Applied Intelligence 18, 7–25, 2003

Incorporating Ancestors' Influence in Genetic Algorithms

SUSMITA GHOSH (NEE DE)
Department of Computer Science & Engineering, Jadavpur University, Kolkata 700 032, India
susmita_de@hotmail.com

ASHISH GHOSH AND SANKAR K. PAL
Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700 108, India
ash@isical.ac.in
sankar@isical.ac.in

Abstract. A new criterion of fitness evaluation for Genetic Algorithms is introduced where the fitness value of an individual is determined by considering its own fitness as well as those of its ancestors. Some guidelines for selecting the weighting coefficients for quantifying the importance to be given to the fitness of the individual and its ancestors are provided. This is done both heuristically and automatically under fixed and adaptive frameworks. The Schema Theorem corresponding to the proposed concept is derived. The effectiveness of this new methodology is demonstrated extensively on the problems of optimizing complex functions including a noisy one and selecting optimal neural network parameters.

Keywords: evolutionary computation, neural networks, object extraction, optimization, schema analysis

1. Introduction

Genetic algorithms (GAs) [1–6] are adaptive and robust computational procedures modeled on the mechanics of natural genetic systems. They can be viewed as randomized yet structured search and optimization techniques. GAs efficiently exploit the historical information so that new offspring with expected improved performance can be generated [1]. They iteratively perform the following cycle of operations on a set of coded solutions or chromosomes, called a *population,* until some termination condition is achieved: *selection* (including fitness evaluation of each solution), *reproduction* (including crossover and mutation), and *reduction/replacement* of the old population with a new one.

Conventional genetic algorithms (CGAs) consider only the fitness value of the chromosome under consideration for measuring its suitability for selection for the next generation, *that is,* the fitness of a chromosome z is $g(f(z))$, where $f(z)$ is the objective function and g is another function which by operating on $f(z)$ gives the fitness value. Hence, a CGA does not discriminate between two identical offspring, one produced from better (highly fit) parents and the other from comparatively weaker (low fit) parents. In nature, normally an offspring is more fit or suitable if its ancestors (parents) are better, *that is,* an offspring possesses some extra facility to exist in its environment if it belongs to a better family (or its ancestors are highly fit) [7, 8]. In other words, the fitness of an individual depends also on the fitness of its ancestors.

The present article provides an investigation based on the aforesaid observation. We propose a new way of measuring the fitness of an individual by considering its own fitness as well as the fitnesses of its ancestors. The fitness of a chromosome z is $g(f(z), a_1, a_2, \ldots, a_n)$ where a_is are the original fitness values of its n ancestors. The function g may be of various types considering the amount of importance given to the fitness of different ancestors. Various procedures are described for determining the weights which characterize the degree of ancestors' importance. The weights may be kept

constant or varying during the operation of GAs. The effectiveness of this concept has been demonstrated experimentally on various problems such as complex function optimization, noisy function evaluation, selection of optimal set of weights in a multilayer perceptron [9–11], and evolving Hopfield type optimum neural network architectures for object extraction [12]. A schema analysis is also provided (Appendix A).

The rest of the article is organized as follows. In Section 2 a brief introduction to genetic algorithms is provided. Section 3 describes the proposed fitness evaluation methodology. Implementation details and analysis of results are presented in Section 4 and concluding remarks are put in Section 5.

2. Genetic Algorithms: Basic Principles and Features

Conventional GAs (CGAs) are intended to mimic some of the processes observed in natural evolution. The evolution starts from a set of individuals and proceeds from generation to generation through genetic operations. Replacement of an old population with a new one is known as a generation when generational replacement technique (replace all the members of the old population with the new ones) is used. Another reproduction technique, called steady-state reproduction [13], replaces one or more individuals in each iteration instead of the whole population. This cycle is repeated until a desired termination condition is attained. A schematic diagram of the basic structure of a genetic algorithm is shown in Fig. 1.

A GA typically consists of the following components [1, 14]:

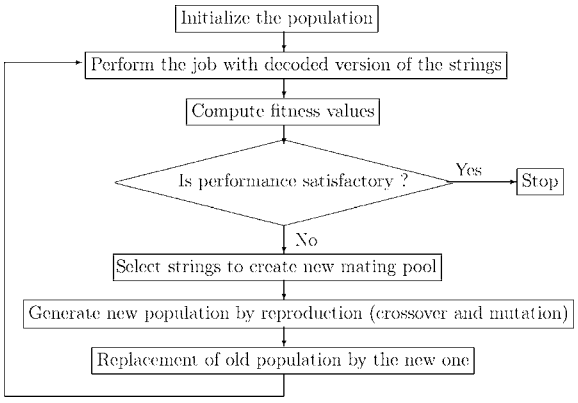

Figure 1. Basic steps of a genetic algorithim.

- A population of strings or coded possible solutions (often referred to as chromosomes).
- A mechanism to decode and encode a possible solution (often as a binary string).
- Objective function and associated fitness evaluation techniques.
- Selection procedure.
- Reproduction (with the help of some genetic operators, e.g., crossover and mutation).
- Probabilities to perform genetic operations.
- Reduction/replacement of population for the next generation.

3. A New Fitness Evaluation Criterion

In this section we describe a new method for evaluating the fitness of an individual chromosome by considering the effect of fitness of its ancestors along with its own fitness. As mentioned before, in a CGA, chromosomes are selected for reproduction based on their own fitness values. The process does not consider any influence of its ancestors (predecessors). But in nature, 'family background' plays a significant role to determine the characteristics and suitability of offspring; descendants from a better family (highly fit ancestors) invariably possess some extra advantages to be treated as fit in an environment [7, 8]. This observation has motivated us to consider the effect of fitness of ancestors (or parents) to measure the fitness of individuals.

Considering the influence of n ancestors, the modified fitness value (MFV) of an individual chromosome z will be

$$MFV = g(fit, a_1, a_2, \ldots, a_n), \tag{1}$$

where fit is the original fitness of the individual z, and $a_1, a_2, \ldots, a_n$ are the fitnesses of its n ancestors. A simple form of g may be as follows:

$$MFV = \alpha\, fit + \sum_{i=1}^{n} \beta_i a_i, \tag{2}$$

where α and β_is are the weights quantifying the degree of importance of the fitness of the individual under consideration and those of its ancestors. For convenience, we have taken $\alpha + \sum_{i=1}^{n} \beta_i = 1$, where $0 \leq \alpha, \beta \leq 1$. These weighting coefficients may be taken as static (initially set to some value, based on heuristics, and kept constant throughout the procedure), or dynamic (these coefficients will change automatically with evolution).

Now, if we consider the effect of a single previous generation, we need to take into account the fitness of the parents (say, p_1 & p_2) only; in that case (2) reduces to

$$MFV = \alpha\, fit + \beta_1 p_1 + \beta_2 p_2. \tag{3}$$

As a special case, if we choose $\beta_1 = \beta_2 = \beta$, (*that is,* we put equal weight to both parents), then

$$MFV = \alpha\, fit + \beta(p_1 + p_2). \tag{4}$$

If we go two generations back, *that is,* fitness of parents (p_1 & p_2) as well as that of grandparents (say, gp_1, gp_2, gp_3 & gp_4) are considered, (2) will take the form

$$MFV = \alpha\, fit + \sum_{i=1}^{2} \beta_i p_i + \sum_{j=1}^{4} \gamma_j\, gp_j, \tag{5}$$

where β_is and γ_js are the weights corresponding to parents and grandparents, respectively. Moreover, if $\beta_1 = \beta_2 = \beta$, and $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = \gamma$, then

$$MFV = \alpha\, fit + \beta(p_1 + p_2) + \gamma(gp_1 + gp_2 + gp_3 + gp_4). \tag{6}$$

Intuitively, $\beta_i > \gamma_j$, $\forall i$ & $\forall j$, *that is,* the influence of grandparents is smaller than that of the parents. In this context one may note that even when we explicitly consider the influence of only parents for determining the suitability of an individual, the influence of grandparents and their ancestors also comes into account implicitly because the computation of fitnesses of chromosomes at generation t requires the fitness values of chromosomes of generation $(t-1)$ which, in turn, are computed using those in generation $(t-2)$.

A more general form of g can be used to calculate *MFV* as

$$MFV = \left(\alpha\, fit^r + \sum_{i=1}^{n} \beta_i a_i^r\right)^{\frac{1}{r}}, \quad \forall r \geq 1, \tag{7}$$

where α, *fit*, a_is, and β_is carry the same meaning as stated earlier (see Eq. (2)). The term inside the bracket should always be positive. Equation (2) corresponds to the case $r = 1$. For fixed a_is, β_is, α, and *fit*, *MFV* monotonically increases with r. The value of r determines the amount of importance to be given on the fitness of different ancestors.

Schemes for Selection of Weighting Coefficients

In this section we describe a few ways of determining the weighting coefficients (α, β).

Scheme 0. Here we use conventional genetic algorithm (CGA), *that is,* $\alpha = 1$ and $\beta = 0$ (see Eq. (4)).

Scheme 1. In this case weighting coefficients are heuristically assigned. We use ($\alpha > \beta_i$), *that is,* more importance is given to the offspring itself than to its parents. We have considered $\alpha = 0.5$ & $\beta_1 = \beta_2 = \beta = 0.25$ (see Eq. (3)).

Scheme 2. Here also weighting coefficients are assigned heuristically. Values are altered depending on whether mutation has occurred on a chromosome or not. Since the purpose of applying mutation operation is to bring out a drastic change in the characteristics of an individual chromosome, the influence of the parents on the said mutated chromosome should be smaller. Hence, less weight will be given on the fitness of parents and more weight on that of the individual chromosome, in case mutation occurs. We have considered, $\alpha = 0.8$ & $\beta_1 = \beta_2 = 0.1$ if mutation occurs on a chromosome; otherwise, $\alpha = 0.5$ & $\beta_1 = \beta_2 = 0.25$ (see Eq. (3)).

Scheme 3. This scheme considers the reverse effect of mutation (Scheme 2) on selecting the weighting coefficients. Hence, more weight will be given on the fitness of parents and less weight on that of the individual chromosome, if mutation occurs. In case of mutated chromosome, $\alpha = 0.5$ & $\beta_1 = \beta_2 = 0.25$; otherwise, $\alpha = 0.8$ & $\beta_1 = \beta_2 = 0.1$ (see Eq. (3)).

Scheme 4. Here the difference between the fitness values of the individual chromosome and that of its parents is used to find the weighting coefficients (βs) of parents (weighting coefficient of children (α) can be determined automatically since $\alpha + \beta = 1$ where, $0 \leq \alpha, \beta \leq 1$). The influence of both the parents is treated separately. The criterion is such that when the fitness of parent/child is greater than that of child/parent then increase/decrease the weighting coefficient (β) of a parent in proportion to this difference. This ensures higher β value for a parent having higher fitness value.

Let f_{ch} be the fitness value of a chromosome and f_{p1} and f_{p2} be the fitness values of its two parents. If $f_{ch} \geq f_p$ (f_p is taken as the fitness value of a parent),

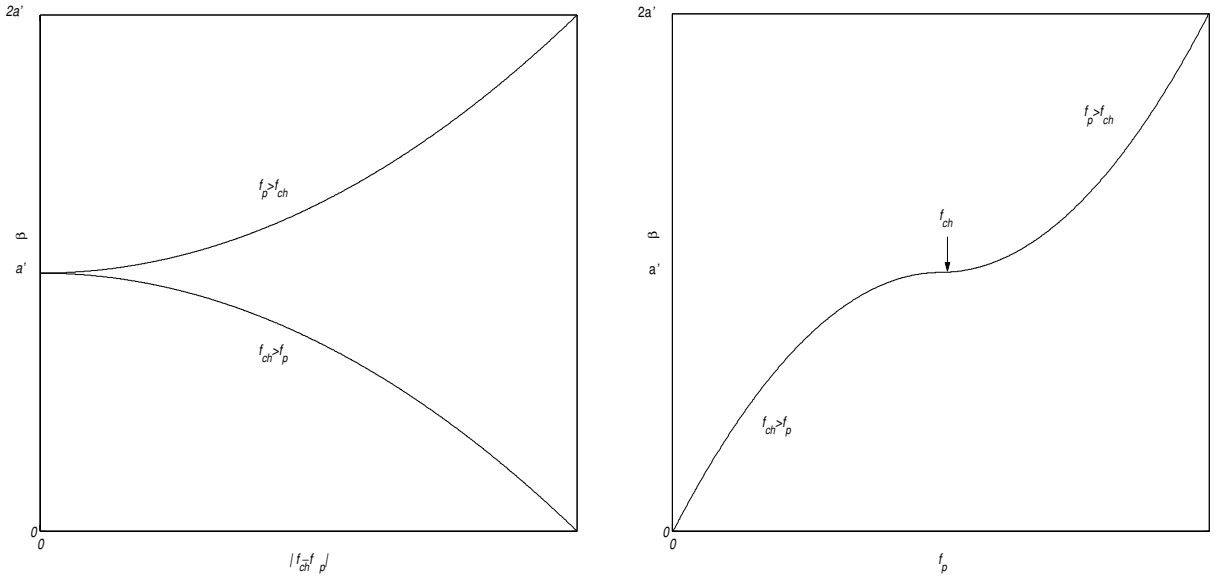

Figure 2. Variation of β ($a' = 0.25, m = 2$) with (a) $|f_{ch} - f_p|$; (b) f_p keeping f_{ch} constant.

the weight β of that parent is defined as

$$\beta = a'\left(1 - \left(\frac{f_{ch} - f_p}{\text{const}}\right)^m\right), \quad m \geq 1, \qquad (8)$$

where

$$\text{const} = \max\{((f_p)_{bst} - (f_{ch})_{wst}), ((f_{ch})_{bst} - (f_p)_{wst})\},$$

is a normalizing factor. a' is a preassigned constant which controls the maximum value of β. $(f_p)_{bst}$ & $(f_p)_{wst}$ denote, respectively, the fitness values of the best and worst chromosomes of the parent population and $(f_{ch})_{bst}$ & $(f_{ch})_{wst}$ correspond to those of the child population. Here, $\beta \in [0, a']$. As $f_{ch} - f_p$ increases, β decreases.

On the other hand, if $f_p > f_{ch}$, then

$$\beta = a'\left(1 + \left(\frac{f_p - f_{ch}}{\text{const}}\right)^m\right), \quad m \geq 1. \qquad (9)$$

As $f_p - f_{ch}$ increases, β increases. Here, $\beta \in [a', 2a']$. Combining (8) and (9), the expression for β can be written as

$$\beta = a'\left(1 + n'\left(\frac{|f_{ch} - f_p|}{\text{const}}\right)^m\right), \quad m \geq 1, \qquad (10)$$

where $|f_{ch} - f_p|$ represents the absolute difference between two fitness values f_{ch} and f_p. $n' = 1$ if $f_p > f_{ch}$, else $n' = -1$.

Therefore, unlike Schemes 1–3, here the weighting coefficients are determined based on fitness values (using (10)). For simulation we have taken $a' = 0.25$. Values of m are considered to be 2 and 3, and accordingly we define two subschemes *viz 4a*: when $m = 2$ and *4b*: when $m = 3$.

From the previous discussion, it is clear that $\beta \in [0, 2a']$ and β is symmetric about a'. Here, the maximum value of a' can be 0.25 to ensure $\alpha \geq 0$ (since $\alpha = 1 - (\beta_1 + \beta_2)$, considering the influence of parents only). Figure 2(a) shows the variation of β with $|f_{ch} - f_p|$ ($a' = 0.25, m = 2$) while Fig. 2(b) shows the variation of β with f_p keeping f_{ch} constant ($a' = 0.25, m = 2$).

Note. In (8) and (9), $\beta = 2a'$ indicates that a parent having the maximum fitness value is producing a child with minimum fitness value; this is a rare event in nature. Therefore, attainment of this condition in a GA might affect the evolutionary process.

It is also to be noted that, in many optimization techniques historical information is exploited to determine the present status. For example, in backpropagation algorithm of neural networks we see that the present weight values depend on previous weight values and

the difference between actual output and target output. Similarly, starting from an initial state (solution) a simulated annealing algorithm changes its state according to some probabilistic transition rule that depends on the fitness values of the previous and the newly generated states. This strengthens our key idea of introduction of Scheme 4 where we tried to quantify the historical information by considering the difference between the fitness values of the individual chromosome and those of its parents.

Scheme 5. Here the weighting coefficients are evolved automatically over the sequence of generations by considering them as a part of chromosomes of the population. Hence, some fields of the chromosome representation of possible solutions are kept for α and β_i. Due to crossover and mutation, values of α and β_i will be changing with time and thus they will evolve automatically. As we choose better chromosomes from generation to generation, the evolved values of α and β_i will be more suitable for that environment. Parents are given equal importance, *that is,* we use (4). Since, $\alpha + \beta = 1$, we need to evolve only one weighting factor (say, α).

Scheme 6. Similar to Scheme 5, but the amount of importance given to different parents is different ((3)). Since, $\alpha + \beta_1 + \beta_2 = 1$, we need to evolve any two weights. We restrict β_1 & β_2 in [0, 0.5); it ensures that $\alpha > 0$, *that is*, some weight will always be there for the individual chromosome.

Scheme 7. The influence of grandparents in addition to parents is taken into account in Schemes 1, 2, 5, and 6 using (5) and (6).

4. Implementation and Experimental Results

The effectiveness of the aforesaid concept has been demonstrated on three problems, namely, (i) optimizing complex functions including a noisy one (De Jong $F4$ [1]), (ii) selection of an optimal set of weights and thresholds in a multilayer perceptron (MLP) for two input XOR function, and (iii) evolution of Hopfield type optimum neural network architectures for object extraction from noisy images [12]. These are described below.

Table 1. Functions used for optimization.

Function	Functional form	Domain	Optimum value
$F1$	$\prod_{i=1}^{10}(x - 2i)$	$[0, 20]$	3.72×10^9
$F2$	$0.5 - \frac{\{\sin\left(\sqrt{(x^2+y^2)}\right)^2 - 0.5}{(1.0+0.001(x^2+y^2))^2}$	$[-100, 100]$	1.0
De Jong $F4$	$\sum_{i=1}^{30} i x_i^4 + \text{Gauss}(0, 1)$	$[-1.28, 1.27]$	0.0 (without noise)

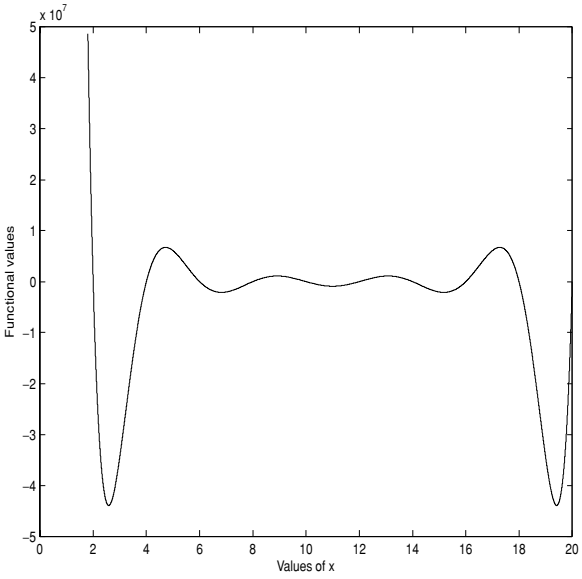

Figure 3. Sketch of the function $\prod_{i=1}^{10}(x - 2i)$ ($x \in [0, 20]$).

4.1. Function Optimization

Table 1 shows the three functions which have varying degrees of complexity with respect to number of optima. The first one ($F1$) is a univariate function, the second ($F2$) is a bivariate, and the third one (De Jong $F4$) has 30 variables [1, 15]. The complex behavior of $F1$ and $F2$ is depicted in graphical form in Figs. 3 and 4. $F1$ has 10 maxima with the global maximum at $x = 0$. $F2$ is a rapidly varying multimodal function with several close oscillating hills and valleys with a global maximum at $x = y = 0$. Gaussian noise with a mean value zero and a standard deviation of 1.0 is added to the functional value of De Jong $F4$.

4.1.1. Experimental Set-up. To optimize functions $F1$ and $F2$, the following steps are adopted.

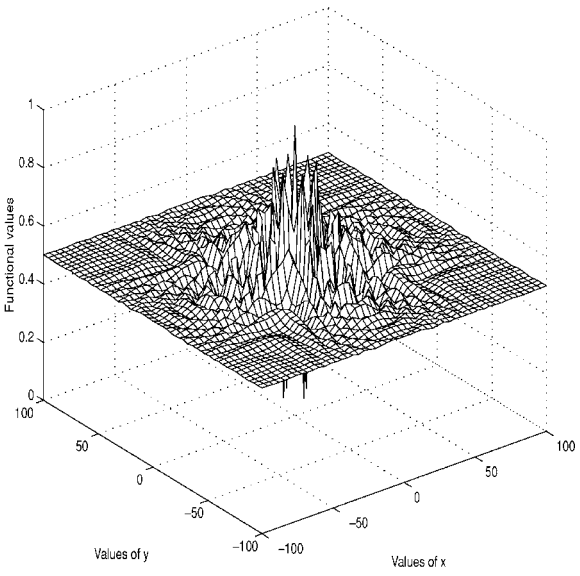

Figure 4. Sketch of the function $0.5 - \frac{\{\sin(\sqrt{(x^2+y^2)})^2 - 0.5}{(1.0+0.001(x^2+y^2))^2}$ ($x \in [-100, 100]$, $y \in [-100, 100]$).

Binary coding is used for chromosomes. Substring length for each parameter (variable) has been taken as 22 and that for α, β, and γ as 10. Each chromosome or string is a concatenation of binary substrings, generated randomly, of the parameters of the optimization problem to be solved. Let the substring length of each parameter be l. Let the domain of the parameter be $[l_{lt}, u_{lt}]$. The substring *str* is decoded into [0,1] and mapped into the domain of the parameter. The decoded value $v([0, 1])$ of *str* is obtained from

$$v = \frac{2^l}{2^l - 1} \sum_{i=1}^{l} \frac{str_i}{2^i}, \quad 0 \le v \le 1.0, \tag{11}$$

where str_i is the value of the ith bit of *str*. This decoded value is then transformed to v', $v' \in [l_{lt}, u_{lt}]$ with

$$v' = l_{lt} + v(u_{lt} - l_{lt}). \tag{12}$$

The population size is kept fixed at 20 throughout the simulation. (In a part of the experiment, it was also chosen as 50.) The initial population is chosen randomly. Generational replacement technique is used [2].

The objective function is the identity function. It means, fitness of a chromosome is equal to its functional value. Therefore, the higher the functional value is, the better is the chromosome. Both the elitist model [2] and the standard GA (*that is,* non-elitist model) are implemented. In the case of elitism, the lowest fit string of the present generation is replaced by the best fit string of the previous one, if the latter one is better than the best fit string of the present generation.

Linear normalization selection procedure (which works better in a close competitive environment [2]) is adopted. In this technique, instead of considering original fitness values (e.g., 100, 48, 43, 10, 3), normalized fitness values (e.g., 100, 90, 80, 70, 60) are used for selecting chromosomes. The fitness values are normalized using two parameters, namely, the decrement/increment parameter (here, 10) and the maximum/minimum normalized value (here, 100 or 60). Thus these normalized values will decrease/increase linearly. In our experiment, both the difference between successive fitness values and the minimum fitness value have been taken as 1. The number of copies produced by the ith individual (chromosome) with normalized fitness value f_i in a population of size k is taken as $round(c_i)$; where

$$c_i = \frac{kf_i}{\sum_{i=1}^{k} f_i}. \tag{13}$$

$round(c_i)$ gives the nearest integer of the real number c_i.

The crossover and (bitwise) mutation probabilities are taken as 0.8 and 0.008, respectively. Multi-point crossover operation is performed where for each substring (each parameter encoded as a part of the chromosome) the crossover operation is one-point. Hence, the number of crossover sites is taken to be equal to the number of parameters encoded in a chromosome. One crossover site is chosen randomly on each substring. In a part of the investigation, the mutation probability was also varied from 0.001 to 0.1 in the following way.

Generations	1–200	201–400	401–600	601–800	801–1000
Mutation probability	0.1	0.01	0.001	0.01	0.1

The reason for this is as follows. At the initial stage of execution of the GA the chromosomes are assumed to be random in nature. Higher mutation probability leads to generate diverse chromosomes and helps to explore the search space properly. As generation goes, the mutation probability is lowered. Lower mutation probability helps to exploit the search. Further, as execution continues the chromosomes become more and

more homogeneous in nature. To avoid this and the premature convergence, the mutation probability is increased again. The same procedure was also adopted in [16]. Note that, one may use some other procedures to vary the probability of mutation.

It may happen that the fitness value *fit* of a chromosome becomes negative (since the fitness value of a chromosome is taken as the functional value of the chromosome). In that case, to differentiate between the chromosomes having positive and negative fitness values (when (7) is used), we consider $-(fit^r)$ instead of $(-fit)^r$ and $-(fit^r)^{1/r}$ instead of $(-fit^r)^{1/r}$ while computing *MFV*. The value of r (Eq. (7)) is taken as 1 and 2.

The algorithm has been run for 1000 generations in each simulation. Fifty simulations are performed. The same initial populations are taken for all the schemes. Initial populations are different for different simulations.

To maintain consistency with previous studies [15], the parameters considered for De Jong $F4$ minimization are different from those used for $F1$ and $F2$. Here the parameters are taken to be the same as those used in [15] and only the elitist model is considered. Different criteria are taken into account to compare the performance obtained using different schemes mentioned in Section 3. The performance of each of the schemes is evaluated by measuring its ability to detect a solution within a specific accuracy. The parameters used for this function are as follows.

Population size: 10
Crossover probability: 0.7
Mutation probability: 0.005
Substring length corresponding to each parameter: 8 bits
Stopping condition: 2500 generations
Number of simulations: 50
The other parameters taken are equal to those used for optimization of $F1$ and $F2$.

4.1.2. Analysis of Results. Let us now explain the results in terms of mean (taken over fifty simulations) fitness values of the best chromosomes at the last generation using various schemes (Section 3) for the functions $F1$ and $F2$ (Table 1) with $r = 1$ and 2. For convenience, we mention here the said schemes in brief. These are Scheme 0 (conventional genetic algorithm, CGA), Scheme 1 (fixed weighting coefficients), Schemes 2 and 3 (weighting coefficients dependent on mutation), Scheme 4 (adaptive weighting coefficients dependent on the fitness values of parents and children), Scheme 5 (equal weighting coefficients automatically evolved) and Scheme 6 (unequal weighting coefficients automatically evolved). Table 2 shows such

Table 2. Average (over fifty simulations) of maximum fitness values (NE: non-elitism, E: elitism, r: exponent of (7)) and standard deviations from maximum fitness values (Std. dev.).

	$F1$				$F2$			
	NE		E		NE		E	
Scheme no.	Fitness $\times 10^9$	Std. dev. $\times 10^9$	Fitness $\times 10^9$	Std. dev. $\times 10^9$	Fitness	Std. dev.	Fitness	Std. dev.
0	2.23	1.82	2.11	1.84	0.958143	0.052953	0.953111	0.058854
1, $r = 1$	2.60	1.70	2.60	1.70	0.965642	0.053894	0.964476	0.047178
1, $r = 2$	2.48	1.75	2.73	1.64	0.968671	0.043855	0.967184	0.045812
2, $r = 1$	**2.85**	1.57	2.48	1.75	0.960611	0.051774	0.974791	0.035812
2, $r = 2$	2.60	1.70	2.48	1.75	0.970324	0.041188	0.972677	0.044505
4a, $r = 1$	2.60	1.70	2.73	1.64	**0.980903**	0.024643	0.976722	0.034396
4a, $r = 2$	2.60	1.70	2.71	1.63	0.974090	0.036987	0.972567	0.039746
4b, $r = 1$	2.36	1.70	2.97	1.64	0.977907	0.036136	**0.978830**	0.026404
4b, $r = 2$	2.48	1.75	**3.08**	1.34	0.980664	0.025113	0.975721	0.026762
5, $r = 1$	2.36	1.79	2.48	1.75	0.956425	0.049105	0.970551	0.038064
5, $r = 2$	2.36	1.79	2.73	1.64	0.955280	0.072822	0.977399	0.026527
6, $r = 1$	2.36	1.79	2.48	1.75	0.958676	0.047182	0.971053	0.041560
6, $r = 2$	**2.85**	1.87	2.97	1.48	0.955074	0.052986	0.970445	0.045738

results for Schemes 0, 1, 2, 4, 5, and 6. NE and E denote respectively, the non-elitist model and the elitist model. *Std. dev.* denotes the standard deviation of all the best chromosomes obtained at the last generation of different simulations. This provides a comparative study among various schemes based on the performance attained at the end of evolutionary process. It is seen from Table 2 that Scheme 0 performs the worst, except for three cases of $F2$ with NE (e.g., the cases of $r = 1$ and 2 for Scheme 5 and $r = 2$ for Scheme 6). It is mostly Scheme 4 and then Scheme 2 or 6 which are seen to produce the best result (marked bold). Note that Scheme 2 involves fixed weighting coefficient, whereas it is adaptive for Schemes 4 and 6.

The effect of varying mutation probability is shown in Table 3. Here we considered $F2$, as an example, using elitism with Schemes 0, 2, 4b, and 6 only. The performance of Schemes 2, 4, and 6 is seen to be slightly better than that of Scheme 0.

Table 4 demonstrates the performance of the aforesaid best resulting schemes when the population size is increased from 20 to 50. Here too, the conclusion as in Table 3 holds good. Thus it appears that with the increase in population size, the improvement of Schemes 2, 4, and 6 over Scheme 0 decreases.

Figures 5(a) and (b) demonstrate graphically the improvement in performance of the Schemes 2, 4b and 6 over Scheme 0 with generations as per the experimental set-up of Table 2. As an illustration, we consider the case of $r = 1$ with elitism for both $F1$ and $F2$. The curves are drawn using least square fit method. Here the ordinate represents the average (computed over fifty simulations) of maximum fitness values obtained at a generation. In a part of the experiment, we have also plotted the average (computed over fifty simulations) of average fitness values obtained over the population at a generation. Its variation is shown in Fig. 6 for the function $F1$, as a typical illustration. From these figures Scheme 4b is seen to be the best. One may note in this connection that the version 4a also produces comparable results.

Results of Schemes 0, 2, 4b and 6 on De Jong $F4$, with Gaussian noise injected on it, is presented in Table 5. *Percent solved* represents the percentage of fifty simulations where the functional value was $\leq -2\sigma$ (where σ denotes standard deviations of added noise). The *noisy average best* denotes the average best solution (over fifty runs) found after 2500 generations with respect to the noisy function evaluation value. The *true average best* is the average of the true function evaluation value (without noise) for the same population evaluated for the *noisy average best* value. The smaller the average functional value is, the better is the corresponding chromosome. The last column of this table indicates the minimum average (over fifty runs) number of generations required to attain a value $\leq -2\sigma$ using noisy evaluation. The entry '—' in this column

Table 3. Average (over fifty simulations) of maximum fitness values (r: exponent of (7)) for $F2$ using elitism: variable mutation probability.

Scheme no.						
	2		4b		6	
0	$r = 1$	$r = 2$	$r = 1$	$r = 2$	$r = 1$	$r = 2$
0.991580	0.991861	0.992551	0.992775	0.992875	0.993199	0.992227

Table 4. Average (over fifty simulations) of maximum fitness values (NE: non-elitism, E: elitism, r: exponent of (7)) for $F2$: population size = 50.

	Scheme no.						
		2		4b		6	
Model	0	$r = 1$	$r = 2$	$r = 1$	$r = 2$	$r = 1$	$r = 2$
NE	0.981849	0.987065	0.983719	0.991950	0.983112	0.984962	0.991033
E	0.983398	0.984697	0.985565	0.978694	0.981722	0.992282	0.989668

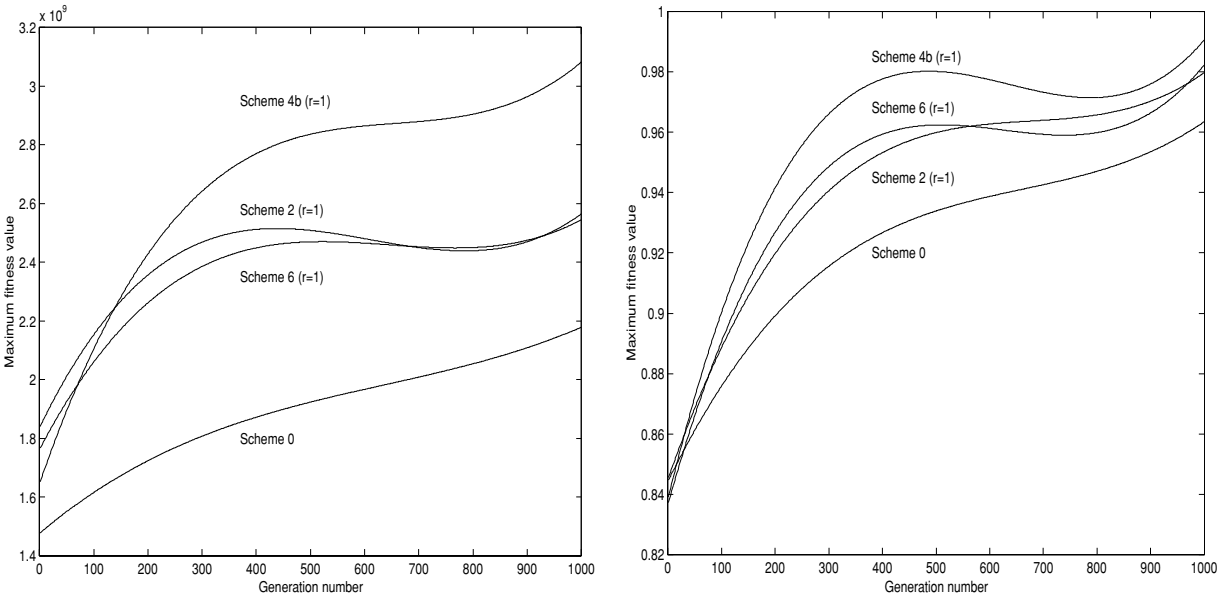

Figure 5. Variation of maximum fitness value over generations corresponding to Scheme 0, Scheme 2 ($r = 1$), Scheme 4b ($r = 1$), and Scheme 6 ($r = 1$) using the elitist model for the function: (a) F1; (b) F2.

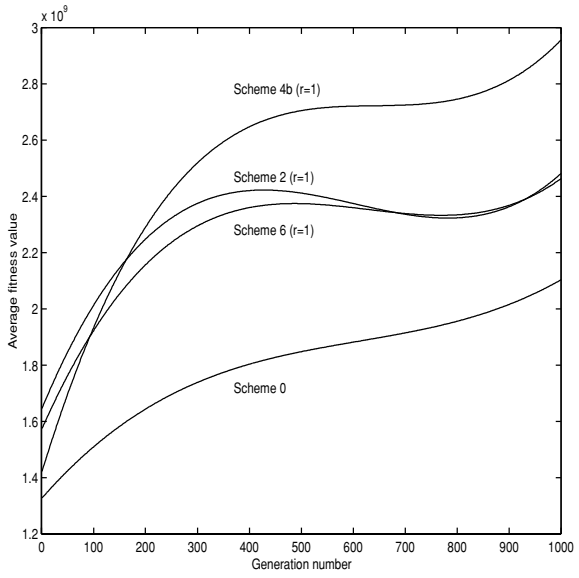

Figure 6. Variation of average fitness value over generations corresponding to Scheme 0, Scheme 2 ($r = 1$), Scheme 4b ($r = 1$), and Scheme 6 ($r = 1$) using the elitist model for the function $F1$.

means that the above mentioned value was not found within 2500 generations. Here, in five out of six cases (Table 5), MGA performed better than CGA even for noisy environment.

In a part of our study, instead of using linear normalization selection procedure and multi-point crossover operator, we used roulette wheel selection process and single point crossover operator. Comparison between them with respect to CGA and MGA is studied. As an illustration some of the results are put in Table 6. The results further strengthen the superiority of MGA.

Furthermore, a comparative study is made with some enhanced variants of GA, namely, Eshelman's CHC [17] (CHC is chosen because of its robust performance on a wide variety of problems). Some of these results are shown in Table 7 for illustration. As expected, CHC performs better than CGA; and Scheme 4b of MGA

Table 5. Results of De Jong $F4$ using elitism (r: exponent of (7)).

Scheme no.	Percent solved	Average best (Noisy)	Average best (True)	-2σ obtained at generation
0	48.0	3.63	7.18	—
2, $r = 1$	58.0	3.49	6.97	2134
2, $r = 2$	50.0	3.37	6.81	2428
4*b*, $r = 1$	38.0	4.08	7.85	—
4*b*, $r = 2$	64.0	3.50	6.95	1479
6, $r = 1$	58.0	3.55	7.04	2282
6, $r = 2$	66.0	3.25	6.94	1746

Table 6. Average (over fifty simulations) of maximum fitness values for F2 (r: exponent of (7)) using elitism, roulette wheel selection and single point crossover.

Scheme no.						
	2		4b		6	
0	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
0.986979	0.991743	0.987460	0.993254	0.994410	0.988290	0.992013

Table 7. Average (over fifty simulations) of maximum fitness values for F2 (NE: non-elitism, E: elitism, r: exponent of (7)).

	Scheme no.				
		2	4b	6	
Model	0	$r=1$	$r=1$	$r=1$	CHC
NE	0.973921	0.974337	0.982587	0.975642	0.975109
E	0.976982	0.978812	0.985606	0.978624	0.980236

produces the best result. Moreover, CHC is computationally more intensive than MGA.

4.2. *Selection of Multi Layer Perceptron (MLP) Parameters for an XOR Problem*

To determine an optimal set of connection weights and thresholds in an MLP for classification problem, the overall error, that needs to be minimized, is defined as [10]

$$\text{Error} = \frac{1}{s}\sum_{s}\sum_{j=1}^{out}(t_{sj} - V_{sj})^2, \tag{14}$$

where s and out represent the number of training samples and the number of neurons in the output layer, respectively. Variable t_{sj} and V_{sj} denote the target and obtained output for the jth neuron, (*that is,* the activation of the jth output neuron) respectively, corresponding to the sth training pattern. The weights and thresholds are modified (using GAs) so that *Error* is minimized. Each neuron j in the output and hidden layers is associated with a set of p' input values I_{ij}, $1 \le i \le p'$, a threshold value θ_j, a set of interconnection weights w_{ij}, an activation function (which is taken as sigmoidal), and an output value

$$V_{sj} = \frac{1}{1 + \exp(I_{ij}w_{ij} - \theta_j)}. \tag{15}$$

4.2.1. Experimental Set-up. Input values to the network are in binary form. Total number of patterns in the data set is 148. These patterns are generated by replicating the four input-output patterns of two-input XOR function 32 times. The size of the training set is considered to be 10%, 20%, 30% and 40% of the data set and these samples are taken randomly. In this problem, there are two neurons in the hidden layer. Since it is a two-class problem, the number of neurons in the output layer is 2. Hence, the total number of parameters of the problem (including threshold values of the neurons of hidden layer and output layer) $= 2 \times 2 + 2 \times 2 + 2 + 2 = 12$. Substring length of each parameter e.g., w_{ij}, θ_j has been taken as 10. Values of these parameters lie in $[-25, +25]$ and these can be obtained by decoding the substrings using Eqs. (11) and (12). Thus each string represents a set of weights and thresholds corresponding to a complete network. The objective function to be minimized is the error value. The lower this value is, the higher is the fitness. The population size is kept fixed at 20. The crossover and mutation probabilities are taken as 0.8 and 0.008, respectively. Equation (13) is used for selecting chromosomes. The algorithm is also run, like the previous experiments, with fifty different initial populations.

4.2.2. Analysis of Results. Table 8 shows the average (over fifty simulations) error values of the best chromosomes in the last generation obtained for different training sets using Schemes 0, 2, 4 and 6 for the problem of selecting MLP parameters. Here, in six cases out of eight, Scheme 4 (either Scheme 4a or Scheme 4b) is seen to be the best.

4.3. *Selection of Hopfield type Network Architecture for Object Extraction*

For determining the optimum architecture of Hopfield type neural network for object extraction, we

Table 8. Minimum error value (averaged over fifty simulations) for MLP problem (NE: non-elitism, E: elitism, r: exponent of (7)).

Scheme no.	Training sample used							
	10%		20%		30%		40%	
	NE	E	NE	E	NE	E	NE	E
0	1.168863	1.295544	2.083853	1.722724	2.756636	3.877250	5.555961	3.635180
2, $r = 1$	1.107500	0.846727	1.588272	1.397355	1.448601	3.022454	5.207902	5.014730
2, $r = 2$	1.081293	1.189643	2.049644	2.916845	2.708330	3.427654	4.433890	2.126972
4a, $r = 1$	0.905767	1.136387	2.658492	2.100025	1.855749	**1.347899**	2.913776	**1.727618**
4a, $r = 2$	0.770073	1.054493	1.342199	1.915655	**1.207395**	2.026888	3.323396	2.090645
4b, $r = 1$	0.622872	0.911022	1.362805	**1.126525**	2.012772	2.990306	5.055459	3.314264
4b, $r = 2$	**0.572429**	0.921926	**1.274718**	1.505582	2.764329	3.051799	4.739758	3.446146
6, $r = 1$	1.094390	**0.776514**	2.169206	1.221610	3.288752	3.287268	**2.347590**	3.438966
6, $r = 2$	1.258785	1.402526	2.676056	2.676635	3.208401	1.889062	4.061546	2.907758

considered various noisy images as input. In order to demonstrate the effectiveness of the proposed concept of ancestors' influence, we have considered here only Scheme 4b, as it is seen to produce the best overall performance compared to others.

4.3.1. Experimental Set-up. The original image (Fig. 7) is a synthetic binary (two-tone) one and is of size 40×40. Two different noisy versions are generated by adding Gaussian noise ($N(0, \sigma^2)$), with mean value zero and standard deviation $\sigma = 20$ and 32, to each pixel of this binary image. The range of pixel value is [1, 32].

For an $m1 \times n1$ image, each pixel (neuron) being connected to at most $k1$ of its neighbors, the length of a chromosome is $m1 \times n1 \times k1$. Each bit of the chromosome represents W_{ij}. If a neuron is connected to any of its neighbors, the corresponding bit of the chromosome (i.e., value of W_{ij}) is set to 1, else 0. Hence for an image of size 40×40, a binary string of length $40 \times 40 \times 8$ (here we consider maximum 8 neighbors for a pixel) is used for chromosome representation. Each string represents a possible network architecture for object extraction. Since the number of parameters to be determined here is very large, we considered a population size of 30. Fitness of a chromosome is taken as a function of the energy value (Appendix B) of the (converged) network. The lower the energy value is, the better is the corresponding chromosome. The crossover probability is taken as 0.8. The mutation probability is chosen as 0.002. Selection procedure is the same as in previous experiments (*that is,* Eq. (13) is used).

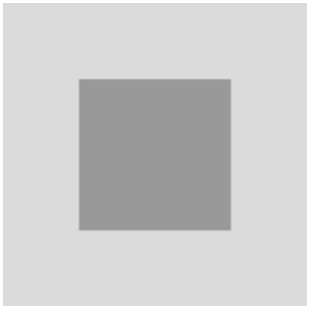

Figure 7. Original synthetic image.

4.3.2. Analysis of Results. Figures 8(a) and (b) show the improvement of average fitness value of the network with generations using Scheme 4b with respect to Scheme 0 for $\sigma = 20$ and $\sigma = 32$, respectively. Here too, the curves are drawn using least square fit method. For a typical illustration the noisy input for $\sigma = 20$ and the corresponding outputs using Scheme 0 and Scheme 4b ($r = 1$) are shown in Figs. 9(a)–(c), respectively. Since this problem is computationally intensive five simulations are considered. It is seen that the percentage of correct classification of pixels are 97.4375 and 98.2500 corresponding to Figs. 9(b) and (c). Moreover, the upper and the right edges of the object are better preserved in Fig. 9(c).

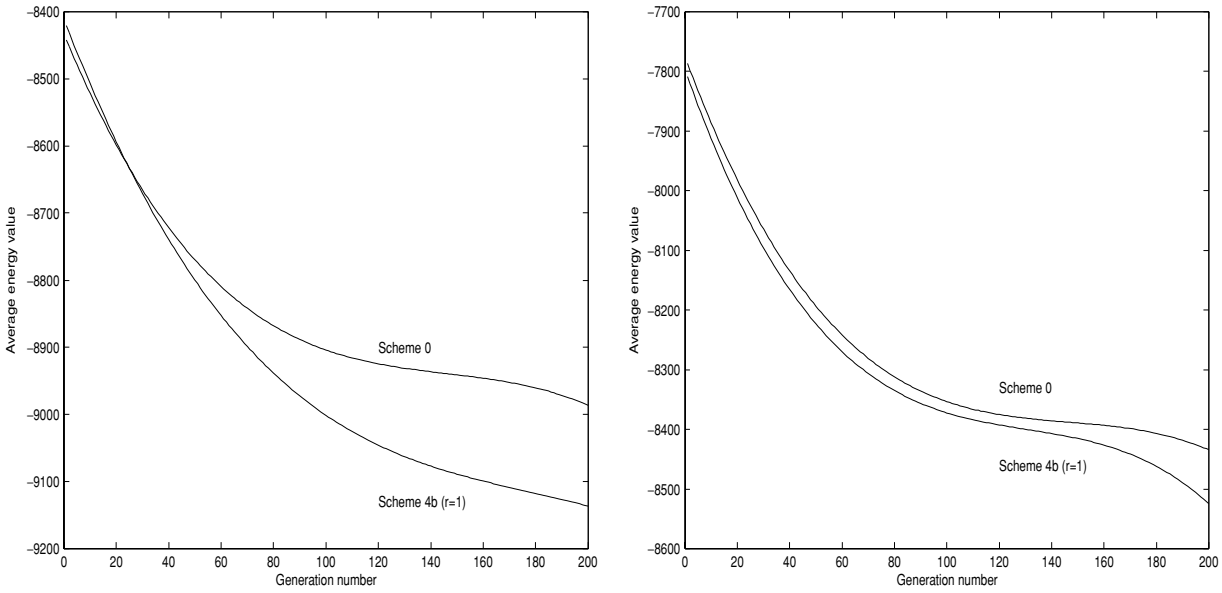

Figure 8. Variation of average fitness value over generations for Scheme 0 and Scheme 4b ($r = 1$) using the non-elitist model for noisy image with (a) $\sigma = 20$; (b) $\sigma = 32$.

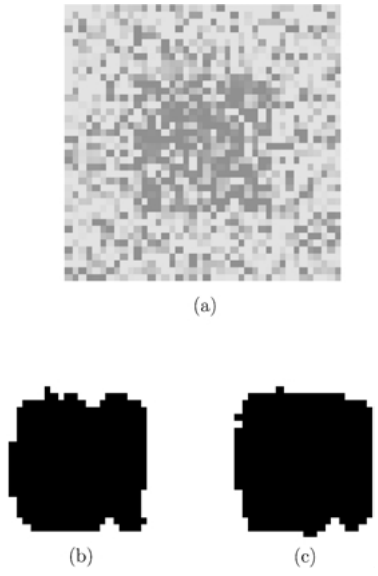

Figure 9. (a) Noisy version ($\sigma = 20$) of Fig. 7; (b) output using Scheme 0 considering Fig. 9(a) as input; and (c) output using Scheme 4b ($r = 1$) considering Fig. 9(a) as input.

5. Conclusions

A new fitness evaluation criterion for GAs has been introduced by considering the effect of fitness of ancestors (predecessors) in addition to the fitness of the individual itself. Selection of chromosomes is made based on these modified fitness values. The Schema Theorem for this new model is derived. Some conditions (Eqs. (21) and (25) of Appendix A) are found where the proposed concept leads to superior performance compared to the conventional GA in terms of the lower bound of the number of instances of a good schema in subsequent generations. Different schemes are provided considering the amount of weight, to be given to the ancestors, either in a fixed or adaptive manner, both automatically and manually.

As the population size increases, the improvement of the proposed schemes over the CGA decreases. The method also works better for noisy environments and complex problem domains including preservation of shape (edges) of image regions.

Scheme 2 (mutation dependent fixed weighting coefficient), Scheme 4 (dynamic weighting coefficient, dependent on the difference of fitness values between parents and children) and Scheme 6 (automatically evolved dynamic weighting coefficient) are given more emphasis from the point of conducting experiments, because of their relatively improved performance. Although the adaptive methods are computationally more

intensive, they are seen to have an edge over the fixed ones even for noisy and complex functions. Note that unlike Scheme 4, Scheme 6 does not consider the difference between the fitness values of parents and children, though the parents are given unequal weighting coefficients in both the schemes under an adaptive framework. Overall, it is Scheme 4 which is seen to provide superior results.

Although we have considered the value of r to be 1 and 2, one can take, as mentioned in Section 3, any value $r \geq 1$ for conducting experiments. Results under Scheme 7 are not included because the influence of grandparents was seen to be not much effective in enhancing the performance. Moreover, in a part of the investigation, the new criterion of fitness evaluation is also tested using gray coding and the results are found to corroborate our above mentioned findings. In this context it may be mentioned that this new criterion may sometimes lead to a premature convergence.

Appendix A

Influence of Parents on Offspring and the Schema Theorem

The Schema Theorem [1] estimates the lower bound of the number of instances of different schemata at any point of time. According to this theorem, a short-length, low-order, above-average schema will receive exponentially increasing instances in subsequent generations at the expense of below average ones. In this section we derive the Schema Theorem for GAs using the aforesaid *MFV* (let us call it 'Modified Genetic Algorithms' (*MGA*s)) and find the lower bound of the number of instances of a schema. We also compare this bound with that of the CGA. Before deriving the theorem let us first introduce some definitions.

A *schema* is a similarity template describing a subset of strings with similarities at certain string positions. As an example, $^{**}10^{*}1^{*}$ is a schema where '$*$' indicates that the corresponding positions may be either 1 or 0 (considering binary strings). Then 1010111 and 1110011 are two instances (chromosomes) of this schema. The number of fixed bits of a schema is the *order* of that schema. The distance between the first and the last fixed positions is termed as *defining length* of the schema.

For the sake of convenience, let us first show the derivation of the Schema Theorem for CGA [1]. The notations that we will be using are listed below:

h: a short-length, low-order, above-average schema
$\delta(h)$: the defining length of schema h
$o(h)$: order of schema h
L: length of a chromosome
k: size of the population
$m(h, t)$: number of instances of a schema h in a population at generation t for the CGA
$\bar{f}$: average fitness value of the population for the CGA
$\bar{f}_h$: average fitness value of the strings representing schema h for the CGA
p_c: crossover probability
p_{mut}: mutation probability

Let m instances of a particular schema h exist in the population at time t (denoted as $m(h, t)$). Now selection process copies each string into the mating pool according to its fitness value. In a population of size k the ith string (where, $1 \leq i \leq k$) A_i, with fitness value f_i, gets selected with probability $prob = \frac{f_i}{\sum_{j=1}^{j=k} f_j}$. A non-overlapping population of size k is produced with replacement from the population at time t. Hence in the population at time $(t + 1)$ the number of instances of schema h (denoted as $m(h, t + 1)$) is

$$m(h, t+1) = m(h, t) k \frac{\bar{f}_h}{\sum_k f_j} = m(h, t) \frac{\bar{f}_h}{\bar{f}}.$$

Hence the number of above average schemata will grow exponentially, and below average ones will receive a decreasing number of samples.

If a crossover site is selected uniformly at random among $(L - 1)$ possible sites, a schema h will be destroyed with a probability

$$p_d = \frac{\delta(h)}{(L-1)}.$$

Thus, the survival probability (p_s), when the crossover site falls outside the defining length, is

$$p_s = 1 - p_d = 1 - \frac{\delta(h)}{(L-1)}.$$

If p_c is the crossover probability then,

$$p_s \geq 1 - p_c \frac{\delta(h)}{(L-1)}.$$

Moreover, in order to survive a schema h, all the fixed positions of h ($o(h)$) should remain unaltered. If p_{mut} is the mutation probability, a single allele survives with a probability $(1 - p_{mut})$. Hence, for $o(h)$ number of fixed positions of a schema h to survive, the survival probability is

$$(1 - p_{mut})^{o(h)}.$$

If $p_{mut} \ll 1$, the above value is $(1 - o(h)p_{mut})$. Therefore,

$$m(h, t+1) \geq m(h, t)\frac{\bar{f}_h}{\bar{f}}\{1 - p_c\delta(h)/(L-1)\} \times \{1 - o(h)p_{mut}\}. \quad (16)$$

Neglecting the small cross-product term, we have

$$m(h, t+1) \geq m(h, t)\frac{\bar{f}_h}{\bar{f}} \times \{1 - p_c\delta(h)/(L-1) - o(h)p_{mut}\}. \quad (17)$$

We now derive the expression for the expected number of instances of schema h, *that is,* we determine $m(h, t+2)$ from $m(h, t+1)$ for the MGA considering the influence of parents only. For deriving the theorem we have computed the fitness distributions of the populations both before and after the selection procedure in each generation. Let the time instants 'before selection' and 'after selection' of the tth generation be denoted as (t^{bs}) and (t^{as}), respectively. The other notations that we will be using for this purpose are listed below:

$m^*(h, t)$: number of instances of a schema h in a population at generation t for the MGA
$\bar{f}(t^{as})$: average fitness value of the population at t^{as} for the CGA
$\bar{f}^*(t^{as})$: average fitness value of the population at t^{as} for the MGA
$\bar{f}_h(t^{as})$: average fitness value of schema h at t^{as} for the CGA
$\bar{f}^*_h(t^{as})$: average fitness value of schema h at t^{as} for the MGA
$\bar{f}(t+1)^{bs}$: average fitness value of the population at $(t+1)^{bs}$ for the CGA
$\bar{f}^*(t+1)^{bs}$: average fitness value of the population at $(t+1)^{bs}$ for the MGA
$\bar{f}_h(t+1)^{bs}$: average fitness value of schema h at $(t+1)^{bs}$ for the CGA
$\bar{f}^*_h(t+1)^{bs}$: average fitness value of schema h at $(t+1)^{bs}$ for the MGA
$\bar{f}^{p1}_h$: average fitness value of schema h (at t^{as}) of the first parent for the MGA
$\bar{f}^{p2}_h$: average fitness value of schema h (at t^{as}) of the second parent for the MGA
f_i: fitness value of ith chromosome at $(t+1)^{bs}$ for the CGA
f^{p1}_i: fitness value of the first parent (at t^{as}) of ith chromosome
f^{p2}_i: fitness value of the second parent (at t^{as}) of ith chromosome
f^*_i: modified fitness value of ith chromosome at $(t+1)^{bs}$ for the MGA

For the CGA, the expected number of instances of schema h obtained from $m(h, t+1)$ can be written from (17) as

$$m(h, t+2) \geq m(h, t+1)\frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} \times \left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\}. \quad (18)$$

In the MGA, the *MFV* of the ith $(\forall i = 1, \ldots, k)$ chromosome is calculated based on its own fitness value and its ancestors' (here, $p1$ and $p2$) fitness values as

$$f^*_i = g\big(f_i, f^{p1}_i, f^{p2}_i\big),$$

where g is the fitness function. The *MFV* of each chromosome changes the average fitness value of the population (at $(t+1)^{bs}$). Let the modified average fitness value be $\bar{f}^*(t+1)^{bs}$. Then

$$\bar{f}^*(t+1)^{bs} = \frac{\sum_{i=1}^{i=k} f^*_i}{k}.$$

(Note that for the CGA, $\bar{f}(t+1)^{bs} = \sum_{i=1}^{i=k} / f_i / k$.)

The *MFV* of chromosomes also changes the average fitness value of the schema. The fitness values of the parent chromosomes at $(t+1)^{bs}$ are obtained from t^{as}. Hence, the modified average fitness value of schema h ($\bar{f}^*_h(t+1)^{bs}$) can be represented as

$$\bar{f}^*_h(t+1)^{bs} = g(\bar{f}_h(t+1)^{bs},\ \bar{f}_h(t^{as})). \quad (19)$$

Chromosomes from stage $(t+1)^{bs}$ are selected based on their both modified fitness values and the modified average fitness value $\bar{f}^*(t+1)^{bs}$. Thus, the probability of selection of each chromosome for the MGA may be different from that of the corresponding chromosome of the CGA.

The expected number of instances of the schema h at $(t+2)^{bs}$ for the MGA will therefore be

$$m^*(h,t+2) \geq m(h,t+1)\frac{g(\bar{f}_h(t+1)^{bs},\bar{f}_h(t^{as}))}{\bar{f}^*(t+1)^{bs}} \times \left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\}. \quad (20)$$

Now we will derive various conditions for different values of r (exponent of (7)) under which the right hand side of (18) will be smaller than that of (20), *that is,* the expected number of instances of schema h in the CGA will be smaller than that of the MGA. Here we consider the general form of g as in (7).

Case 1: $r = 1$:

$$\begin{aligned}
&\frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}}\left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\} \\
&< \frac{g(\bar{f}_h(t+1)^{bs},\bar{f}_h(t^{as}))}{\bar{f}^*(t+1)^{bs}} \\
&\quad \times \left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\} \\
&\Leftrightarrow \frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} < \frac{g(\bar{f}_h(t+1)^{bs},\bar{f}_h(t^{as}))}{\bar{f}^*(t+1)^{bs}} \\
&\Leftrightarrow \frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} < \frac{\alpha\bar{f}_h(t+1)^{bs} + (\beta_1+\beta_2)\bar{f}_h(t^{as})}{\alpha\bar{f}(t+1)^{bs} + (\beta_1+\beta_2)\bar{f}(t^{as})} \\
&\quad \text{(using (3) and any } \alpha,\ 0<\alpha<1) \\
&\Leftrightarrow \frac{\alpha\bar{f}_h(t+1)^{bs}}{\alpha\bar{f}(t+1)^{bs}} < \frac{\alpha\bar{f}_h(t+1)^{bs} + (1-\alpha)\bar{f}_h(t^{as})}{\alpha\bar{f}(t+1)^{bs} + (1-\alpha)\bar{f}(t^{as})} \\
&\Leftrightarrow \frac{\alpha\bar{f}_h(t+1)^{bs}}{\alpha\bar{f}(t+1)^{bs}} < \frac{(1-\alpha)\bar{f}_h(t^{as})}{(1-\alpha)\bar{f}(t^{as})} \\
&\quad \left[\text{since, } \frac{a}{b} < \frac{a+c}{b+d} \Rightarrow \frac{a}{b} < \frac{c}{d}\right] \\
&\Leftrightarrow \frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} < \frac{\bar{f}_h(t^{as})}{\bar{f}(t^{as})}. \qquad (21)
\end{aligned}$$

Equation (21) means that if the proportion of average fitness value of the strings representing schema h with respect to the average fitness value of the population in a mating pool (at t^{as}) in CGA is greater than that after performing crossover and mutation operation (at $(t+1)^{bs}$), then the performance of MGA in terms of the number of instances of schema h will be better than that in CGA. Intuitively, this condition is likely to be satisfied in many problems since due to crossover and mutation some of the instances of schema h would always get disrupted. The possibility of such a disruption is smaller in the case of MGA since more bias is given here to keep schema undisrupted through ancestors.

Case 2: $r = 2$:

$$\begin{aligned}
&\frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}}\left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\} \\
&< \frac{g(\bar{f}_h(t+1)^{bs},\bar{f}_h(t^{as}))}{\bar{f}^*(t+1)^{bs}}\left\{1 - p_c\frac{\delta(h)}{L-1} - o(h)p_{mut}\right\} \\
&\Leftrightarrow \frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} < \frac{g(\bar{f}_h(t+1)^{bs},\bar{f}_h(t^{as}))}{\bar{f}^*(t+1)^{bs}} \\
&\Leftrightarrow \frac{\bar{f}_h(t+1)^{bs}}{\bar{f}(t+1)^{bs}} \\
&< \frac{\left\{\alpha(\bar{f}_h(t+1)^{bs})^2 + \beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2\right\}^{\frac{1}{2}}}{\frac{1}{k}\sum_{i=1}^{k}\left\{\alpha\left(f_i^2\right) + \beta_1\left(f_i^{p_1}\right)^2 + \beta_2\left(f_i^{p_2}\right)^2\right\}^{\frac{1}{2}}} \\
&\quad \text{(using (7))} \\
&\Leftrightarrow \frac{(\bar{f}_h(t+1)^{bs})^2}{(\bar{f}(t+1)^{bs})^2} \\
&< \frac{\alpha(\bar{f}_h(t+1)^{bs})^2 + \beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2}{\left\{\frac{1}{k}\sum_{i=1}^{k}\left\{\alpha\left(f_i^2\right) + \beta_1\left(f_i^{p_1}\right)^2 + \beta_2\left(f_i^{p_2}\right)^2\right\}^{\frac{1}{2}}\right\}^2}. \qquad (22)
\end{aligned}$$

It is of the form

$$\frac{X1}{X2} < \frac{X3}{X4},$$

where

$$\begin{aligned}
X1 &= (\bar{f}_h(t+1)^{bs})^2, \\
X2 &= (\bar{f}(t+1)^{bs})^2, \\
X3 &= \alpha(\bar{f}_h(t+1)^{bs})^2 + \beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2, \quad \text{and} \\
X4 &= \left\{\frac{1}{k}\sum_{i=1}^{k}\left\{\alpha\left(f_i^2\right) + \beta_1\left(f_i^{p_1}\right)^2 + \beta_2\left(f_i^{p_2}\right)^2\right\}^{\frac{1}{2}}\right\}^2.
\end{aligned}$$

Let us simplify the components.

$$\begin{aligned} X2 &= (\bar{f}(t+1)^{bs})^2 \\ &= \left(\frac{f_1+f_2+\cdots+f_k}{k}\right)^2 \\ &= \frac{1}{k^2}(f_1+f_2+\cdots+f_k)^2 \\ &= \frac{1}{k^2}\left[f_1^2+f_2^2+\cdots+f_k^2+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k} f_i f_j\right] \\ &= \frac{1}{k^2}D, \end{aligned}$$

where

$$D = \left[f_1^2+f_2^2+\cdots+f_k^2+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k} f_i f_j\right].$$

Let us denote

$$\left(\alpha\left(f_i^2\right)+\beta_1\left(f_i^{p_1}\right)^2+\beta_2\left(f_i^{p_2}\right)^2\right)^{\frac{1}{2}} = E_i.$$

Now $X4$ can be written as

$$\begin{aligned} X4 &= \left\{\frac{1}{k}\sum_{i=1}^{k}\left\{\alpha\left(f_i^2\right)+\beta_1\left(f_i^{p_1}\right)^2+\beta_2\left(f_i^{p_2}\right)^2\right\}^{\frac{1}{2}}\right\}^2 \\ &= \frac{1}{k^2}\left[(E_1+E_2+\cdots+E_k)^2\right] \\ &= \frac{1}{k^2}\left[E_1^2+E_2^2+\cdots+E_k^2+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k} E_i E_j\right]. \end{aligned} \tag{23}$$

Substituting the values of E_i in (23) we obtain

$$\begin{aligned} X4 = \frac{1}{k^2}\Bigg[&\alpha\sum_{i=1}^{k}(f_i)^2+\beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2 \\ &+\beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k}\left\{\alpha\left(f_i^2\right)\right. \\ &\left.+\beta_1\left(f_i^{p_1}\right)^2+\beta_2\left(f_i^{p_2}\right)^2\right\}^{\frac{1}{2}}\left\{\alpha\left(f_j^2\right)\right. \\ &\left.+\beta_1\left(f_j^{p_1}\right)^2+\beta_2\left(f_j^{p_2}\right)^2\right\}^{\frac{1}{2}}\Bigg]. \end{aligned} \tag{24}$$

Now

$$E_i^2 = \left\{\alpha\left(f_i^2\right)+\beta_1\left(f_i^{p_1}\right)^2+\beta_2\left(f_i^{p_2}\right)^2\right\}.$$

Hence

$$\begin{aligned} E_i E_j = \big[&\alpha^2 f_i^2 f_j^2+\beta_1^2\left(f_i^{p_1}\right)^2\left(f_j^{p_1}\right)^2 \\ &+\beta_2^2\left(f_i^{p_2}\right)^2\left(f_j^{p_2}\right)^2+F\big]^{\frac{1}{2}}, \end{aligned}$$

where

$$\begin{aligned} F = &\alpha\beta_1(f_i)^2\left(f_j^{p_1}\right)^2+\alpha\beta_2(f_i)^2\left(f_j^{p_2}\right)^2 \\ &+\alpha\beta_1\left(f_i^{p_1}\right)^2(f_j)^2+\beta_1\beta_2\left(f_i^{p_1}\right)^2\left(f_j^{p_2}\right)^2 \\ &+\alpha\beta_2\left(f_i^{p_2}\right)^2(f_j)^2+\beta_1\beta_2\left(f_i^{p_2}\right)^2\left(f_j^{p_1}\right)^2. \end{aligned}$$

or

$$E_i E_j = \left(\alpha^2 f_i^2 f_j^2+G\right)^{1/2},$$

where

$$G = \beta_1^2\left(f_i^{p_1}\right)^2\left(f_j^{p_1}\right)^2+\beta_2^2\left(f_i^{p_2}\right)^2\left(f_j^{p_2}\right)^2+F.$$

or

$$E_i E_j = \alpha f_i f_j + H,$$

where

$$H = E_i E_j \mp \sqrt{E_i^2 E_j^2 - G}.$$

Therefore, from (24)

$$\begin{aligned} X4 = \frac{1}{k^2}\Bigg[&\alpha\sum_{i=1}^{k}(f_i)^2+\beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2+\beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2 \\ &+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k}(\alpha f_i f_j+H)\Bigg] \end{aligned}$$

or

$$\begin{aligned} X4 = \frac{1}{k^2}\Bigg[&\alpha D+\beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2+\beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2 \\ &+2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k}H\Bigg] \end{aligned}$$

or

$$X4 = \frac{1}{k^2}\left[\alpha D+\beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2+\beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2+H'\right],$$

where

$$H' = 2\sum_{i=1}^{k}\sum_{\substack{j=1\\ i\neq j}}^{k} H.$$

Substituting the values of $X2$ and $X4$ in (22) we get

$$\begin{aligned}
&\frac{\alpha(\bar{f}_h(t+1)^{bs})^2}{\alpha\left(\frac{1}{k^2}\right)D} \\
&\quad< \frac{\alpha(\bar{f}_h(t+1)^{bs})^2 + \beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2}{\left(\frac{1}{k^2}\right)\left[\alpha D + \beta_1\sum_{i=1}^{k}(f_i^{p_1 2}) + \beta_2\sum_{i=1}^{k}(f_i^{p_2})^2 + H'\right]} \\
&\Leftrightarrow \frac{\alpha(\bar{f}_h(t+1)^{bs})^2}{\alpha D} \\
&\quad< \frac{\alpha(\bar{f}_h(t+1)^{bs})^2 + \beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2}{\alpha D + \beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2 + \beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2 + H'} \\
&\Leftrightarrow \frac{(\bar{f}_h(t+1)^{bs})^2}{D} \\
&\quad< \frac{\beta_1\left(\bar{f}_h^{p_1}\right)^2 + \beta_2\left(\bar{f}_h^{p_2}\right)^2}{\beta_1\sum_{i=1}^{k}\left(f_i^{p_1}\right)^2 + \beta_2\sum_{i=1}^{k}\left(f_i^{p_2}\right)^2 + H'}. \qquad (25)
\end{aligned}$$

Thus the right hand side of (18) will be less than that of (20) if (25) holds.

Similarly, the expressions for $r > 2$ can be deduced. One may note that the right hand side of (21) is independent of α and β_i, whereas it is not the case for (25).

Appendix B

Hopfield type Neural Network Architecture for Object Extraction

To use a Hopfield type neural network for object background classification [18], a neuron is assigned corresponding to every pixel. Each neuron can be connected to all of its neighbors (over a window) only. The connection can be full (a neuron is connected to all of its neighbors) or can be partial (a neuron may not be connected with all of its neighbors). The network topology for a fully connected third order neighborhood is depicted in Fig. 10. Here the maximum number of connections of a neuron with its neighbors is 8. In practice, all these connections may not exist. Again, different neurons may have different connectivity configuration within its neighbors. The initial status and input bias of each neuron are set depending on the gray value of the corresponding pixel. The status updating rules are similar to those of Hopfield's model [19]. The objective function to be minimized for object extraction is similar to the expression of energy of the above mentioned network.

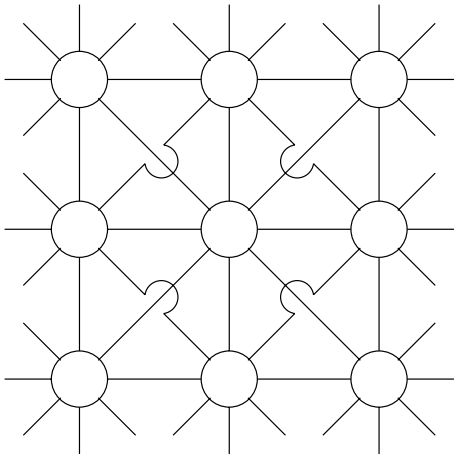

Figure 10. Topology of the neural network with third order connectivity (in the proposed system all connections may not exist).

The energy function of this model has two parts. The first part is due to the local field or local feedback and the second part corresponds to the input bias of the neurons. In terms of images, the first part can be viewed as the impact of the gray levels of the neighboring pixels, whereas the second part can be attributed to the gray value of the pixel under consideration. The total energy contributed by all pixel pairs will be $-\sum_i\sum_j W_{ij}S_iS_j$, where S_i, S_j are the status of the ith and jth neurons, respectively and W_{ij} is the connection strength between these two neurons. In our experimental study W_{ij} is either 0 or 1 (connection is absent or present).

For every neuron i there is an initial input bias B_i which is taken to be proportional to the actual gray level for the corresponding pixel. If the gray value of a pixel is high (low), the corresponding intensity value of the scene is expected to be high (low). The input bias value is taken in the range $[-1, 1]$. Under this framework an ON (1) neuron corresponds to an object pixel and the OFF (-1) one as background pixel. So the threshold between object and background can be taken as 0. Thus the amount of energy contributed by the input bias values is $-\sum_i B_iS_i$. Therefore, the expression of energy for the object extraction problem takes the form

$$\text{Energy} = -\sum_i\sum_j W_{ij}S_iS_j - \sum_i B_iS_i. \qquad (26)$$

From a given initial state, the status of a neuron is modified iteratively to attain a stable state. Stable states of the network (local minima of the energy function) are made to correspond to the partitioning of a scene into compact regions.

References

1. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley: Massachusetts, 1989.
2. L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold: New York, 1991.
3. J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press: Ann Arbor, MI, 1975, 1992.
4. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag: Berlin, 1994.
5. M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press: Cambridge, Massachusetts, 1996.
6. S.K. Pal and P.P. Wang (eds.), *Genetic Algorithms for Pattern Recognition*, CRC Press: Boca Raton, 1996.
7. V.A. McKusick, *Mendelian Inheritance in Man*, John Hopkins University Press: Baltimore, 1992.
8. L.L. Cavalli-Sforza and W.F. Bodmer, *The Genetics of Human Populations*. W.H. Freeman and Company: San Francisco, 1971.
9. D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," *Parallel Computing*, vol. 14, pp. 347–361, 1990.
10. S.K. Pal and D. Bhandari, "Selection of optimal set of weights in a layered network using genetic algorithms," *Information Sciences*, vol. 80, pp. 213–234, 1994.
11. J. Schmidhuber and S. Hochreiter, "Guessing can outperform many long time lag algorithms," *Technical Note IDSIA - 19-96*, pp. 1–3, 1996.
12. S.K. Pal, S. De, and A. Ghosh, "Designing Hopfield type networks using genetic algorithms and its comparison with simulated annealing," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 11, no. 3, pp. 447–461, 1997.
13. D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, Co, 1988, pp. 118–130.
14. G.J.E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers: San Mateo, California, 1991.
15. K. Mathias, D. Whitley, A. Kusuma, and C. Stork, "An empirical evaluation of genetic algorithms on noisy objective functions," in *Genetic algorithms for pattern recognition*, edited by S.K. Pal and P.P. Wang, CRC Press: Boca Raton, pp. 65–86, 1996.
16. S.K. Pal, S. Bandyopadhyay, and C.A. Murthy, "Genetic algorithms for generation of class boundaries," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, no. 6, pp. 816–828, 1998.
17. L.J. Eshelman, "Preventing premature convergence in the genetic algorithms by preventing incest," in *Proceedings 4th International Conference on Genetic Algorithms*, edited by R.K. Belew and L.B. Booker, San Diego, pp. 115–122, 1991.
18. A. Ghosh, N.R. Pal, and S.K. Pal, "Object background classification using Hopfield type neural network," *International Journal of Pattern recognition and Artificial Intelligence*, vol. 6, no. 5, pp. 989–1008, 1992.
19. J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proceedings National Academy of Science*, USA, vol. 81, pp. 3088–3092, 1984.

Susmita Ghosh (nee De) received the B.Sc. degree with Honors in Physics and B.Tech. degree in Computer Science and Engineering from the University of Calcutta, India, in 1988 and 1991, respectively. She obtained the M.Tech. degree in Computer Science and Engineering from the Indian Institute of Technology, Bombay, India in 1993. She worked as a Dr. K.S. Krishnan Senior Research Fellow of the Dept. of Atomic Energy, Govt. of India, from 1994 to February, 1997 in the Indian Statistical Institute, Calcutta towards the Ph.D. degree.

At present, Ms. Ghosh is a Lecturer at the Dept. of Computer Science and Engineering, Jadavpur University, Calcutta, India. Her research interests include *Genetic Algorithms, Neural Networks, Image Processing, Pattern Recognition, and Soft Computing*.

Ashish Ghosh is an Associate Professor of the Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He received the B.E. degree in Electronics and Telecommunication from the Jadavpur University, Calcutta in 1987, and the M.Tech. and Ph.D. degrees in Computer Science from the Indian Statistical Institute, Calcutta in 1989 and 1993, respectively. He received the prestigious and most coveted *Young Scientists* award in Engineering Sciences from the Indian National Science Academy in 1995; and in Computer Science from the Indian Science Congress Association in 1992. He has been selected as an *Associate* of the Indian Academy of Sciences, Bangalore in 1997. He visited the Osaka Prefecture University, Japan with a Post-doctoral fellowship during October 1995 to March 1997; and Hannan University, Japan as a *visiting scholar* during September-October, 1997. During May 1999 he was at the Institute of Automation, Chinese Academy of Sciences, Beijing with CIMPA (France) fellowship. He was at the German National Research Center for Information Technology, Germany, with a German Govt. (DFG)

Fellowship during January–April 2000. He also visited various Universities/Academic Institutes and delivered lectures in different countries including South Korea, Poland, and The Nederland.

His research interests include *Evolutionary Computation, Neural Networks, Image Processing, Fuzzy Sets and Systems, Pattern Recognition, and Data Mining*. He has already published about 45 research papers in internationally reputed journals and referred conferences, and has edited 2 books.

Sankar K. Pal is a *Distinguished Scientist*, and *Founding Head* of Machine Intelligence Unit, at the Indian Statistical Institute, Calcutta. He received the M.Tech. and Ph.D. degrees in Radio Physics and Electronics in 1974 and 1979 respectively, from the University of Calcutta. In 1982 he received another Ph.D. in Electrical Engineering along with DIC from Imperial College, University of London. He worked at the University of California, Berkeley and the University of Maryland, College Park during 1986–87 as a *Fulbright Post-doctoral Visiting Fellow*; at the NASA Johnson Space Center, Houston, Texas during 1990–92 and 1994 as a *Guest Investigator* under the *NRC-NASA Senior Research Associateship program*; and at the Hong Kong Polytechnic University, Hong Kong in 1999 as a *Visiting Professor*. He served as a *Distinguished Visitor of IEEE Computer Society (USA)* for the *Asia-Pacific Region* during 1997–99.

Prof. Pal is a *Fellow* of the IEEE, USA, Third World Academy of Sciences, Italy, and all the four National Academies for Science/Engineering in India. His research interests include Pattern Recognition, Image Processing, Soft Computing, Neural Nets, Genetic Algorithms, and Fuzzy Systems. He is a co-author of six books including *Fuzzy Mathematical Approach* to *Pattern Recognition*, John Wiley (Halsted), N.Y., 1986, and *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley, N.Y. 1999.

He has received the *1990 S.S. Bhatnagar Prize* (which is the most coveted award for a scientist in India), *1993 Jawaharlal Nehru Fellowship 1993, Vikram Sarabhai Research Award*, *1993 NASA Tech Brief Award*, *1994 IEEE Trans. Neural Networks Outstanding Paper Award*, *1995 NASA Patent Application Award*, *1997 IETE—Ram Lal Wadhwa Gold Medal*, *1998 Om Bhasin Foundation Award*, the *1999 G.D. Birla Award for Scientific Research*, and the *2000 Khwarizmi International Award, Iran*.

Prof. Pal is an *Associate Editor*, IEEE Trans. Neural Networks (1994–98), Pattern Recognition Letters, Neurocomputing, Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, and Fundamenta Informaticae; a *Member, Executive Advisory Editorial Board*, IEEE Trans. Fuzzy Systems and Int. Journal of Approximate Reasoning, and a *Guest Editor* of many journals including the IEEE Computer.