Directed Mutation in Genetic Algorithms

DINABANDHU BHANDARI,
NIKHIL R. PAL

and

SANKAR K. PAL

Machine Intelligence Unit, Indian Statistical Institute, Calcutta -700 035, India

Communicated by Abe Kandel

ABSTRACT

Mutation is an important genetic operation that helps to maintain the genetic diversity of the population in order to achieve a good solution to an optimization problem. The conventional mutation, with its random nature and low probability of occurrence, is unable to guide the algorithm for fast convergence. In this article, we propose a new mutation technique called, *directed mutation* following the concept of induced mutation in biological systems. Directed mutation deterministically introduces new points in the population guided (directed) by the information acquired in the previous generations. It expedites the algorithm using an acceleration function which can be based on gradient or extrapolation. The effectiveness of the new technique has been demonstrated through optimization of a set of complex functions, and the results have also been compared with those of the conventional GAs.

1. INTRODUCTION

Genetic algorithms (GAs) [1–3] are probabilistic heuristic search processes based on natural genetic system. They are highly parallel and believed to be robust in searching global optimal solution of complex optimization problems. They recombine structural information to locate new points in the search space with expected improved performance.

GAs are capable of solving wide range of complex optimization problems [2, 4, 5] using three simple genetic operations (selection/reproduction, crossover, and mutation) on coded solutions (strings/chromosomes) for the parameter set, not the parameters themselves, in an iterative fashion. GAs consider several points in the search space simultaneously, which

655 Avenue of the Americas, New York, NY 10010 0020-0255/94/$7.00

reduces the chance of convergence to a local optima. GAs use only the payoff or penalty (i.e., objective) function called, the fitness function and do not need any other auxiliary information. GAs are theoretically and empirically proven to provide robust search in complex spaces [2], even if the functions are not smooth or continuous, which are very difficult (sometimes impossible) to search using calculus based methods.

GAs exploit historical information to speculate on new search points by the crossover operation on a pair of potential strings selected by the reproduction/selection operation. Mutation, on the other hand, is a secondary genetic operation of genetic algorithms. It has a great importance to sustain the genetic diversity, which reduces the chance of getting stuck to a local optimum. Mutation also helps the algorithm to recover information, which is sometimes essential to obtain a good solution, lost in the earlier generations.

As to the authors' knowledge GAs use simple mutation, which occasionally alters a random bit position of a randomly selected string. The probability of mutation, usually, is very low and therefore, may not be enough to maintain the scadasticity of the population. To overcome this Whitley, Starkweather, and Bogart [4] introduced an adaptive mutation process which increases the mutation rate with generation.

For biological systems mutation can be induced (known as induced mutation [6]) through irradiation. For example, treatment of sperm with relatively high doses of x-rays induces occurrences of true gene mutation in a high proportion. In other words, induced mutation increases the rate of mutation. Following the concept of induced mutation, we have proposed a method for generating a new string based on the information gathered in earlier steps. We call this *directed mutation* because the generation of the new string is directed by the potential strings of some previous populations. The new string is generated deterministically, without changing the probabilistic nature of GAs. The proposed directed mutation enhances the convergence rate without affecting the merits of conventional genetic algorithms. Several strategies for controlling the directed mutation process have been proposed and their effectiveness is demonstrated on a set of optimization problems.

2. GENETIC ALGORITHMS AND FUNCTION OPTIMIZATION

Let us consider the problem of optimizing (maximizing) a complex function $f(\mathbf{x})$ having n parameters $(\mathbf{x}=)x_1, x_2, \ldots x_n$. To solve such a problem genetic algorithms start with a set of strings/chromosomes $P=\{S_i: i=1,2,\ldots,M\}$ as the initial approximations of the parameter set. In GA

literature P is called a population, and $f(\mathbf{x})$ is called the fitness function. Each string S_i represents a coded version of an approximate solution set $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$. Usually binary strings of length $L = n \cdot l$ is taken as a string or chromosomal representation of an approximation. Here substring comprising bits $(i-1).l+1$ through $i.l$; $i = 1, 2, \ldots, n$, represents an approximation of the ith parameter. For example, let 1010101010, 1100110011,..., and 1110001110 be the coded representation of a_{i1}, a_{i2},..., and a_{in}, respectively then the realization $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$ is represented as a binary string (chromosome) of length $10n$ of the form

$$\underset{a_{i1}}{1010101010} \quad \underset{a_{i2}}{1100110011} \quad \cdots \quad \underset{a_{in}}{1110001110}$$

The population of strings then undergoes a sequence of three genetic operations to produce usually an improved [with respect to optimization of $f(\mathbf{x})$] population. These operations are: (i) selection/reproduction, (ii) crossover, and (iii) mutation.

Selection is a process in which potential strings (strings with higher fitness value) of the population P are copied to a mating pool, $T = \{S: S \in P\}$, depending on their fitness function values. More specifically selection is a process which gets P as input and produces an output T. A string $S_i \in P$ having fitness value f_i is replicated $p_i (= (f_i / \Sigma_i f_i)|T|)$ times in T. $|T|$ is the size of the mating pool T. Generally, the size of the pool is the same as that of the population.

Crossover operation produces offspring by exchanging information between two potential strings selected randomly from the mating pool generated by selection process. For example, let

$$a = 0000000000000000000000000,$$

$$b = 1111111111111111111111111$$

be two strings of length 25 selected randomly from the mating pool T, and a random position 15 (say) be selected for crossing over. The operation will produce the following two offspring (strings)

$$a' = 0000000000000001111111111,$$

$$b' = 1111111111111110000000000.$$

The underlying philosophy with this operation is the random recombination of structured information. A sequence of this operation over the

randomly selected pairs of strings from the mating pool will produce a new population.

When multiple parameters are represented by a string the crossover operation may be performed once for each parameter. Let S_i and S_j be two strings of length $l \cdot n$ (substring of length l is considered for each parameter) be selected for crossing over. Then n random numbers (integer) in the range of 1 to $l-1$, one for each substring of length l, are chosen. Now two new strings are generated by crossover operation corresponding to a pair of substrings chosen from two different strings (as described above) [7]. This method is a version of multiple point crossover.

Mutation means an occasional random alteration of a bit position. A random bit of a randomly selected string in the population is selected and the bit value is replaced by some other value, i.e., for a binary string a 0 is replaced by a 1 and vice versa. The probability of performing mutation is of the order of 0.001 to 0.005. A schematic diagram of the basic structure of a genetic algorithm is shown in Figure 1.

In the literature no stopping criteria exist for genetic algorithms. It is very difficult to decide when to terminate the algorithm if we do not have prior knowledge regarding the optimum value of the function. Typically, the process is iterated for a large number of times which is chosen heuristically or by intuition. On the other hand, if we have some idea about the global optimal value of the fitness function (for example, when

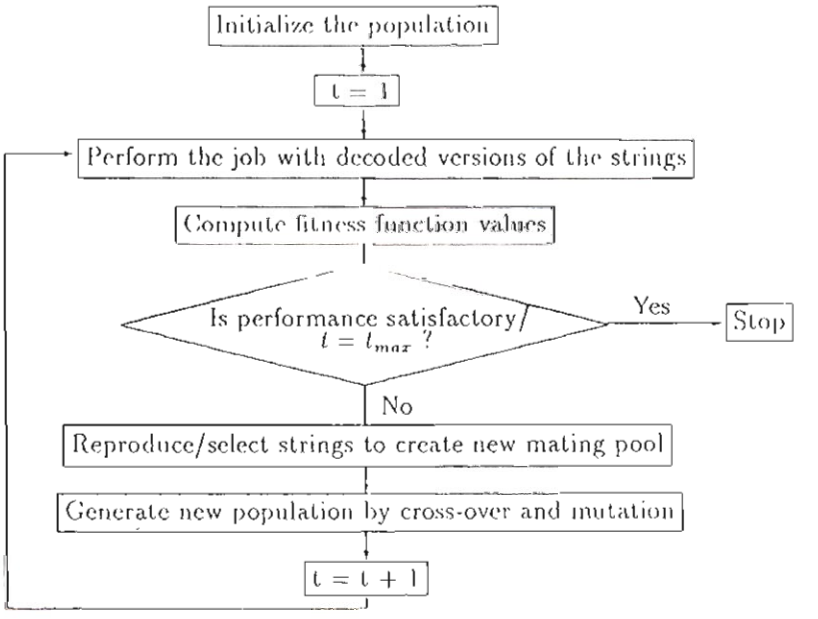


Fig. 1. Basic steps of genetic algorithm.

the square error is used as fitness function) we can pose a termination condition on fitness value [7].

Though mutation is performed with a very low probability, it has an important role in the convergence of genetic algorithms. Mutation is a necessary genetic operation because it enables (i) to regain the information lost during early generations; (ii) to obtain a bit value (allele) which does not exist at the locus of that bit in any of the strings in the population; and (iii) to sustain the genetic diversity in order to reduce the chance of getting stuck to local optima.

The effect of mutation is not always worth. The randomness in performing mutation may cause a replacement of an important bit value, and consequently the fast convergence to a good solution may be affected. Sometimes it diverges the homogeneity of the population during final generations which may delay the process of convergence. On the other hand, slow rate of mutation is unable to sustain the genetic diversity in the population during earlier generations.

Adaptive mutation: To maintain the diversity in the population Whitley, Starkweather, and Bogart [4] introduced the concept of adaptive mutation. Their method differs from the conventional technique only in the varying rate of mutation. In the early generations the mutation rate is kept low and it increases with time. The homogeneity of the population increases with generations which may hinder the convergence of the algorithm to a global optimum solution. The varying rate of mutation, as suggested in [4], helps to maintain the genetic diversity. Note that this may make the algorithm slow due to higher rate of mutation.

In the following section we describe a new technique for mutation called, *directed mutation* which enjoys the existing merits of the conventional mutation operation and at the same time drives the algorithm fast to reach to an optimum solution.

3. DIRECTED MUTATION FOR GAs

Directed mutation follows from the concept of induced mutation in biological systems [6]. This operation does not involve probabilistic decision rules but the information acquired in the previous generations. Again, it does not alter the probabilistic nature of the search technique. In certain environment this operation will deterministically introduce a new point in the population. The new point is directed (guided) by the solutions obtained earlier and therefore we call it *directed mutation*.

Let S_t^* be the best fit string of the tth generation and f_t^* be the corresponding fitness value. Let $\mathbf{x}_t^* = (x_{t,1}, x_{t,2}, \ldots, x_{t,n})$ be the decoded

value of the parameters. Now if in the τth ($\tau > t$) generation the best fit string is S_τ^* with fitness value f_τ^* ($f_\tau^* > f_t^*$) then a new point $\mathbf{x}^*$ is generated as

$$\mathbf{x}^* = \mathbf{x}_\tau^* + \alpha \mathbf{g}(\mathbf{x}_t^*, \mathbf{x}_\tau^*). \tag{1}$$

Here $\mathbf{g}(\cdot)$ can be termed as an acceleration function, which is explained in the next section; $0 < \alpha < 1$ is a constant multiplier and τ represents the first generation after tth generation with $f_\tau^* > f_t^*$. The point $\mathbf{x}^*$ is then encoded as a string S^* and is introduced in the population. The other steps of GA remain unaltered.

The function $\mathbf{g}(\cdot)$ controls the directed mutation. There could be several choices for $\mathbf{g}$; for example, $\mathbf{g}$ can be a function of local gradient or it may be an extrapolation function. Choosing the multiplier α is an important task like any gradient based technique. Equation (1) reveals that higher the value of α, the greater is the difference between x^* and x_τ^*. Sometimes, this may speed up convergence to an optimum solution, but it may also lead to a bad solution or to oscillation. On the other hand, a choice of small α is expected to drive the algorithm consistently towards a good solution, but may not speed up the process greatly. A better strategy would be to use higher α in the earlier generations and relatively lower α values towards the end of the process.

To achieve this, one can use $\alpha(1 - t/t_{\max})$, $t_{\max}$ being the maximum number of iterations, instead of α, and consequently equation (1) becomes

$$\mathbf{x}^* = \mathbf{x}_\tau^* + \alpha \left(1 - \frac{t}{t_{\max}}\right) g(\mathbf{x}_t^*, \mathbf{x}_\tau^*). \tag{2}$$

3.1. *THE ACCELERATION FUNCTION*

The function $g(\mathbf{x}_t^*, \mathbf{x}_\tau^*)$ can be the gradient of $f(\mathbf{x})$. If the function is differentiable everywhere, we can use

$$g(\mathbf{x}_t^*, \mathbf{x}_\tau^*) = f'(\mathbf{x}_\tau^*). \tag{3}$$

The choice of (3) is motivated by the steepest descent technique.

If the function $f(\mathbf{x}_\tau^*)$ is not differentiable everywhere, one can use the

following acceleration function

$$g(\mathbf{x}_l^*, \mathbf{x}_\tau^*) = \frac{f_\tau^* - f_l^*}{\mathbf{x}_\tau^* - \mathbf{x}_l^*}. \tag{4}$$

Equation (4) is a crude approximation of derivative—to be more precise, it is a measure of trend.

Another possible choice may be to use simple linear extrapolation, i.e.,

$$\mathbf{x}^* = \mathbf{x}_\tau^* + \alpha(\mathbf{x}_\tau^* - \mathbf{x}_l^*). \tag{5}$$

The extrapolated point $\mathbf{x}^*$ lies on the line joining $\mathbf{x}_\tau^*$ and $\mathbf{x}_l^*$.

Consider Figure 2. Here the function is not defined at $\mathbf{x}'$, but still equation (4) or (5) generates a very desirable point. Obviously, this may not be the case always. It may happen that $\mathbf{x}^*$ generated using (4) or (5) has a very low fitness value. But in the new population, $\mathbf{x}^*$ will automatically be eliminated due to its lower fitness value. Hence, this process of adding new strings is not expected to hinder the convergence rate to a good solution—mostly it will expedite the process.

Algorithm

Given a function $f(x_1, x_2, \ldots, x_n)$ having n parameters, a set of M binary strings of length $L = n \cdot l$ (substring of length l is assumed for each

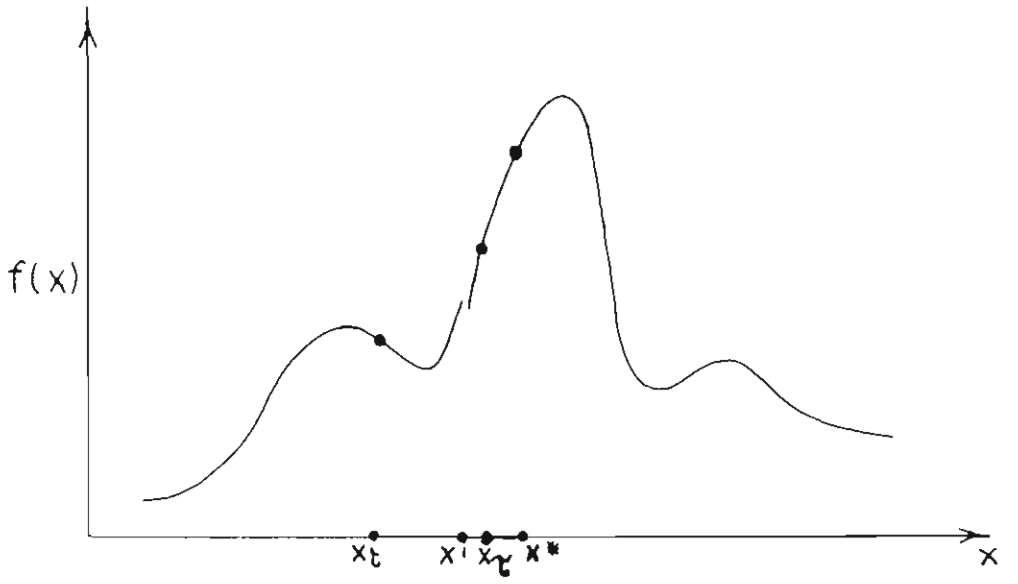


Fig. 2.

parameter) is taken as the initial population. The substrings are then decoded into real numbers in the interval $[-1, 1]$ and multiplied by some suitable constant to make them lie into their respective domains. Corresponding to each string S_i $(i = 1, 2, \ldots, M)$ the fitness function value f_i is calculated. Using the best solutions of the present and previous generations a new point is generated by directed mutation process described earlier. After generating the new population by the conventional genetic operators (viz., selection, crossover and mutation), the encoded version of the induced point is introduced in the new population and the entire process is repeated for a desired number of times.

We emphasize that the use of (4) does not convert the algorithm to gradient search. Because, the entire set of genetic operations is still in use, so the probabilistic nature of the algorithm does not change, nor does it increase the chance of getting stuck to a local optimum. The directed mutation often will make the transition to an optimal solution faster. Using equation (4) it exploits the benefits of both gradient search and GA. Similarly using equation (5) it also enjoys benefits of both extrapolation and GA.

4. RESULTS AND DISCUSSIONS

To demonstrate the effectiveness of the proposed concept of directed mutation, we have used four functions f_i, $i = 1$, 2, 3, and 4 as shown in Table 1. Except for f_1, the rest are multivariate functions. Table 1 shows the global maximum values of each function along with the domains used. The complex functional behavior of f_is is depicted through their two

TABLE 1

Functions Considered for Optimization

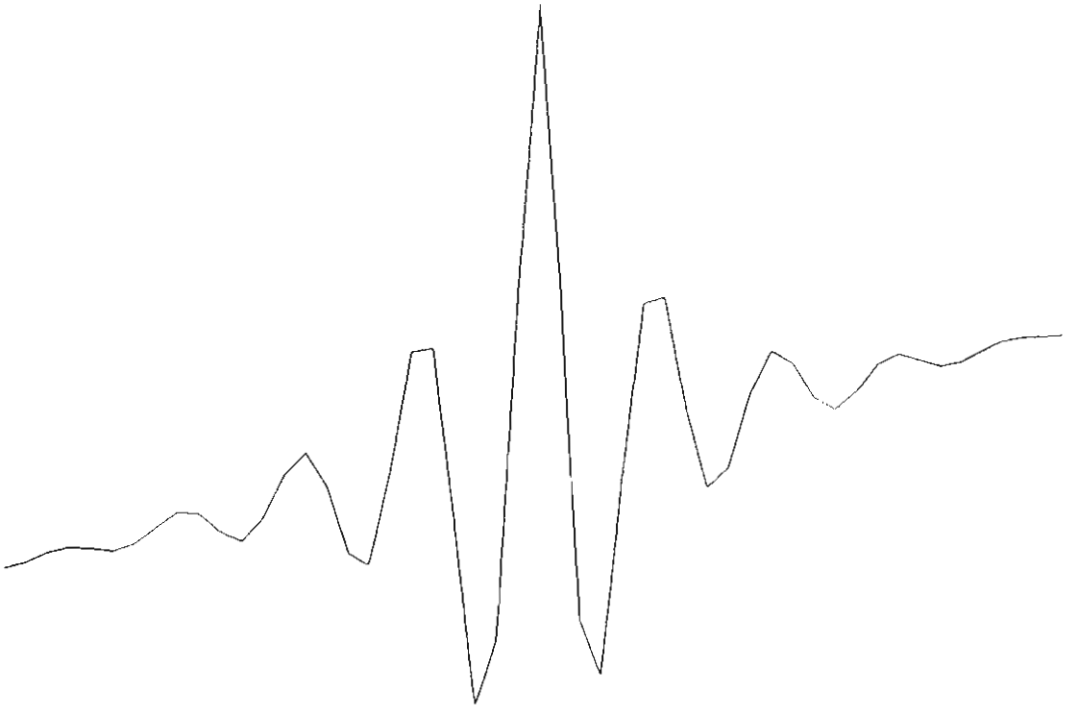
Function	Functional form	Domain	Maximum value
f_1	$2+e^{10-x}\,\mathrm{Cos}(10-x)$ if $x \leqslant 10$ $2+e^{x-10}\,\mathrm{Cos}(x-10)$ if $x > 10$	$0 \leqslant x \leqslant 20$	3.00
f_2	$\sum_{i=1}^{5} \left\{ x_i e^{1-x_i} + (1-x_i) e^{x_i} \right\}$	$0 \leqslant x_i \leqslant 1$	8.30
f_3	$\sum_{i=1}^{3} x_i^2$	$0 \leqslant x_i \leqslant 10$	300.00
f_4	$1 + \sum_{j=1}^{4} \frac{1}{1 + \sum_{i=1}^{4} (x_i - 1)^6}$	$0 \leqslant x_i \leqslant 10$	5.00

dimensional sketches (one dimensional for f_1) in Figure 3(a)–(d). f_1 and f_2 have several local maxima but only one global maximum; on the other hand, f_3 and f_4 have only one maximum each. f_4 is bell-shaped, while f_3 is monotonic.

In our experiment, we have used binary coding with l, the substring length for each parameter, $= 15$. (Note that the higher the value of l, the greater is the accuracy.) Here M, the population size, $= 10$; p_{mut}, probability of mutation, $= 0.002$; t_{max}, iteration limit, $= 100$.

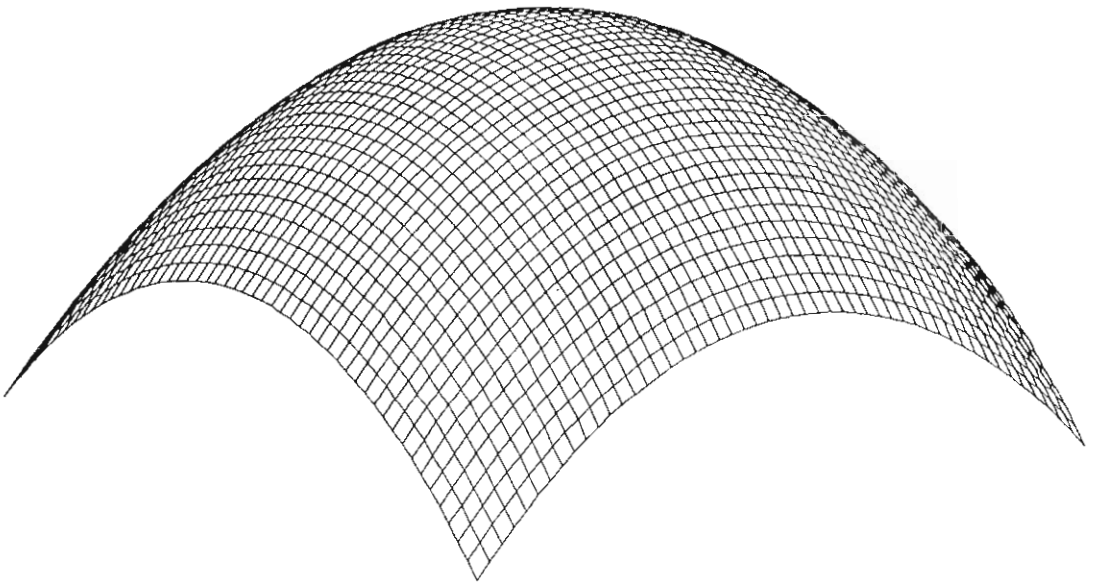
First we summarize experimental observations for linear extrapolation (equation 5). Figure 4(a) depicts the graph of fitness value vs. generation for f_1. MF in Figure 4(a) and in subsequent figures indicates the multiplication factor α. We have investigated with various choices of MF, however, in Figure 4(a) and in others we reported results for $MF = 0$, 0.2, 0.4, and 0.6. The case with $MF = 0$ represents the original GA—without directed mutation.

Figure 4(a) shows that with 100 iterations original GA cannot reach the global maxima, but directed mutation is able to drive the algorithm to attain it. One can see that for $MF = 0.2$, the optimum is reached steadily. On the other hand, for $MF = 0.6$, initially the rate of increase of the



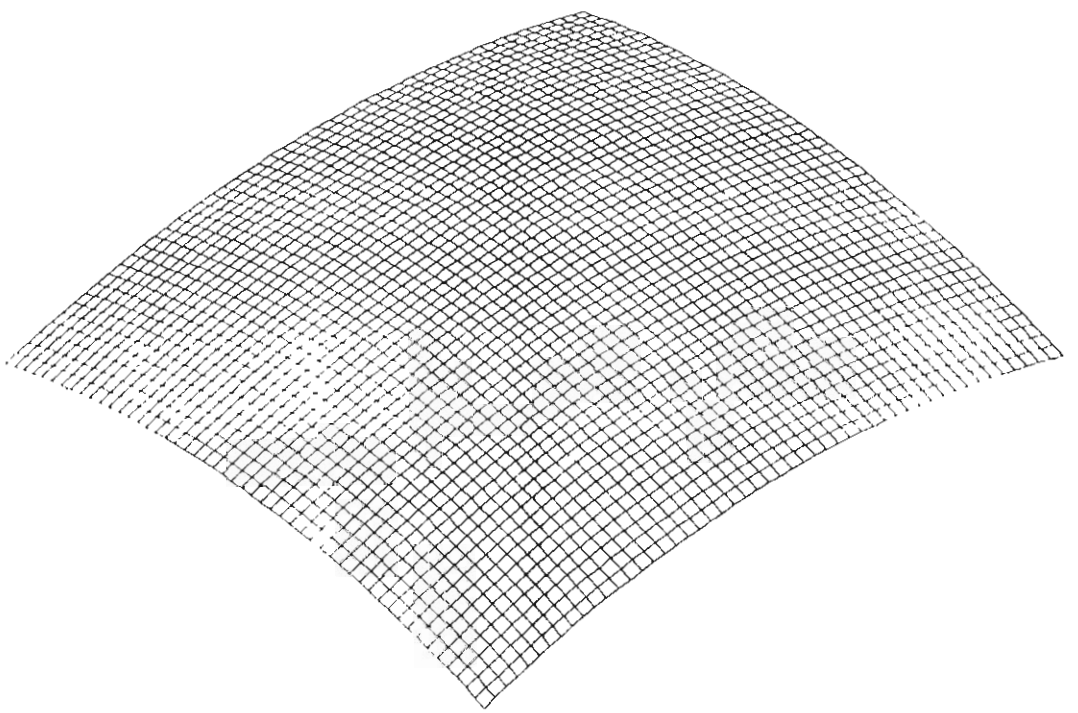
(a)

Fig. 3(a). Sketch of $2 + e^{10-x}\,\mathrm{Cos}(10-x)$ if $x \leq 10$; $2 + e^{x-10}\,\mathrm{Cos}(x-10)$ otherwise.



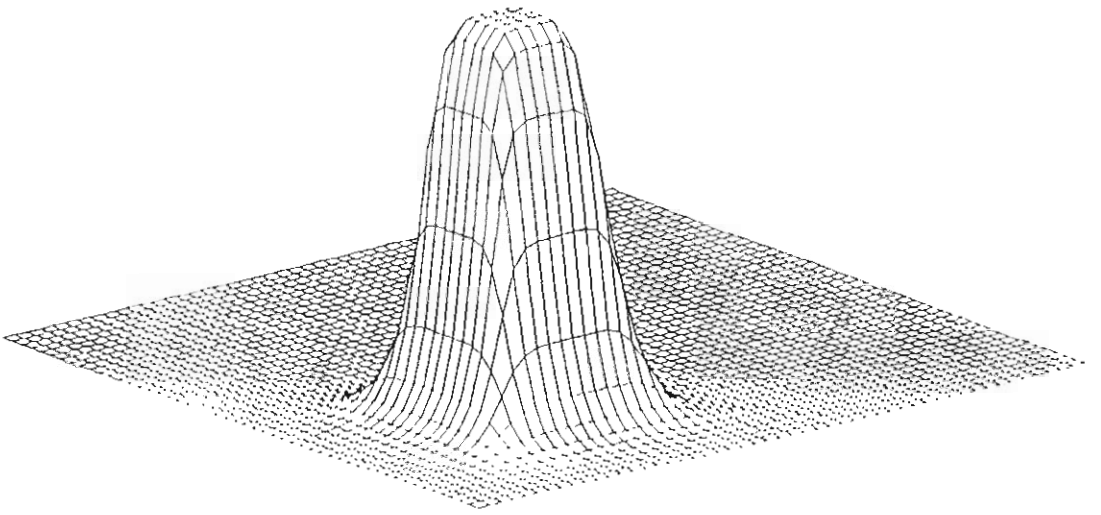
(b)

Fig. 3(b). Sketch of $\Sigma_{i=1}^{2}\{x_i e^{1-x_i} + (1-x_i)e^{x_i}\}$.



(c)

Fig. 3(c). Sketch of $\Sigma_{i=1}^{2} x_i^2$.



(d)

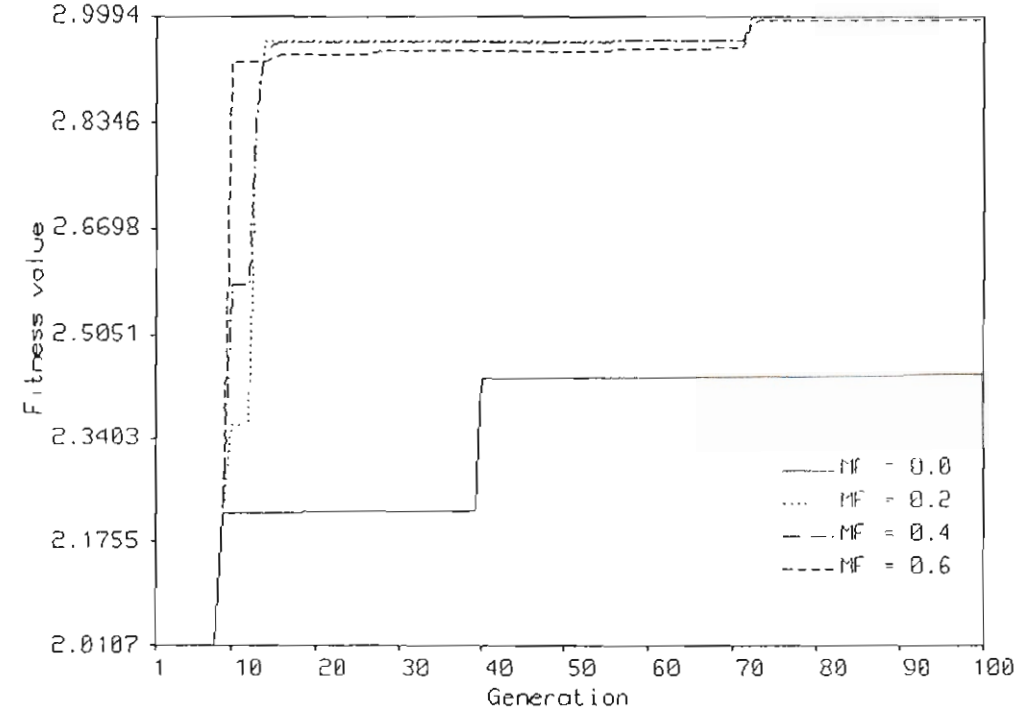
Fig. 3(d). Sketch of $1 + \sum_{j=1}^{2} (1/1 + \sum_{i=1}^{4} (x_i - 1)^6)$.

functional value is higher, but it takes more iterations to attain the maximum. The behavior of the curve for $MF = 0.4$ is somewhat in between situations with $MF = 0.2$ and 0.6. This is natural as for small MF, like any gradient type search (we remind readers, in our case it is not a gradient search), optimum is reached slowly but steadily, while for large MF initial movements towards a local optimum may be fast, but near the optimum, high MF may have an adverse effect. To get around this problem one strategy could be to start with a high MF and then reduce MF with iteration. This strategy has been explored when gradient information is used. Figure 4(b) represents the plot of fitness values with generation for f_2. Unlike f_1, here ordinary GA also attains the optimum within 100 iterations. However, the basic characteristics of Figure 4(b) are the same as that of Figure 4(a).

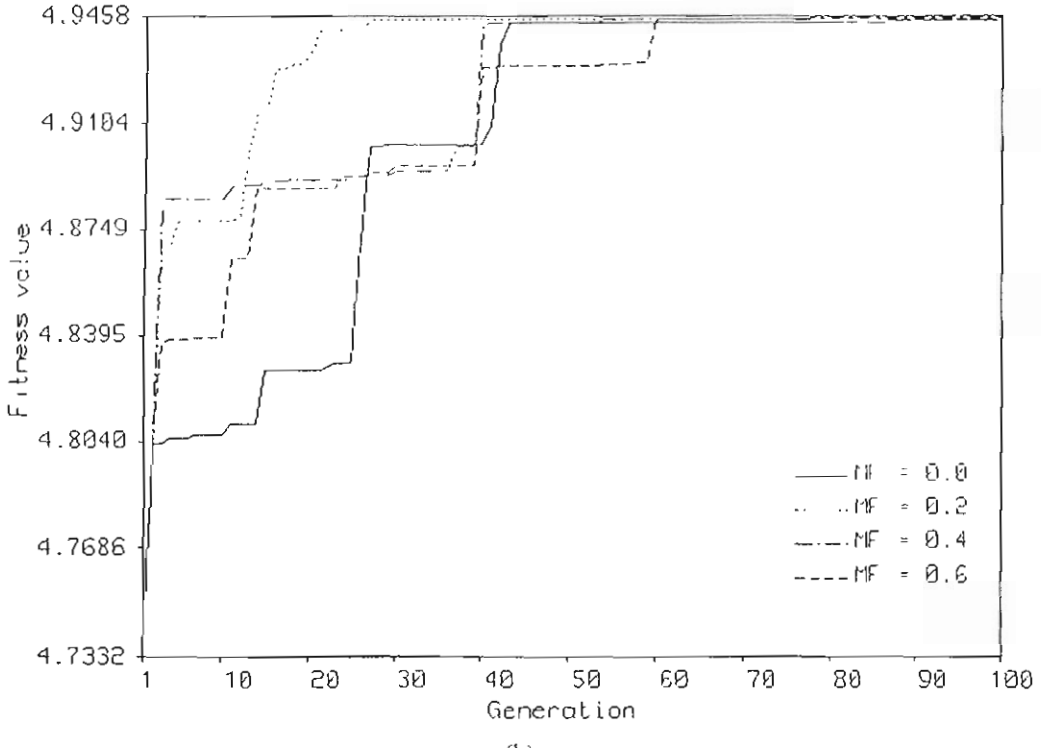
Again for Figure 4(c), for f_3, ordinary GA fails to attain the global maximum in 100 iterations. This, of course, does not mean that ordinary GA cannot attain the global maximum—it can, provided we allow sufficient number of iterations. For this function there is not much difference between the graphs for $MF = 0.4$ and $MF = 0.6$.

For f_4 [Figure 4(d)], $MF = 0.2$ and 0.4 produce better results than the ordinary GA, and directed mutation with $MF = 0.6$. We summarize that for all of the four functions directed mutation with extrapolation is very effective in detecting the global optimum quickly.

Like linear extrapolation, the gradient [Equation (4)] is also used as the acceleration function in implementing the directed mutation. Figure

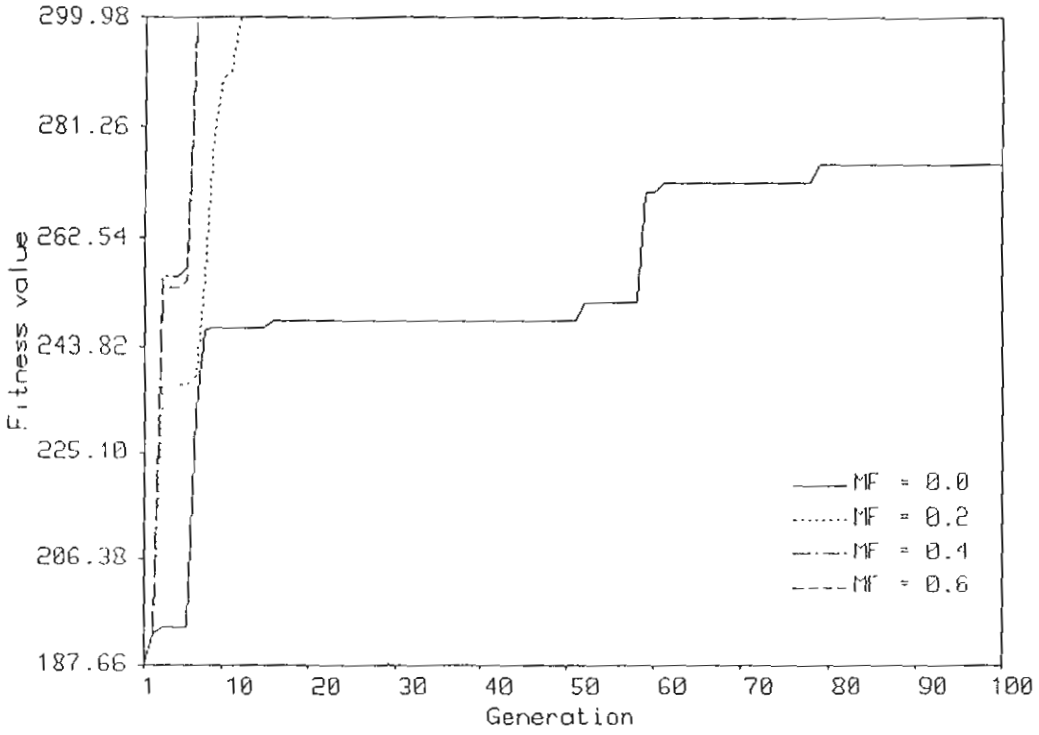


(a)

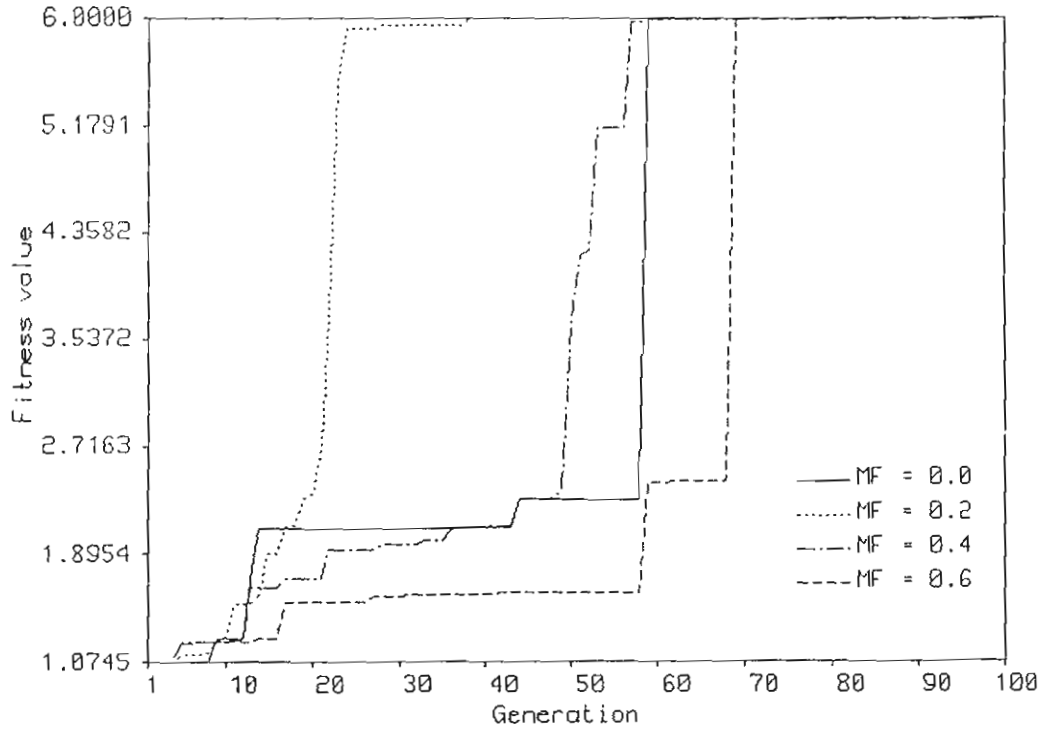


(b)

Fig. 4. Performance of GA with directed mutation using extrapolation as acceleration function and fixed MF (a) for f_1, (b) for f_2, (c) for f_3, and (d) for f_4.



(c)



(d)

Fig. 4. *(Continued).*

5(a)–(d) depict the effect of the proposed mutation on optimization of the four functions f_1 through f_4. The initial population is the same as that of the previous experiment.

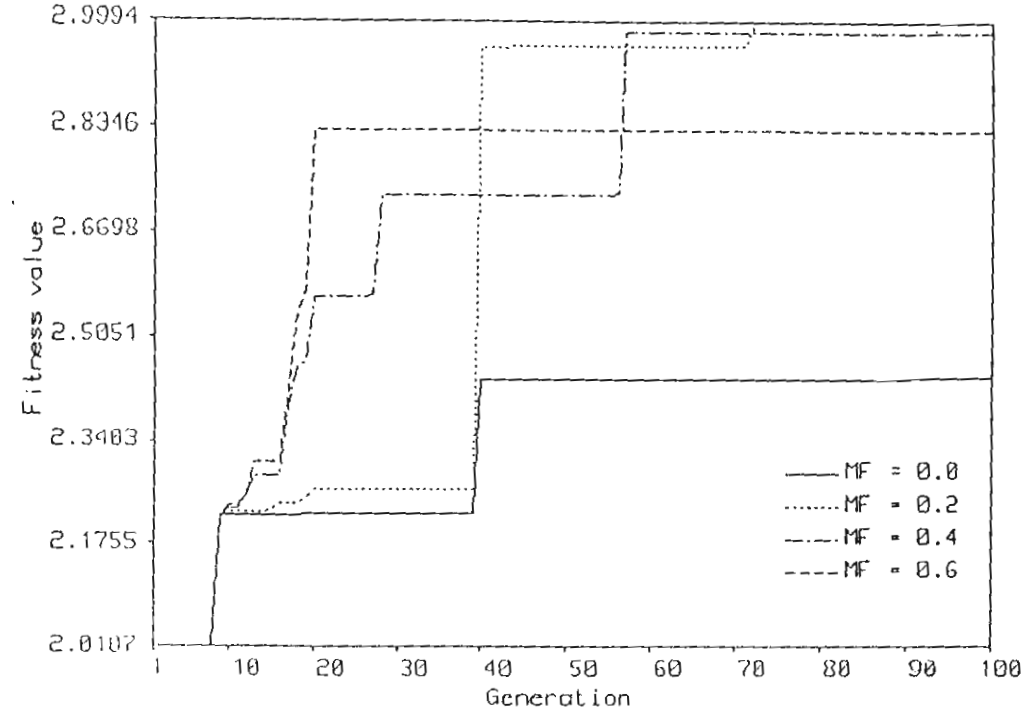
For the function f_1, the algorithm is able to reach the optimum when $MF = 0.2$ within 100 iterations [Figure 5(a)]. But it fails (within 100 iterations) when $MF = 0.4$ and 0.6. This may be due to the loss of genetic information in the earlier iterations. For f_2, like f_1, with higher MF value ($MF = 0.6$), the fitness value improves initially at a faster rate, but ultimately fails to attain a good solution in 100 iterations. The fast rate of improvement may cause loss of information (during selection) which may, in turn, affect the algorithm.

Similar result is observed in the case of f_4 [Figure 5(d)], a high value of MF helps the algorithm to speed up in the earlier generations but it takes more number of iterations to attain the optimum solution. Note that a high value of MF does not affect the convergence of the algorithm for f_3 [Figure 4(c)]. It may possibly be due to the monotonic behavior of the function. Investigation, so far, suggests that a choice of MF around 0.4 would be desirable. However, the effect of choice of MF can be minimized by using the strategy [Equation (4)] of varying α (MF). Next we demonstrate the performance of directed mutation under this strategy.

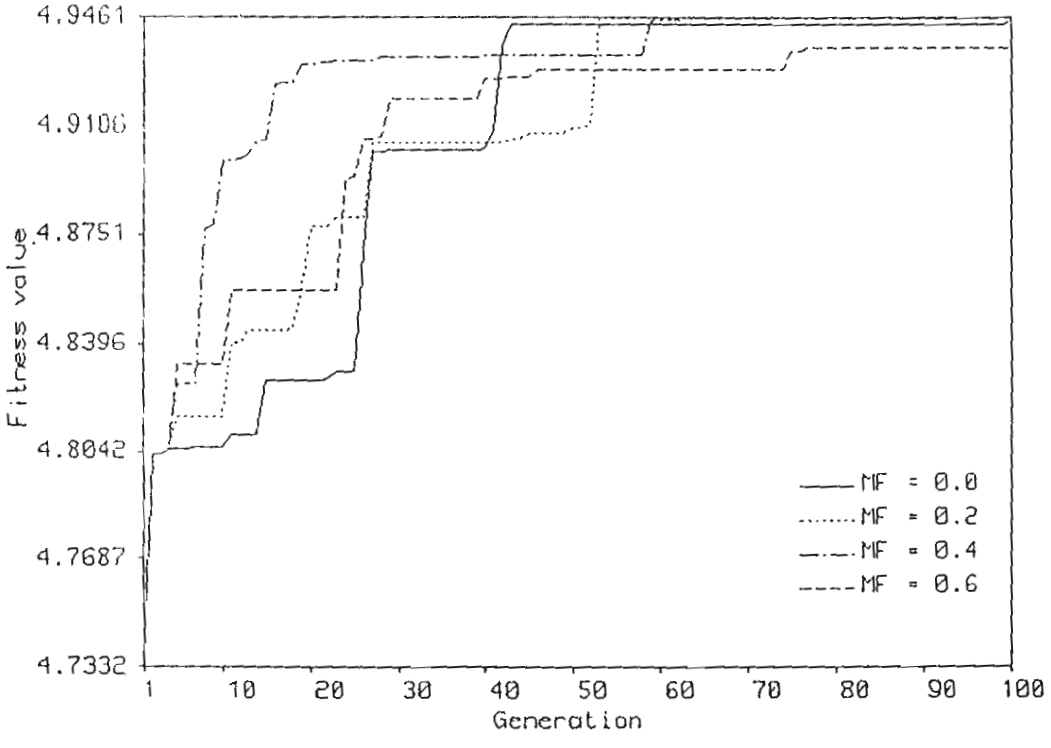
In this case, one can always start with a high $MF(\alpha)$, say $MF = 0.6$ or 0.8. We also report our findings for $MF = 0.2$, 0.4., and 0.6. Figure 6(a) depicts the result for f_1 when equation (4) is used. Comparison of Figure 6(a) with 5(a) shows that strategy of reducing MF with iteration is much better than that with fixed MF. For example, consider the curves corresponding to $MF = 0.6$ in both figures 6(a) and 5(a). We see that GA with directed mutation when used in conjunction with equation (4) converges very close to the global optimum in 75 iterations, while with fixed MF the final fitness value is much away from the desired one even after 100 iterations. Similar is the situation for f_2, f_3, and f_4 [Figure 6(a)–(d)].

Our findings mentioned so far should not give an impression that the original GA fails to attain the global optimum, but the GA with directed mutation does not. We simply demonstrated that GA with directed mutation can reach an optimum solution much quicker (with fewer iteration) than the original GA. If sufficient number of iterations are allowed, conventional GA can also attain the global optimum. Figure 7 demonstrates this when 200 iterations are used with four different initial populations ($P_1 - P_4$) for f_1. Note that in our experiment we considered the same initial population for different runs for a particular function.

In summary, directed mutation is very effective in finding an optimum solution fast. Of the two strategies, fixed MF and reducing MF, the later is better as it is less sensitive to the choice of $MF(\alpha)$.

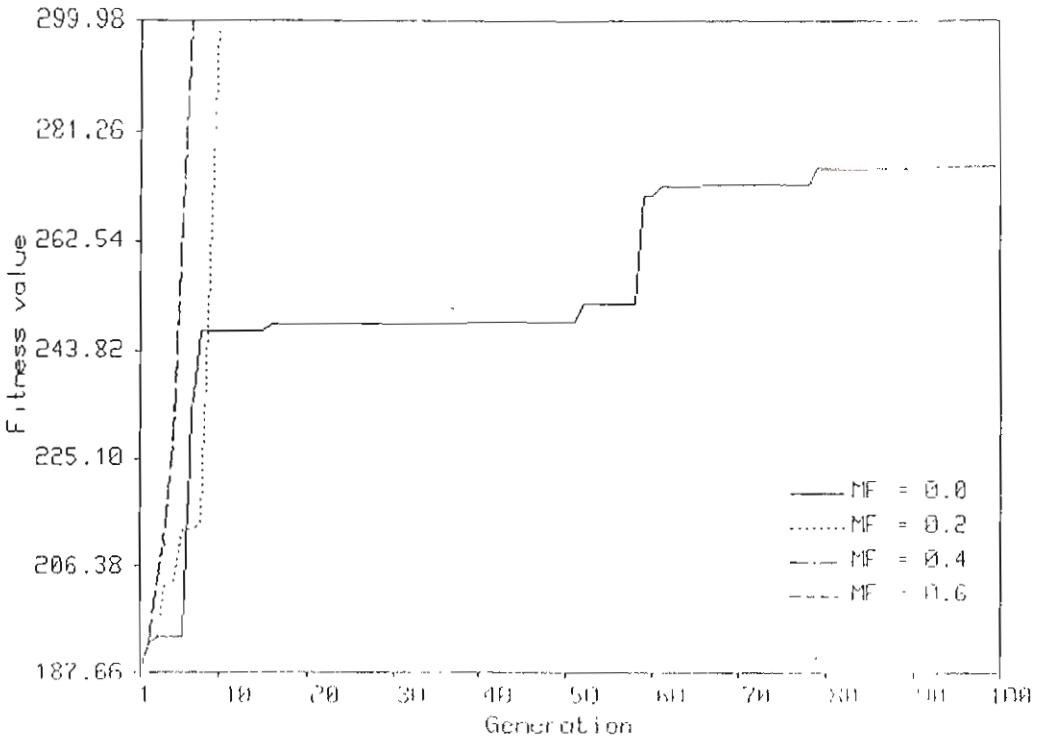


(a)

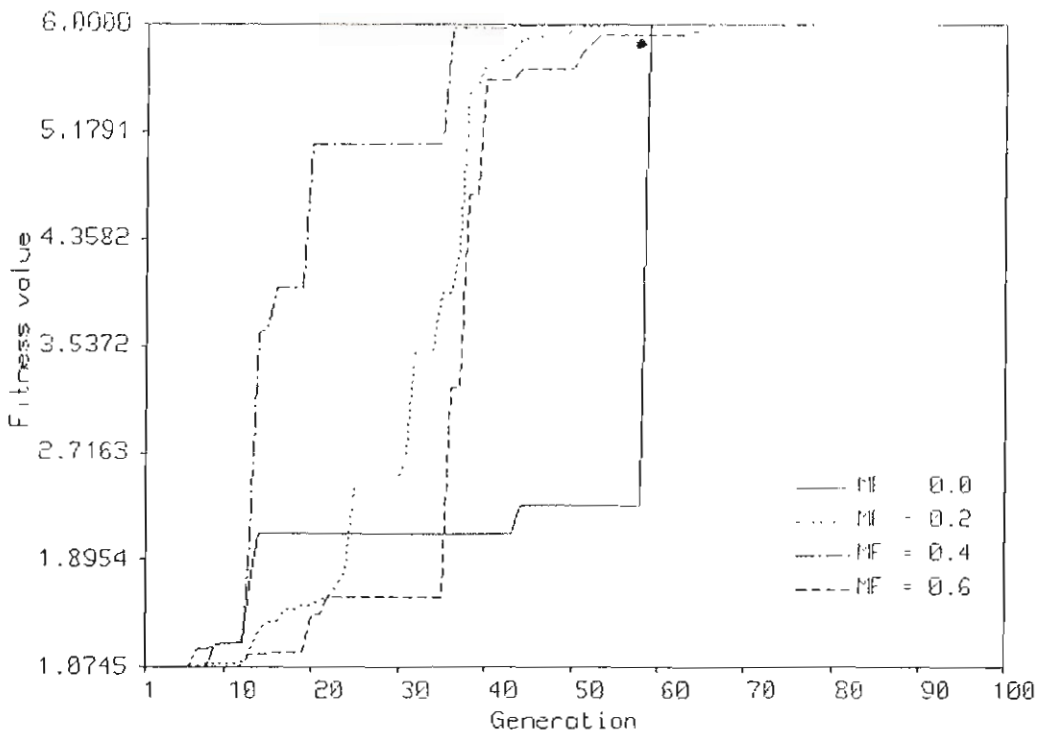


(b)

Fig. 5. Performance of GA with directed mutation using gradient as acceleration function and fixed *MF* (a) for f_1, (b) for f_2, (c) for f_3, and (d) for f_4.

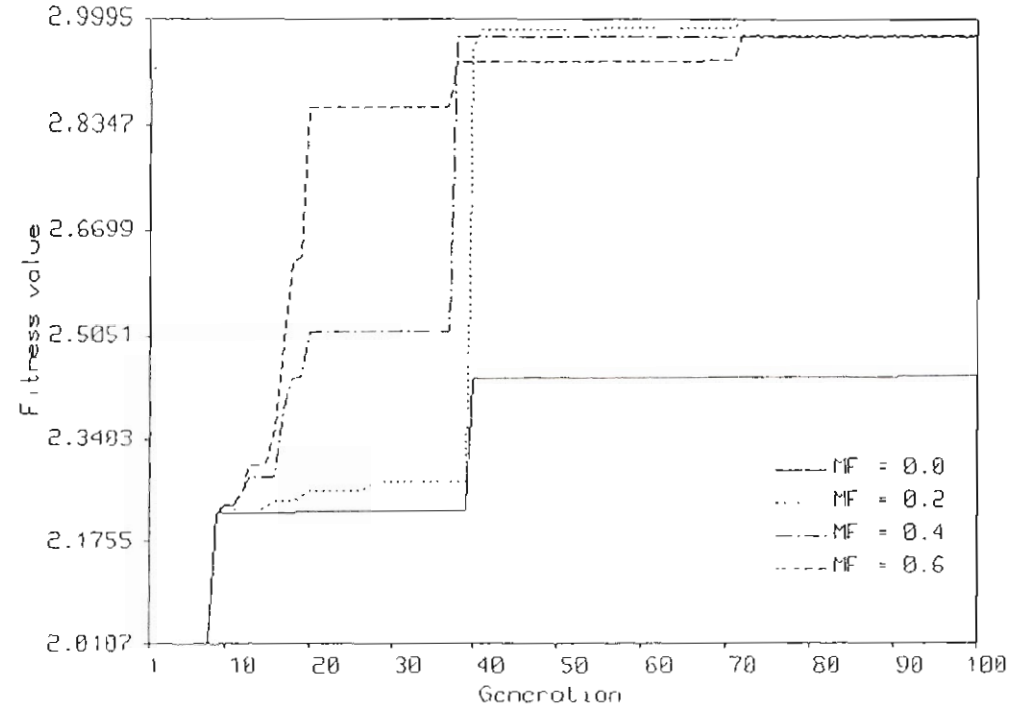


(c)

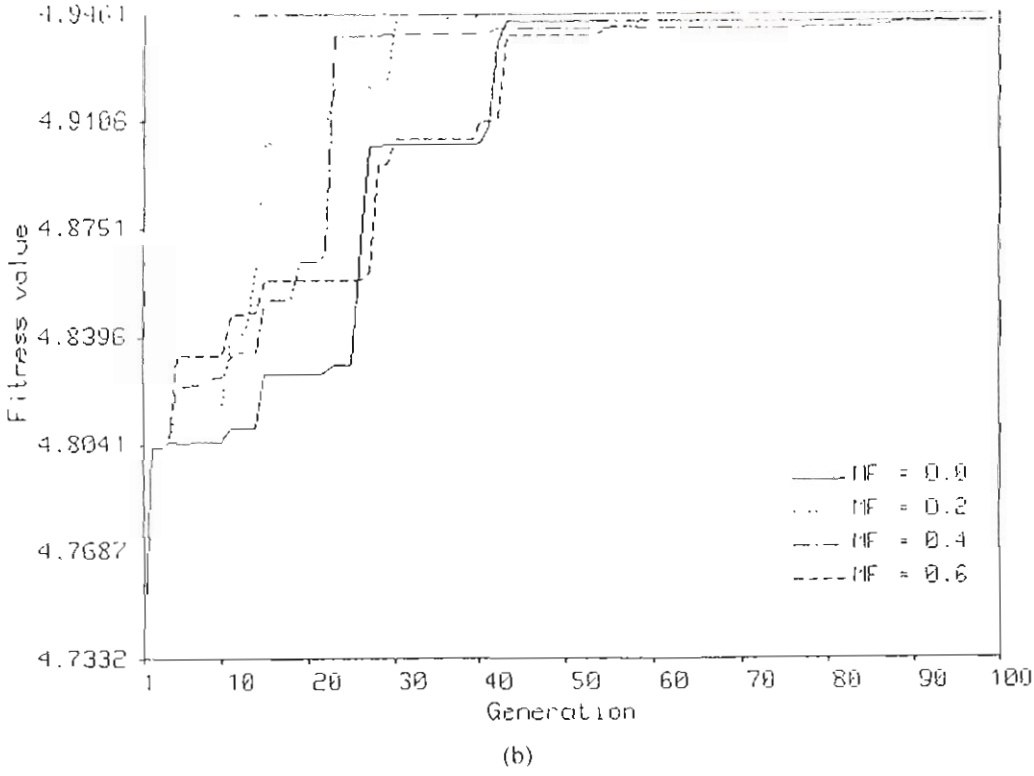


(d)

Fig. 5. *(Continued).*

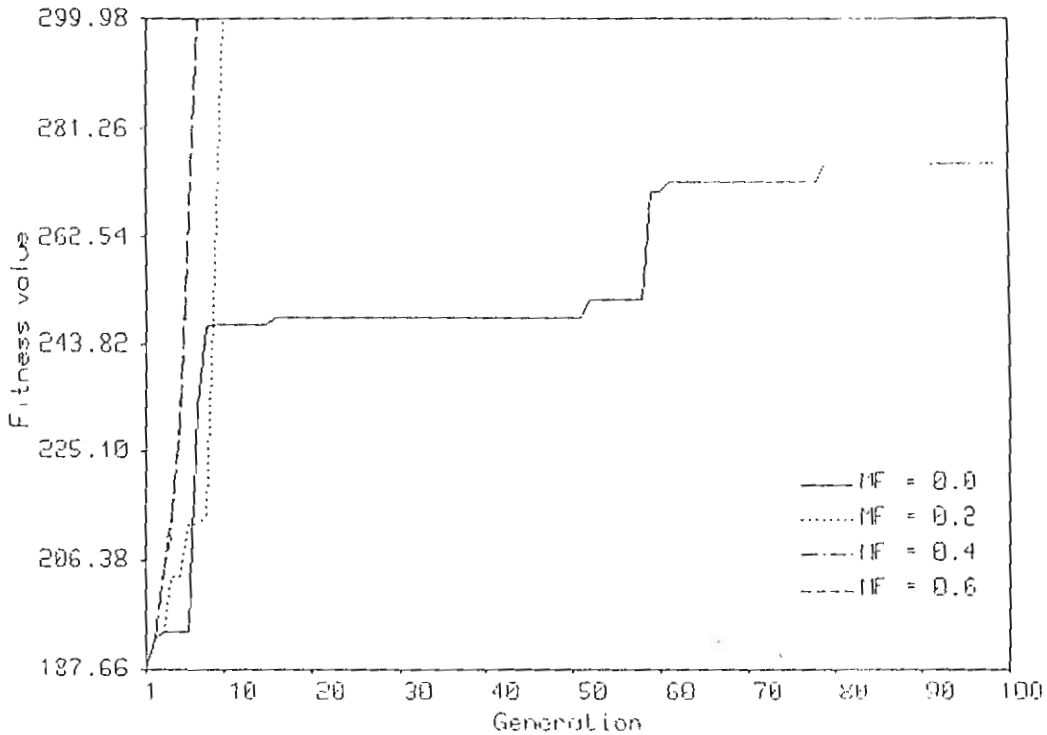


(a)

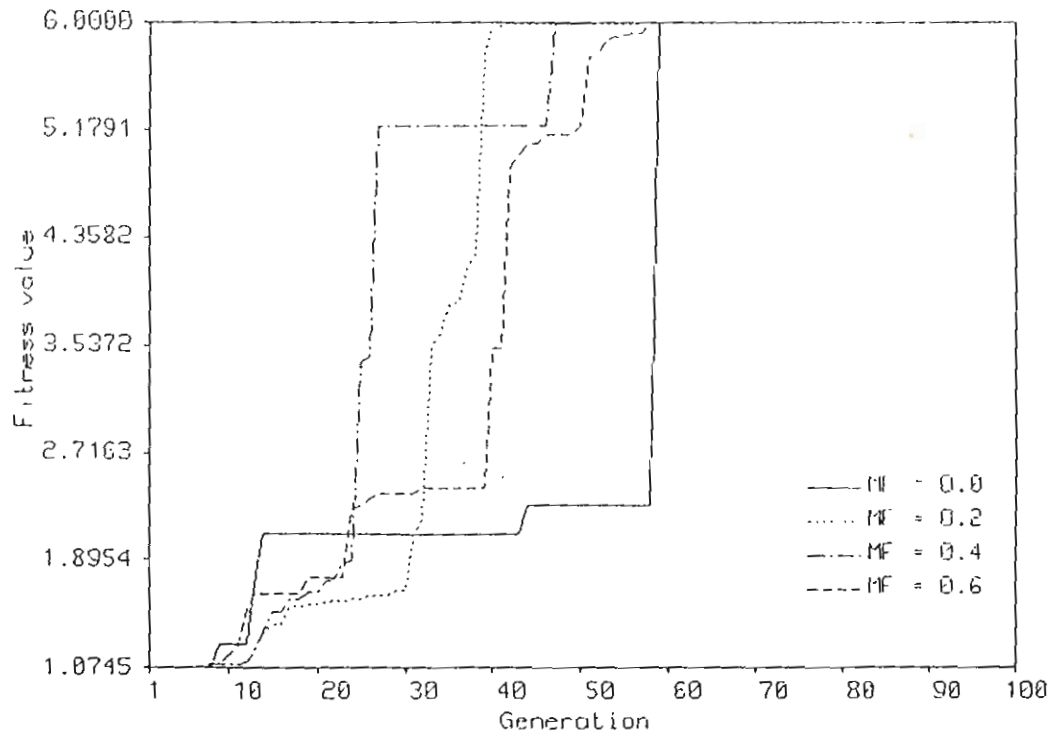


(b)

Fig. 6. Performance of GA with directed mutation using gradient as acceleration function and varying MF (a) for f_1, (b) for f_2, (c) for f_3, and (d) for f_4.



(c)



(d)

Fig. 6. *(Continued).*

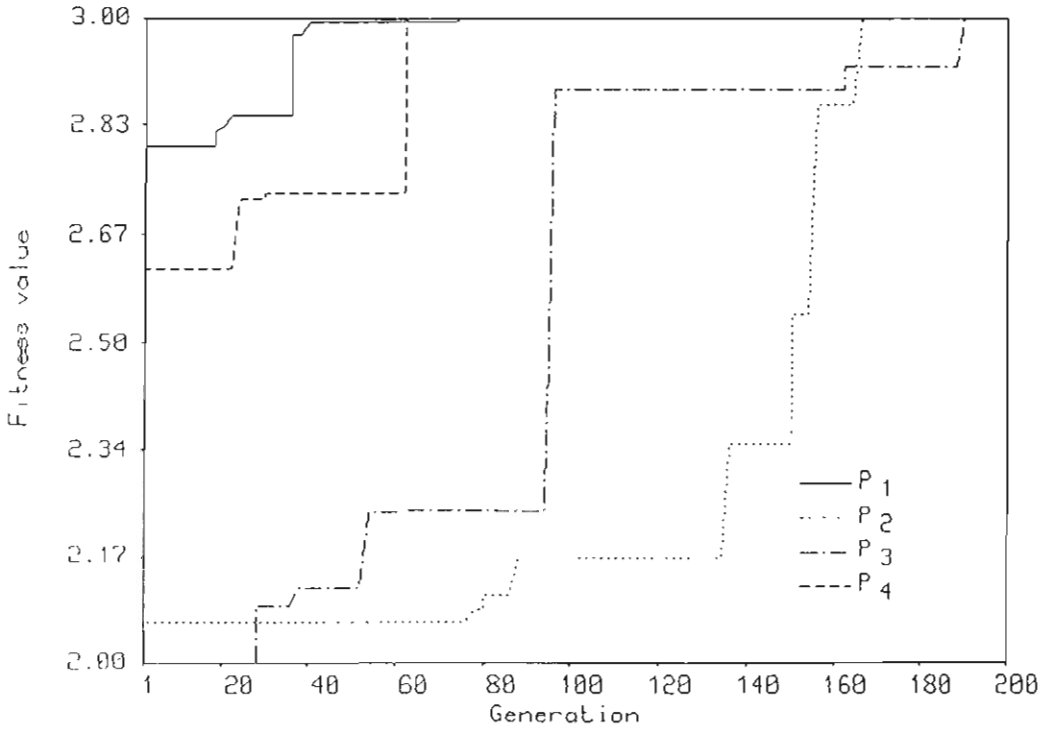


Fig. 7. Performance of ordinary GA with different initial populations on f_1.

5. CONCLUSIONS

Conventional GAs have been modified incorporating a new concept named directed mutation. It helps to maintain the genetic diversity of the population, and at the same time accelerates the convergence of the algorithm to a good solution. Directed mutation exploits the benefits of both gradient search/extrapolation technique and genetic algorithms. The effectiveness of GAs with directed mutation operation is demonstrated for solving complex optimization problems. Empirically it has been found that GAs with directed mutation need lesser number of iterations than the ordinary GAs to obtain a good solution. This operation can be incorporated to any GA based optimization such as automatic selection of optimal image enhancement operator, determining optimal set of weights of a multilayer perceptron [7], etc.

REFERENCES

1. L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*. Pitman Publishing, London, 1987.
2. D. E. Goldberg, *Genetic Algorithms: Search, Optimization and Machine Learning*. Addison Wesley, New York, 1989.

3. Z. Michalewicz, *Genetic Algorithms + Data Structure = Evaluation Programs*. Springer-Verlag, Berlin, 1992.
4. D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: optimizing connections and connectivity," *Parallel Computing*, 14:347–361 (1990).
5. *Proceedings of the Fourth International Conference on Genetic Algorithms*. University of California, San Diego, 1991.
6. H. J. Muller, *Studies in Genetics—Selected Papers*. Indiana University Press, 1962.
7. S. K. Pal and D. Bhandari, Selection of optimum set of weights in a layered network using genetic algorithms, *Inform. Sci.*, 1994 (to appear).

Received 15 September 1993; accepted 3 October 1993