,t+1).$$

Therefore the number of instances of h present in the entire population after crossover is

$$\begin{aligned} m^c(h,t+1) &= P_1 \times m_m^c(h,t+1) + P_2 \times m_f^c(h,t+1) + (1-P_2) \times m_f^c(h,t+1) \\ &\quad + (1-P_1) \times m_m^c(h,t+1) \\ &= m_m^c(h,t+1) + m_f^c(h,t+1). \end{aligned}$$

Otherwise,

$$\begin{aligned} m^c(h,t+1) \geqslant\ & m_m^s(h,t+1)\left\{1 - \frac{\mu_c \delta(h)}{l-1}\left[1 - \frac{m_f^s(h,t+1)}{p_f^s(t+1)}\right]\right\} \\ & + m_f^s(h,t+1)\left\{1 - \frac{\mu_c \delta(h)}{l-1}\left[1 - \frac{m_m^s(h,t+1)}{p_m^s(t+1)}\right]\right\}. \quad (11) \end{aligned}$$

Using (5) and (6), the right-hand side (r.h.s.) of inequality (11) may be written as

$$\frac{m^s(h,t+1)}{m(h,t)}\left\{m_m(h,t)+m_f(h,t)-\frac{\mu_c\delta(h)}{l-1}\right.$$

$$\left.\times\left[m_m(h,t)\left(1-\frac{m_f^s(h,t+1)}{p_f^s(t+1)}\right)+m_f(h,t)\left(1-\frac{m_m^s(h,t+1)}{p_m^s(t+1)}\right)\right]\right\}$$

$$=m^s(h,t+1)\left(1-\frac{\mu_c\delta(h)}{(l-1)}\left[1-\left\{\frac{m_m(h,t)m_f^s(h,t+1)}{p_f^s(t+1)m(h,t)}\right.\right.\right.$$

$$\left.\left.\left.+\frac{m_f(h,t)m_m^s(h,t+1)}{p_m^s(t+1)m(h,t)}\right\}\right]\right).$$

Let us denote the term in the curly brackets by α. Thus we may write

$$m^c_{\mathrm{GA\,CD}}(h,t+1)\geqslant m^s(h,t+1)\left(1-\mu_c\frac{\delta(h)}{l-1}[1-\alpha]\right). \tag{12}$$

In this context a slight modification of the schema theorem [1] is called for, which provides a better lower bound of the number of instances of h that survive after selection and crossover. An instance of schema h may be disrupted due to crossover *iff* it is crossed with an instance of another schema h^* such that h^* is not contained in h and the other conditions for disruptive crossover hold. Accounting for this detail, the disruption probability should be recalculated as

$$\mu_c\frac{\delta(h)}{l-1}\left(1-\frac{m^s(h,t+1)}{p}\right).$$

Hence after selection and crossover,

$$m^c_{\mathrm{CGA}}(h,t+1)\geqslant m^s(h,t+1)\left\{1-\mu_c\frac{\delta(h)}{l-1}\left[1-\frac{m^s(h,t+1)}{p}\right]\right\}. \tag{13}$$

Let us denote the term $(m^s(h,t+1))/p$ by β. Thus,

$$m^c_{\mathrm{CGA}}(h,t+1) \geqslant m^s(h,t+1)\left\{1-\mu_c\frac{\delta(h)}{l-1}[1-\beta]\right\}.$$

Mutation: Since the conventional bit by bit mutation is applied on the strings with a probability μ_m, the probability of disruption of one bit of the schema is μ_m. Probability of its survival is $1-\mu_m$. Hence the probability of survival of the schema is $(1-\mu_m)^{O(h)}$. Thus the number of instances of the schema h that are present in the population at time $t+1$ (after selection, crossover, and mutation) is given by

$$m_{\mathrm{GACD}}(h,t+1) \geqslant m^s(h,t+1)\left\{1-\frac{\mu_c\delta(h)}{(l-1)}(1-\alpha)\right\}\left\{(1-\mu_m)^{O(h)}\right\}.$$

Approximating the r.h.s., the inequality may be written as

$$m_{\mathrm{GACD}}(h,t+1) \geqslant m^s(h,t+1)\left\{1-\frac{\mu_c\delta(h)}{l-1}(1-\alpha)-\mu_m O(h)\right\}. \quad (14)$$

Similarly, the equation for CGA is given by

$$m_{\mathrm{CGA}}(h,t+1) \geqslant m^s(h,t+1)\left\{1-\frac{\mu_c\delta(h)}{l-1}(1-\beta)-\mu_m O(h)\right\}. \quad (15)$$

In order to compare $m_{\mathrm{GACD}}(h,t+1)$ and $m_{\mathrm{CGA}}(h,t+1)$, we have to consider the following cases.

CASE i. Let $m_m(h,t)=m_f(h,t)=m_1$. In that case $m^s_m(h,t+1)=m^s_f(h,t+1)=m_2$. Note that $m(h,t)=2m_1$, $m^s(h,t+1)=2m_2$, and $\beta=2m_2/p$. Then,

$$\alpha=\frac{1}{2m_1}\left(\frac{m_1m_2}{p^s_f(t+1)}+\frac{m_1m_2}{p^s_m(t+1)}\right)$$

$$=\frac{m_2}{2}\left(\frac{p}{p^s_f(t+1)p^s_m(t+1)}\right)$$

$$=\beta\left[\frac{p^2}{4p^s_f(t+1)p^s_m(t+1)}\right].$$

The minimum value of the term in square brackets is 1 when $p_m^s(t+1) = p_f^s(t+1)$. Hence $\alpha \geqslant \beta$. This in turn indicates that *lower_bound*$(m_{\mathrm{GACD}}(h, t+1)) \geqslant$ *lower_bound*$(m_{\mathrm{CGA}}(h, t+1))$, i.e., the lower bound of the number of instances of some schema h sampled by GACD is better than that of CGA.

CASE ii. Let $m_m(h,t) \neq m_f(h,t)$. Let $m_m(h,t) = \gamma m_f(h,t)$ where $\gamma \neq 1$. Then $m_m^s(h,t+1) = \gamma m_f^s(h,t+1)$. Note that $m(h,t) = m_f(h,t)(1+\gamma)$, $m^s(h,t+1) = m_f^s(h,t+1)(1+\gamma)$, and $\beta = (m_f^s(h,t+1)(1+\gamma))/p$. Thus, we may write

$$\alpha = \frac{1}{m_f(h,t)(1+\gamma)}\left(\frac{\gamma m_f(h,t) m_f^s(h,t+1)}{p_f^s(t+1)} + \frac{m_f(h,t)\gamma m_f^s(h,t+1)}{p_m^s(t+1)}\right)$$

$$= \frac{m_f^s(h,t+1)\gamma}{(1+\gamma)}\left(\frac{p}{p_f^s(t+1)p_m^s(t+1)}\right)$$

$$= \frac{m_f^s(h,t+1)(1+\gamma)}{p}\,\frac{\gamma}{(1+\gamma)^2}\,\frac{p^2}{p_f^s(t+1)p_m^s(t+1)}$$

$$= \beta\frac{\gamma}{(1+\gamma)^2}\,\frac{p^2}{p_f^s(t+1)p_m^s(t+1)}.$$

Now, in this case $\alpha \geqslant \beta$ if the following holds

$$\frac{\gamma}{(1+\gamma)^2}\,\frac{p^2}{p_f^s(t+1)p_m^s(t+1)} \geqslant 1.$$

Otherwise,

$$\frac{p}{\sqrt{p_f^s(t+1)p_m^s(t+1)}} \geqslant \frac{1+\gamma}{\sqrt{\gamma}}. \tag{16}$$

Since the previously mentioned condition (inequality 16) cannot be always ensured, we cannot conclude that *lower_bound*$(m_{\mathrm{GACD}}(h, t+1) \geqslant$ *lower_bound*$(m_{\mathrm{CGA}}(h,t+1))$. (Note also that both the functions $(1+\gamma)/\sqrt{\gamma}$ and $p/\sqrt{p_f^s(t+1)p_m^s(t+1)}$ have minimum value 2.)

In order to experimentally compare the values of m_{CGA} and m_{GACD}, an optimization problem is considered. Let $f(x) = x^2$ be the function to be

optimized. A population size of 30 (initially 15 male and female strings are considered for GACD) and string length of 10 is taken. $\mu_c = 0.8$ and $\mu_m = 0.01$. The variation of the number of instances of four schemata with different characteristics is presented over the first five generations in Figures 4a–d. It is found that the growth rates for schemata with high fitness values are greater for GACD as compared to CGA (see Figs. 4a–c). At the same time the decay rate for schema with low fitness value is also greater for GACD (see Fig. 4d).

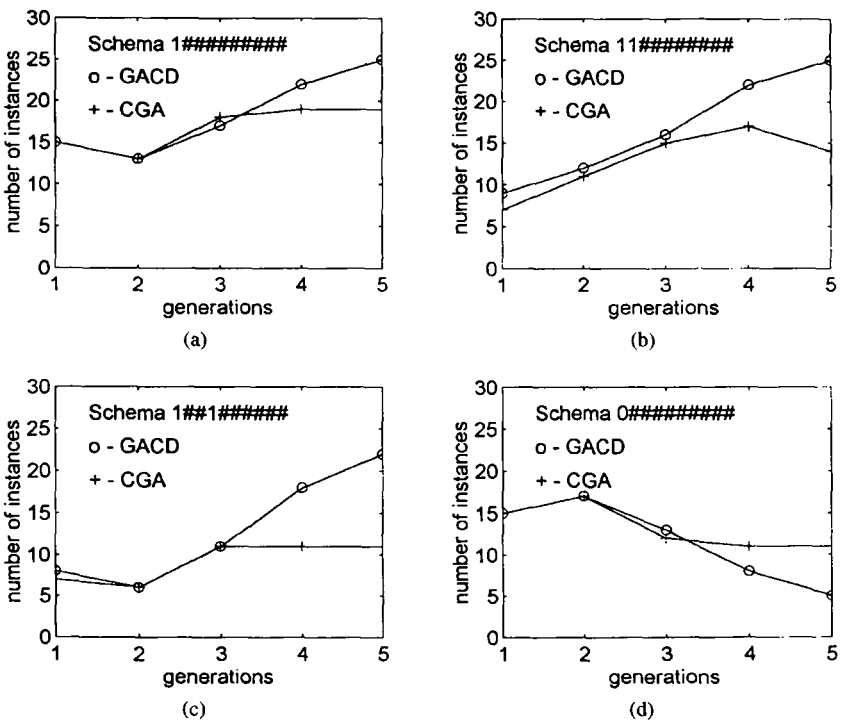

Fig. 4. (a) Variation of number of instances of schema $1\#\#\#\#\#\#\#\#\#$ with generations for the function $f(x) = x^2$. (b) Variation of number of instances of schema $11\#\#\#\#\#\#\#\#$ with generations for the function $f(x) = x^2$. (c) Variation of number of instances of schema $1\#\#1\#\#\#\#\#\#$ with generations for the function $f(x) = x^2$. (d) Variation of number of instances of schema $0\#\#\#\#\#\#\#\#\#$ with generations for the function $f(x) = x^2$.

4. EXPERIMENTAL RESULTS

An extensive comparison of the performance of the proposed method GACD with that of CGA is carried out for a variety of different function optimization and pattern classification problems. Functions of 1, 2, and 3 variables having varying degrees of complexity were chosen.

4.1. GENETIC OPERATORS

Fitness computation is done in the usual problem specific manner such that the fitness of every individual is positive. *Roulette wheel* strategy is used to implement *proportional selection* [1]. *Single point crossover* is applied on the chromosomes with a fixed probability, where the mates are chosen from classes M and F, respectively. Conventional mutation is applied on a bit by bit basis over the data bits with probability μ_m. The class bits are not mutated. The cycle of selection, crossover and mutation is repeated a number of times until one of the following occurs:

1. The average fitness value of a population becomes more or less constant over a specified number of generations.
2. A desired objective function value is attained by at least one string in the population.
3. The number of generations is greater than some threshold.

Elitism is incorporated in the algorithm by preserving the best string of the generation (among both male and female chromosomes) in a location outside the population. The best string seen up to the last generation is the solution to the problem.

4.2. FUNCTION OPTIMIZATION PROBLEMS

The experimental parameters chosen for the function optimization problems are as follows,

Population size	= 40 (CGA),
Initial male and female population sizes	= 20 (GACD),
μ_c	= 0.7,
μ_m	= 0.01,
Max. no. of generations	= 100,
No. of simulations	= 50.

In order to bring CGA and GACD logically closer for more effective comparison, a modified version of the CGA called CGAP (CGA with a different mode of population initialization) is also developed. CGAP differs from CGA only in the construction of the initial population. In CGAP, half of the initial population is generated randomly while the other half is generated in such a way that its hamming distance from the first half is maximized. The algorithm similar to the one described in Figure 3 is used for generating the second half of the population in CGAP. The function descriptions, results and associated discussions are now presented in details.

Function 1: Sparse One Max. This function is similar to the One Max function except that some fake bits are included in the string which do not contribute anything towards the objective function. Strings of length 60 are chosen, where the initial and the final 10 bits are fake. The objective function to be maximized is the number of 1s in bits 11 through 50. Maximum value of the objective function is therefore 40.

Function 2: Two Max. This function has one local and one global maxima [12]. The function is of the form,

$$f(x) = |18n - 8l|,$$

where n is the number of 1s in the l bit string representing x. There is one global maxima with value $10l$ (when x is composed of all 1s i.e., $n = l$), and one local maxima with value $8l$ (when x is composed of all 0s i.e., $n = 0$). The boundary between the two peaks occurs at $\frac{4}{9}l$. Any $n > \frac{4}{9}l$ leads to the global maxima while $n < \frac{4}{9}l$ leads to the local maxima. For $l = 30$ the global maxima has value 300 while the local maxima has value 240.

Function 3: Trap. This function with one global and one local maxima deals with a situation where the collecting area of the local maxima is much larger than the collecting area of the global maxima. The function [12] is defined as follows:

Let $z = \lfloor \frac{3}{4}l \rfloor$.
Then,

$$f(x) = \frac{8l}{z}(z - n) \qquad \text{for } n \leqslant z$$

$$= \frac{10l}{(l - z)}(n - z) \quad \text{for } n > z.$$

This function has a global maxima with value $10l$ when x is composed of all 1s and a local maxima with value $8l$ when x is composed of all 0s. For $l=30$, the global and local maximas have values 300 and 240, respectively.

Function 4: Plateau. This function contains large plateau regions which are areas in the solution space with same objective function value providing no uphill direction [12]. The function is described as follows: The l bits are divided into four equal sized groups. Each group provides a score of $2.5l$ if it contains all 1s or $0.5l$ if it contains at least one 0. The objective function for a chromosome is the sum of the scores of the four groups. Note that the only possible values of the objective function are $2l$, $4l$, $6l$, $8l$, and $10l$. For this function l is 20.

Function 5: Exp function. The function is of the form,

$$f(x) = 2 + \exp^{(x-10)}\cos(10-x) \qquad x \leqslant 10.0$$

$$= 2 + \exp^{(10-x)}\cos(x-10) \qquad x > 10.0,$$

$l=22$ and x is allowed to vary in the range $[0,20]$. Global maxima exists at $x=10.0$ where $f(x)=3.0$.

Function 6: Sine square function. This is a function of two variables, (x_1, x_2), [13] of the following form,

$$f(x_1,x_2) = 0.5 - \frac{\left(\sin\sqrt{x_1^2+x_2^2}\right)^2 - 0.5}{\left[1+0.001\left(x_1^2+x_2^2\right)\right]^2}.$$

Each variable is encoded using 22 bits ($l=44$) and is allowed to vary in the range $[-100,100]$. Global maxima with value 1 occurs when $x_1=x_2=0.0$.

Function 7: DeJong 1 function. This is a minimization problem of three variables [1], where the function to be minimized is

$$f(x_1,x_2,x_3) = \sum_{i=1}^{3} x_i^2.$$

The range for each variable is $[-5.12,5.12]$ and 22 bits are used to encode each variable. Hence $l=66$. The minimum value is 0.0 when $x_1=x_2=x_3=0.0$.

Table 1 presents the comparative results of the best values obtained after 100 iterations for GACD, CGA, and CGAP. The optimal values for the corresponding functions are also included in the table. It is seen from

TABLE 1
Comparative Results for the Best Value Obtained after 100 Iterations

Function	Optimal value	Best value obtained		
		GACD	CGA	CGAP
1	40	37.89	37.42	37.60
2	300	298.80	274.08	274.08
3	300	240.00	220.15	222.11
4	200	178.00	182.99	175.00
5	3.0	2.999997	2.987733	2.987977
6	1.0	0.970266	0.911959	0.914001
7	0.0	0.000118	0.208958	0.224651

the table that GACD outperforms both CGA and CGAP for almost all the functions. Only for Function 4, the result for GACD is inferior to that of CGA. For Function 3, it is found that none of the algorithms can attain values near the global optima. GACD attains the local maxima in all the 50 simulations. CGA and CGAP fail to attain even this value consistently. In fact, results presented later show that all the three algorithms get stuck at the local maxima for this function. Graphical demonstration of the variation of the average and best objective values are shown in Figures 5a, b, 6a, b, and 7a, b for Functions 3, 4, and 6, respectively. For Function 4, although GACD attains an objective function value that is lower than that of CGA (Fig. 6b), the variation of average value is superior (Fig. 6a). Functions 3 and 6 show a marked superior performance of GACD (Figs. 5a, b and 7a, b, respectively). Results for the remaining six functions are similar to that of Function 6 and are omitted.

The ability of the algorithms in attaining a user specified objective function value *thresh* is shown in Table 2. It shows the average number of generations required by GACD, CGA, and CGAP to attain *thresh* as well as the number of times (of a total of 50) in which this is possible. A maximum of 2000 generations are executed.

In most of the cases, it is found that GACD far outperforms both CGA and CGAP in terms of the average number of generations required to attain *thresh* and the number of times that *thresh* is attained. The value of *thresh* is chosen sufficiently close to the global optimum value (since the coding of the parameters itself, i.e., the number of bits used to code the variables, may eliminate the attainment of the exact global optimum value). Note that the average number of generations is computed for only those simulations in which *thresh* is attained.

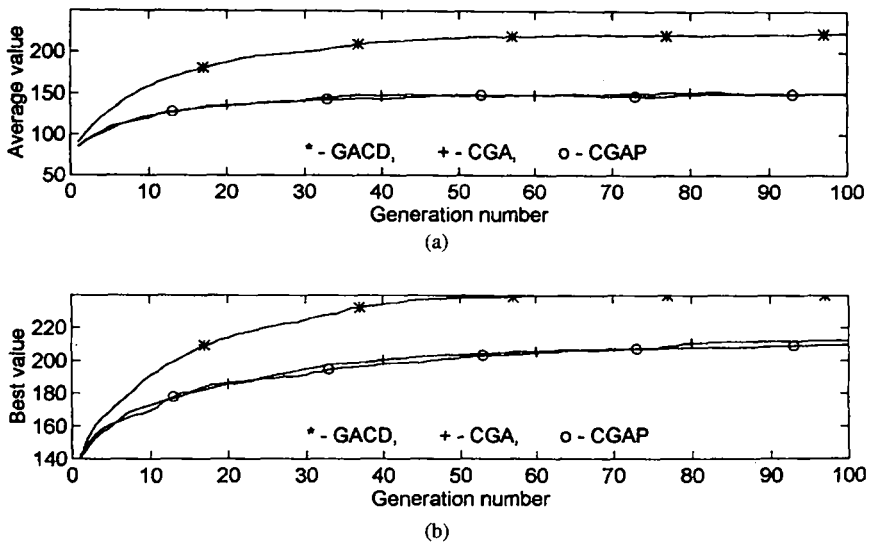

Fig. 5. (a) Variation of average value of objective function with generations for function 3 (Trap function). (b) Variation of best value of objective function with generations for function 3 (Trap function).

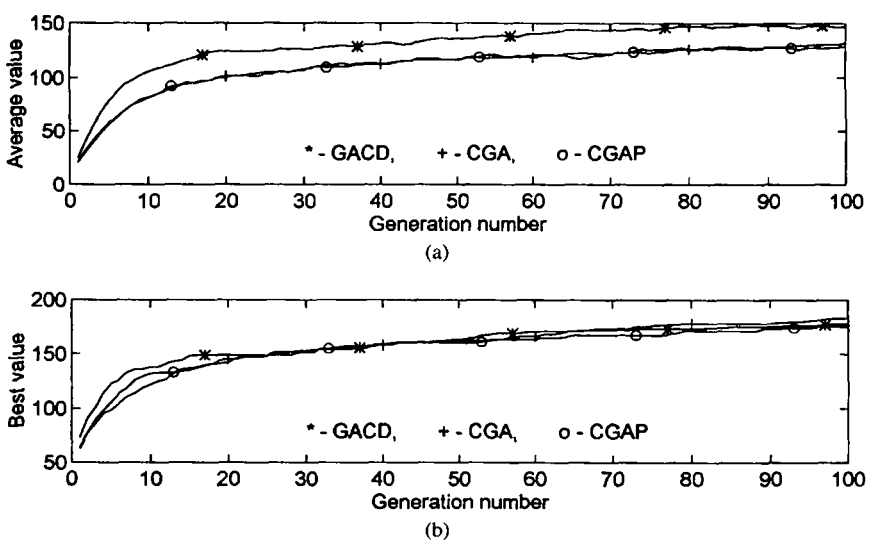

Fig. 6. (a) Variation of average value of objective function with generations for function 4 (Plateau function). (b) Variation of best value of objective function with generations for function 4 (Plateau function).

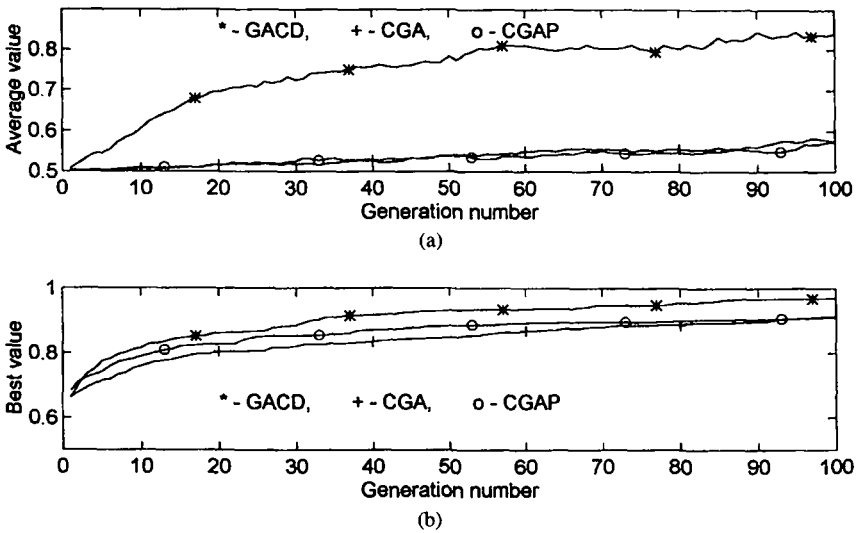

Fig. 7. (a) Variation of average value of objective function with generations for function 6 (Sine square function). (b) Variation of best value of objective function with generations for function 6 (Sine square function).

Interestingly, maximizing Function 1 appears to be difficult to solve for both CGA and CGAP. However, GACD provides the optimal result in all the 50 simulations in a reasonably small number of generations. Function 3 presents an interesting finding. None of the three algorithms could attain the global maxima in even one of the 50 simulations, thereby indicating

TABLE 2

Comparative Results for the Average Number of Generations Required to Attain an Objective Function Value *thresh*

		No. of times *thresh* attained			Avg. no. of generations		
Function	*thresh*	GACD	CGA	CGAP	GACD	CGA	CGAP
1	40	50	0	0	81.66	—	—
2	300	49	49	47	39.16	610.04	591.72
3	300	0	0	0	—	—	—
4	200	50	50	50	81.5	108.42	97.96
5	2.9999	50	45	45	31.84	665.99	755.31
6	0.999	18	7	12	218.78	898.29	986.67
7	0.003	50	16	11	42.22	1108.42	811.00

that like CGA, GACD, and CGAP get stuck at a local optima when the region from which the global optima can be reached is comparatively quite small. The results for the remaining functions show a markedly superior performance of GACD. Note that the comparative results for CGA and CGAP are not very conclusive since neither one consistently outperforms the other for the cases considered.

4.3. PATTERN CLASSIFICATION PROBLEM

Let us consider the GA based pattern classifier (or *GA-classifier*) described in [5, 14]. The supervised classification problem in $\Re^N$ can be viewed as a task of generating appropriate decision boundary in the feature space such that the misclassification of the training data points is minimized. If the decision boundaries can be modeled by a fixed number (say H) of hyperplanes, then the classification problem can be treated as search and placement of a fixed number of hyperplanes in the feature space such that the number of correctly classified samples is maximized.

The parameters of the H (fixed *a priori*) hyperplanes are encoded in the chromosomes. The hyperplanes divide the feature space into several regions. The class associated with each region is determined by the maximum number of points that belong to this class and lie in this region. Only these points are considered to be correctly classified. In this manner, the correctly classified samples in each region are summed up to provide the value of the objective function corresponding to the chromosome. The operations of selection, crossover and mutation are as described before. Elitism [1] is incorporated in GA by replacing the worst string of the current generation by the best string seen up to the last generation. Population size of 20 is chosen for this problem. The crossover probability is fixed at 0.8 while the mutation probability is varied in the range $[0.01, 0.333]$ over every 100 generations for a maximum of 1500 generations. Initially μ_m has a high value, thereby ensuring sufficient diversity in the population. Subsequently, it is decreased gradually to the minimum value, when the algorithm is allowed to make a detailed search in the solution space. The mutation probability is again increased indicating an increase in the randomness of the search. In case the optimal string was already obtained, elitism ensures that it is not lost.

The results for two artificial data, speech data and iris data [15] are presented in Table 3 for GACD and CGA. The two-dimensional artificial data sets, *ADS 1* (Fig. 8) and *ADS 2* (Fig. 9), consist of 557 and 417 points, respectively, belonging to two classes. The real life speech data, *Vowel data* [16], consists of three feature values (corresponding to the three formant

TABLE 3
Comparative Results for the Pattern Classification Problems

Data set	H	Avg. no. of generations		Training (*avg. miss*)		Testing (*avg. recog. score*)	
		GACD	CGA	GACD	CGA	GACD	CGA
ADS 1	5	573.2	854.3	0	0.7	92.23	91.89
ADS 2	5	620.5	911.5	0	0.3	90.53	86.65
iris	5	50.8	49.9	0	0	93.41	89.99
Vowel	5	1500	1500	7.4	9.4	75.98	71.32
Vowel	6	1500	1500	6.5	7.8	70.51	71.34

frequencies) and six classes $\{\delta, a, i, u, e, o\}$. Figure 10 shows the data set in the first and second formant frequency plane. Iris data comprises 150 samples having four features and belonging to three classes with 50 points in each class. The superior performance of *GA-classifier* using CGA, with respect to Bayes classifier, k-NN rule and multilayered perceptron was already demonstrated in [14] for these data sets.

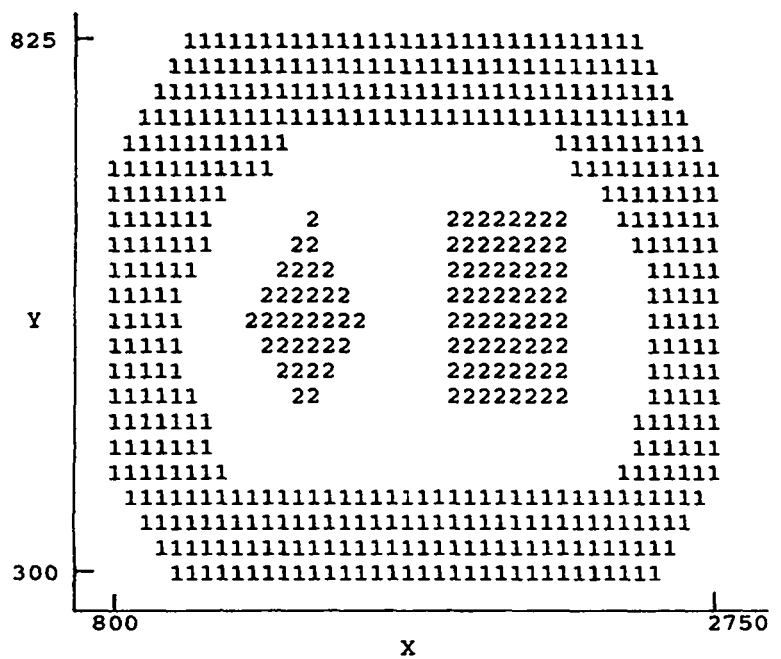

Fig. 8. Artificial data set *ADS 1*.

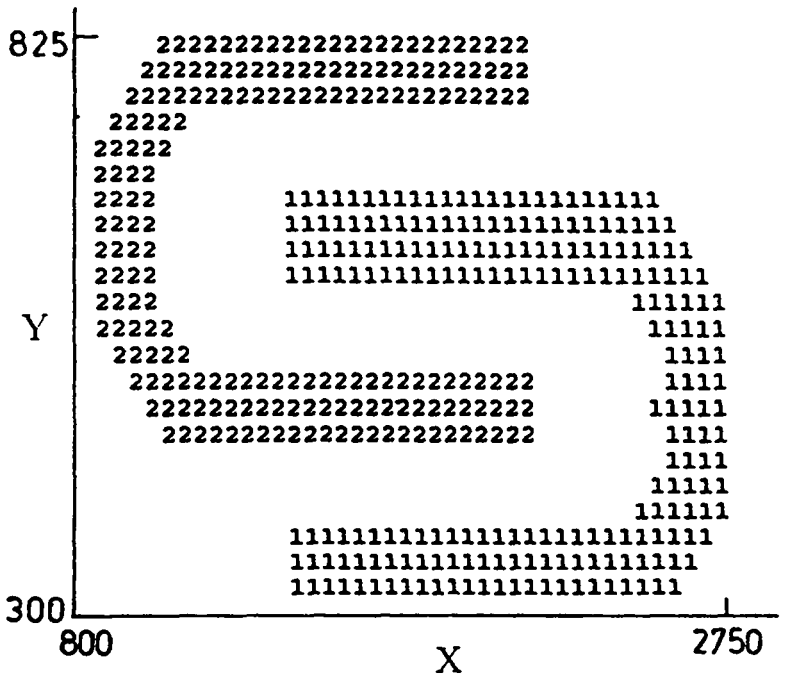

Fig. 9. Artificial data set *ADS 2*.

REMARKS.

i. The results shown are the average values obtained over 10 simulations of GACD and CGA. Ten percent of the training data set is used for training. The remaining 90% sample points are used for testing. *miss* represents the number of points misclassified by *GA-classifier*. *recog. score* denotes the percentage of correctly classified test data.

ii. For this problem, it is obvious that the performance during training is of greater importance for comparison between GACD and CGA. The results for the test case are included for the sake of parity.

For the two artificial data sets it is seen from Table 3 that GACD performs much better than CGA both for the training and test data. A point to be mentioned here is that CGA could attain zero misclassification in 7 and 8 simulations for *ADS 1* and *ADS 2*, respectively, (out of 10) while GACD attained this in all the 10 simulations. For iris data the

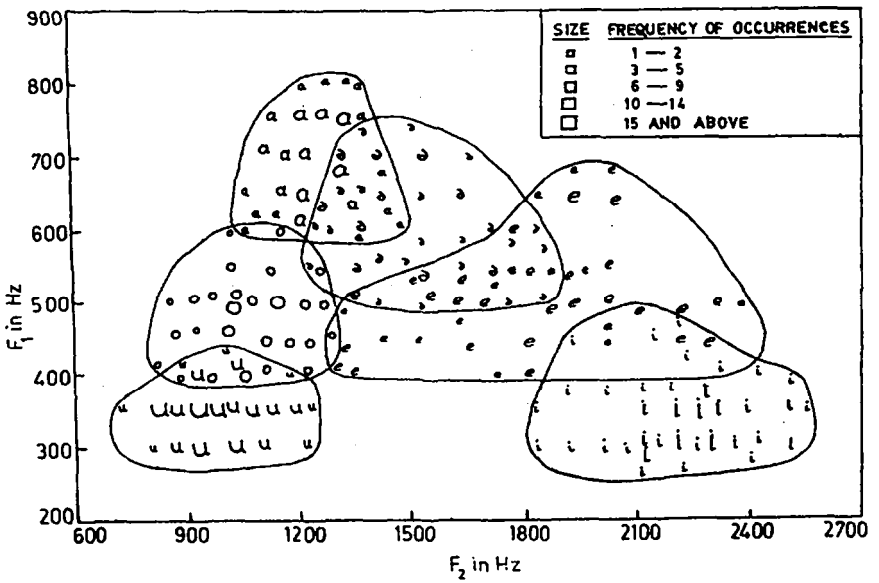

Fig. 10. Real life speech data, *Vowel data*, in the first and second formant frequency planes.

performance of GACD and CGA is comparable in terms of the average number of generations required to attain zero misclassification (which both could attain). Incidentally, the recognition score of test data is better for GACD.

It is known that *Vowel data* has considerable amount of overlap [16]. Therefore it is not surprising that both GACD and CGA fail to attain zero misclassification in all the simulations. As expected, the average *miss* decreases when H is increased from 5 to 6 for both the algorithms. GACD is seen to perform better, in terms of *miss*, for this data set in both the cases. Interestingly, for $H=6$, the recognition score for the test data is found to be marginally better for CGA, although the performance during training is seen to be poorer. However, for $H=5$, GACD performs significantly better both during training and testing.

5. DISCUSSION AND CONCLUSIONS

The effect of differentiating the chromosomes into two distinct classes in GAs is studied in this article. A new methodology called GACD is

formulated in which two separate populations are maintained and crossover is allowed only between individuals belonging to these two different classes.

GACD is shown to satisfy the schema theorem. It is proved that in many cases the lower bound of the number of instances of a schema h sampled by GACD is greater than or equal to that of CGA. Because of this, GACD is better able to exploit the information gained so far. Again, initializing the M and F populations in such a way so as to maximize the hamming distance between them, and allowing mating between individuals from these two dissimilar populations, enhances the exploration capability of GACD. Therefore, GACD appears to strike a better balance between exploration and exploitation, which is crucial for any adaptive optimization technique, thereby giving it an edge over the conventional GA.

Experimental evidence for a simple function optimization problem is provided to show that the growth and decay rate for above and below average schemata, respectively, are greater for GACD as compared to CGA. The superiority of GACD over CGA and CGAP (both in terms of the best value obtained and the average number of iterations required in finding this value) is extensively established through a series of function optimization and pattern classification problems. Although the results demonstrated here assume elitist model, experiments were also conducted for the nonelitist version, and the conclusion as mentioned previously still holds.

Two bits are utilized for differentiating the chromosomes into two classes (keeping analogy with the X,Y chromosomes of human beings). Obviously, this is not the unique choice. An alternative could have been to use one bit. However, since we want the class of the offspring to be determined by both the parents, one bit proves to be insufficient. Again, more than two bits could have been used for this purpose; but this would lead to increased computational complexity. Note also that in the present method we have incorporated differentiation into two categories. Similar differentiation into more than two classes can be formulated within the same framework in case nature demands so.

It was proved in [17] that any elitist model of GAs will definitely converge to the optimal string as the number of iterations tends to infinity provided the probability of going from any population to the one containing the optimal string is greater than zero. Note that the conventional mutation operation alone ensures that this probability is greater than zero. Since GACD utilizes the conventional mutation operation and incorporates elitism, the previously mentioned criteria are fulfilled. Thus GACD is also guaranteed to provide the optimal string as the number of iterations goes to infinity.

APPENDIX: MERITS OF COOPERATION AND SPECIALIZATION

Let an individual spend the available time in two different activities, say nurturing and hunting. If the proportion of the time spent on nurturing and hunting are n and h, respectively, then the survival probability of the offspring, $s(n,h)$, is postulated to be equal to $n*h$. If the loss of time available for either activity is proportional to the product of the activity proportions, which is termed jack-of-all-trade loss, then the constraint equation obtained is

$$n+h+anh=1, \tag{17}$$

where a is the loss coefficient. Elementary analysis shows that $s(n,h)$ attains the maximum value of 0.25 when $n=h=0.5$ and $a=0$.

On the contrary, if two individuals cooperate to act as one unit, then the survival probability (given by $s=\frac{1}{2}(n_1+n_2)(h_1+h_2)$) immediately increases to 0.5 for $a=0$. In this case $n_1+n_2=1$ and $h_1+h_2=1$, where n_1, n_2, h_1, and h_2 are defined analogously for the two individuals. The constraint (17) holds as follows,

$$n_i+h_i+an_ih_i=1, \qquad i=1,2.$$

Hence the individuals must cooperate but need not specialize. For the case when $a\neq 0$, the maximum survival probability is obtained when either $(n_1,n_2)=(1,0)$ or $(0,1)$ while $(h_1,h_2)=(0,1)$ or $(1,0)$. This indicates full specialization and cooperation within the unit. In either case, the survival probability is larger compared to the uncooperative individual [1].

This work was carried out when Sanghamitra Bandyopadhyay held a fellowship awarded by the Department of Atomic Energy, Govt. of India.

REFERENCES

1. D. E. Goldberg, *Genetic Algorithms: Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
2. L. Davis (Ed.), *Handbook of Genetic Algorithms*, Van Nostrand-Reinhold, New York, 1991.
3. S. K. Pal, D. Bhandari, and M. K. Kundu, Genetic algorithms for optimal image enhancement, *Pattern Recognit. Lett.* 15:261–271 (1994).
4. K. DeJong, Analysis of the behaviour of a class of genetic adaptive systems, Ph.D. Thesis, Dept. Computer and Communication Science, University of Michigan, Ann Arbor, MI, 1975.

5. S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, Pattern classification using genetic algorithms, *Pattern Recognit. Lett.* 16:801–808 (1995).
6. D. Whitley, T. Starkweather, and C. Bogart, Genetic algorithms and neural networks, *Parallel Comput.* 14:347–361 (1990).
7. L. Davis, Job shop scheduling with genetic algorithms, in: *Proceedings of the 1st International Conference on Genetic Algorithms*, Lawrence Earlbaum Associates, Hillsdale, NJ, 1985, pp. 136–140.
8. R. K. Belew and J. B. Booker (Eds.), *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann, University of California, San Diego, CA, July 1991.
9. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 1992.
10. L. J. Eshelman, The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination, in: G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, 1991, pp. 265–283.
11. L. J. Eshelman and D. Schaffer, Preventing premature convergence by preventing incest, pp. 115–122 in [8].
12. D. H. Ackley, An empirical study of bit vector function optimization, in: L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Los Altos, CA, 1987, pp. 170–204.
13. J. D. Schaffer, R. Caruana, L. J. Eshelman, and R. Das, A study of control parameters affecting the online performance of genetic algorithms for function optimization, in: *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989, pp. 51–60.
14. S. K. Pal, S. Bandyopadhyay, and C. A. Murthy, Genetic algorithms for generation of class boundaries, *IEEE Trans. Syst., Man, Cybern.* (1997), to appear.
15. R. A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. Eugenics* 3:179–188 (1936).
16. S. K. Pal and S. Mitra, Multilayer perceptron, fuzzy sets and classification, *IEEE Trans. Neural Networks* 3(5):683–697 (1992).
17. D. Bhandari, C. A. Murthy, and S. K. Pal, Genetic algorithm with elitist model and its convergence, *International J. Pattern Recog. Artif. Intell.* 10(6):731–747 (1996).

Received 19 August 1996; revised 7 January 1997