Genetic algorithms with variable mutation rates: application to fuzzy logic controller design

D T Pham and **D Karaboga**
Intelligent Systems Research Laboratory, University of Wales, Cardiff

Abstract: Three variable mutation rate strategies for improving the performance of genetic algorithms (GAs) are described. The problem of optimizing fuzzy logic controllers is used to evaluate a GA adopting these strategies against a GA employing a static mutation regime. Simulation results for a second-order time-delayed system controlled by fuzzy logic controllers (FLCs) obtained using the different GAs are presented.

Keywords: genetic algorithms, mutation rate, control parameters, fuzzy logic control

1 INTRODUCTION

Genetic algorithms (GAs) are optimization algorithms based on principles of natural evolution (**1**–**3**). There are five factors influencing the performance of a GA: the method of representing solutions (how solutions are encoded as chromosomes), initial population of solutions (the group of chromosomes created at the beginning of the evolution process), evaluation function (the metric for measuring the fitness of a chromosome), genetic operators (e.g. the three basic operators, selection, crossover and mutation, for choosing chromosomes for a new generation, creating new chromosomes by combining existing chromosomes and producing a new chromosome by altering an existing chromosome) and control parameters (e.g. the size of the population of chromosomes and rates of crossover and mutation).

The first three factors and, to some extent, the fourth factor also, are problem dependent. For instance, in a problem where there is prior knowledge of reasonably fit solutions, they can be used to form the initial population to speed up the optimization process. In other problems, such as the travelling salesman problem, where chromosomes cannot contain duplicated genes (e.g. the sequence of cities to be visited by the salesman cannot include a given city more than once), some genetic operators, like the crossover operator, are not usable.

The fifth factor, GA control parameters, tends to be much less problem dependent and there is more scope for generic work aimed at improving the performance of a GA by manipulating its control parameters. This paper focuses on one control parameter in particular, the mutation rate. Mutation is an operator used to create diversity in the population of solutions. It is a search operator for exploring the solution space during optimization. A correct mutation rate is required, as too high a rate converts the GA into a random search procedure and too low a rate can cause the process to be trapped at local optima.

The paper is organized as follows. Section 2 summarizes previous work on control parameters. Section 3 describes three strategies for varying the mutation rate. Section 4 presents the results of evaluating the proposed strategies against the constant mutation rate strategy for the problem of optimizing the relation matrix of a fuzzy logic controller (FLC) for a second-order time-delayed plant. This problem had previously been investigated by the authors with a standard GA (**4**) and with a GA involving cross-breeding different populations (**5**). In both cases, the constant mutation rate strategy was adopted. For information on fuzzy logic controllers, relation matrices and GAs, the reader is referred to references (**4**) to (**6**), for example.

2 PREVIOUS WORK ON CONTROL PARAMETERS

De Jong (**7**) has carried out experiments on different test problems to study the effect of control parameters on the performance of a GA. He has suggested a set of parameter values (see Table 1) which have been found generally to

The MS was received on 27 July 1995 and was accepted for publication on 16 December 1996.

Table 1 Control parameter values suggested for GAs

Control parameters	De Jong	Schaffer	Grefenstette
Population size	50–100	20–30	30
Crossover rate	0.60	0.75–0.95	0.95
Mutation rate	0.001	0.005–0.01	0.01

produce good on-line performance (the average performance of all solutions obtained in all generations) and off-line performance (the performance computed by using only the current best solution value for each generation).

Similarly, Schaffer *et al.* (**8**) have conducted a wide-ranging experiment on the effect of control parameters on the performance of a GA and have concluded that good on-line performance can be obtained using the following set of parameters given in Table 1.

Grefenstette (**9**) has investigated the use of a GA to optimize the control parameters for another GA. This method of determining control parameter values is considered more robust than the experimental approach of De Jong and Schaffer *et al.* as it better explores the space of parameter values. The set of parameter values found by Grefenstette is presented in Table 1.

A theoretical investigation of the optimal population size for genetic algorithms employing binary-coded chromosomes has been carried out by Goldberg (**10**) who has found the following relation between population size and length of a chromosome

$$\text{Population size} = 1.65 \times 2^{0.2\ \text{length}}$$

The above studies have focused on determining good control parameter values for genetic operators. Once they have been found, they remain fixed for all applications. However, the optimal parameter setting is likely to vary for different problems. Therefore, to achieve a good performance, new optimal sets of values should be obtained for a given problem, which is a time consuming task. Employing parameters with automatically variable values can save time and yield improved performance. There have been a number of studies on adapting the control parameters during optimization. Davis (**11**) has described a technique for setting the probabilities of applying genetic operators, and therefore their rates, while the GA is executing. The technique involves adjusting the probabilities for the different operators based on their observed performance as the optimization proceeds. Davis's technique is complex as it does not involve just the mutation operator but other operators also.

Whitely and Hanson (**12**) have proposed another form of adaptive mutation. This involves indirectly monitoring the homogeneity of the solution population by measuring the Hamming distance between parent chromosomes during reproduction: the more similar the parents, the higher the probability of applying the operator to their offspring so as to increase diversity in the population. However, as the population naturally contains more multiple copies of good chromosomes towards the end of evolution, this strategy can be disruptive at that stage and therefore delays convergence.

Fogarty (**13**) has studied two variable mutation regimes. The first is to use different mutation probabilities for different parts of a chromosome (higher probabilities for less significant parts). However, the probabilities are fixed for all generations and all chromosomes. In the second regime, the same mutation probability is adopted for all parts of a chromosome and then decreased to a constant level after a given number of generations. Both of Fogarty's strategies are non-adaptive in the sense that mutation rates are all predetermined. Thus, these strategies are implicitly based on the assumption that the evolution process always follows a fixed pattern, which is clearly not true.

3 STRATEGIES FOR VARYING MUTATION RATES

Three new heuristic strategies are described in this section. The first strategy involves changing the mutation rate when a 'stalemate' situation is reached. The second strategy adapts the mutation rate according to the fitness of a chromosome. The third strategy embodies the second strategy but also prescribes different mutation probabilities for different parts of a chromosome.

3.1 First strategy

With this strategy, the minimum ($\text{mutrate}_{\text{min}}$) and maximum ($\text{mutrate}_{\text{max}}$) mutation rates are first decided by the user who also selects a threshold ($\text{nrep}_{\text{max1}}$) for the number of generations (nrep) during which the performance of the best solution can remain the same before the mutation rate has to increase. At the same time, the user chooses a threshold ($\text{nrep}_{\text{max2}}$) for the number of generations required to ramp up to the maximum mutation rate.

At the beginning of the optimization, the mutation rate (mutrate) is set to $\text{mutrate}_{\text{min}}$. As the optimization proceeds, the performance of the best solution in each generation is recorded and if it has not improved after $\text{nrep}_{\text{max1}}$ generations (i.e. a 'stalemate' situation is deemed to have occurred), the mutation rate is increased gradually according to the following equation:

$$\text{mutrate} = \left(\frac{\text{nrep} - \text{nrep}_{\text{max1}}}{\text{nrep}_{\text{max2}}}\right)(\text{mutrate}_{\text{max}} - \text{mutrate}_{\text{min}}) + \text{mutrate}_{\text{min}} \qquad (1a)$$

where $(\text{nrep} - \text{nrep}_{\text{max1}}) \leqslant \text{nrep}_{\text{max2}}$.

The purpose of increasing the mutation rate is to enhance the probability of finding new improved solutions. As soon as an improvement in the performance of the best solution is observed, the mutation rate is directly reduced back to the $\text{mutrate}_{\text{min}}$ value, which is maintained until a new 'stalemate' situation is reached. This strategy is illustrated in Fig. 1a. The figure also shows that the mutation rate is reduced to $\text{mutrate}_{\text{min}}$ from $\text{mutrate}_{\text{max}}$ even though an improved solution has not been found. This is to avoid excessive disruption to the evolution process.

In the initial stages of an optimization process, say the first 50 generations, it is likely that there are few good solutions in the population. Therefore, it would be prefer-

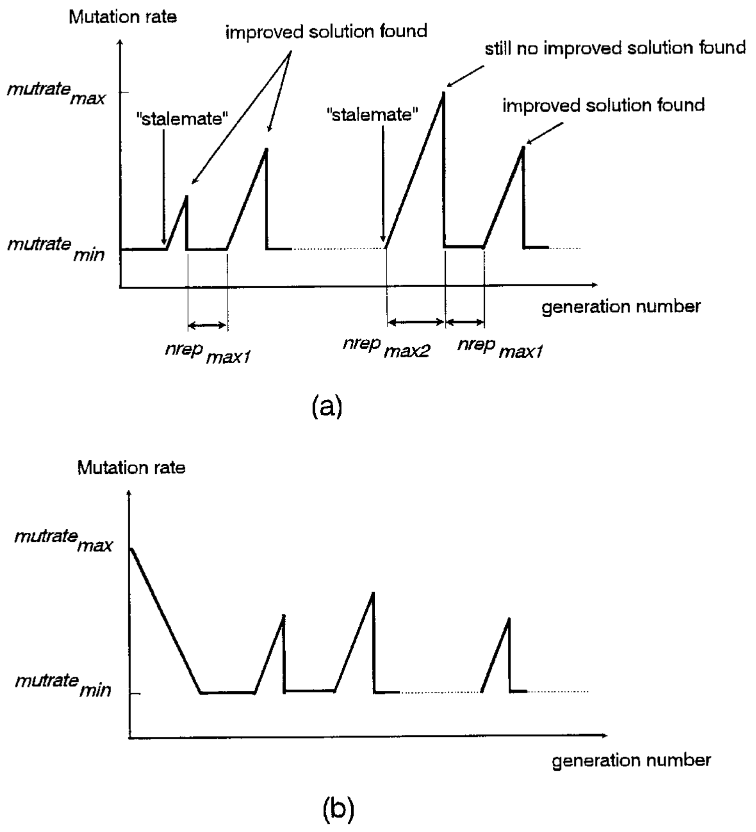


Fig. 1 Examples of mutation rate changes according to the first strategy: (a) variation 1: starting with the minimum mutation rate; (b) variation 2: starting with the maximum mutation rate

able to employ a high mutation rate to accelerate the search. Thus, a better strategy is depicted in Fig. 1b, which involves starting with the maximum mutation rate $\text{mutrate}_{\text{max}}$ and reducing it gradually to $\text{mutrate}_{\text{min}}$. The mutation rate used during a generation g in this initial phase is given by the following equation:

$$\text{mutrate}\ (g) = (\text{mutrate}_{\text{max}} - c^* g) + \text{mutrate}_{\text{min}} \qquad (1b)$$

where c is a constant and g is the current generation number, which should be less than or equal to $(\text{mutrate}_{\text{max}}/c)$. The mutation rate value computed using equation (1a) or (1b) is applied to all chromosomes in a given generation.

Both variations of the first strategy should help the GA to escape from local optima and find improved solutions in new areas of the solution space. When a promising solution is found, the mutation rate is reduced to enable the GA to search those areas in more detail.

3.2 Second strategy

This strategy is based on the idea that poor solutions require more correction than better solutions. Thus, the fitness value fit(i) of each chromosome i can be used to determine the mutation rate associated with it. As with the first strategy, an upper limit and a lower limit are chosen for the mutation rate and within those limits the mutation probability mutrate(i) for chromosome i is calculated according to its fitness value: the higher the fitness value, the lower the mutation rate. A suitable equation for calculating mutrate(i) is:

$$\text{mutrate}(i) = [1 - \text{fit}(i)]^* \text{mutrate}_{\text{max}} \qquad (2)$$

In equation (2), the fitness value fit(i) is assumed to have been normalized in the range 0.0–1.0. This strategy is shown in Fig. 2. When using this strategy, the mutation operation is carried out before other operations that can change the chromosomes in order to reduce the number of times fitness values have to be calculated.

3.3 Third strategy

This strategy is applicable in optimization problems where the genes of a chromosome represent individual numerical parameters. As with the second strategy, this strategy also involves mutating chromosomes according to their fitness. However, instead of assigning all parts of a gene the same mutation probability, the left-most digits which are the most significant are initially given a greater chance of being mutated. This is because, at the beginning of the optimization, promising areas of the solution space have not yet been located and the solutions are still poor and

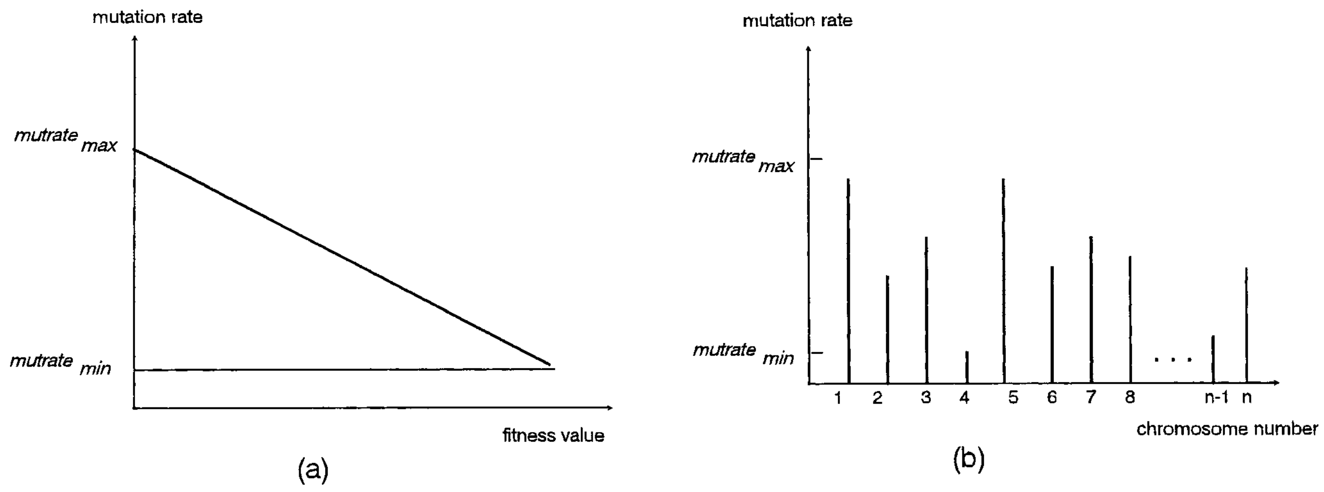


Fig. 2 Illustration of the second strategy: (a) mutation rate as a decreasing function of fitness value; (b) examples of mutation rates for a generation of chromosomes

need more correction. When a good solution is found, the mutation rates for the left-most digits are reduced compared to those for the right-most digits as the solution does not require so much correction. This leads to the fine tuning of solutions.

Thus, with this strategy as illustrated in Fig. 3, the mutation probability for each bit in a chromosome depends on three parameters: the fitness of the chromosome, the generation number and the position of the bit along the particular gene of the chromosome. Note that the second and third parameters are the same as those employed in Fogarty's second and first strategies respectively. However, the use of the fitness of a chromosome to determine its mutation rate makes the proposed strategy adaptive, unlike Fogarty's strategies, as mentioned previously.

Assuming a binary representation is adopted for the

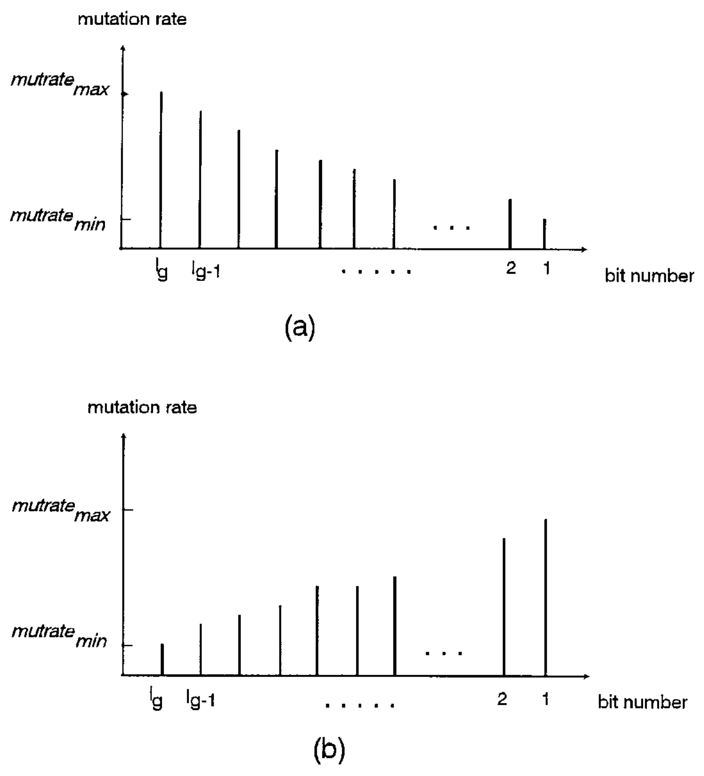


Fig. 3 Third strategy: mutation rates for a chromosome (a) before and (b) after the halfway point (generation maxgen/2) of the evolution process

chromosomes, using this strategy, the mutation rate for bit k ($k = 1$ to l_g) of chromosome i is computed as follows:

$$\text{mutrate}(i, k) = (k/l_g)^* d \qquad \text{if } g \leqslant \text{maxgen}/2 \qquad (3a)$$

$$= (1/k)^* d \qquad \text{if } g > \text{maxgen}/2 \qquad (3b)$$

where

$d = [1 - \text{fit}(i)]^*(\text{mutrate}_{\max})$

l_g = gene length

maxgen = maximum number of optimization generations to be tried

g = current generation number

4 DESIGN OF FUZZY LOGIC CONTROLLERS

The problem of genetically evolving fuzzy logic controllers (FLCs) for a given plant was used to test the mutation strategies described in the previous sections. The design of fuzzy logic controllers using GAs has been the subject of several studies (**5**, **14–17**). As with the previous work of the authors (**5**), the approach adopted here was to optimize the relation matrix of an FLC rather than the rule base. Figure 4 shows an FLC controlling a second-order time-delayed plant in a closed-loop mode. A simple noise-free plant was chosen as the aim was only to demonstrate the ability of the proposed GA to produce fitter solutions than can be achieved with the standard GA. The GA depicted in Fig. 5 was employed to search for the optimum fuzzy relation matrix **R** of the FLCs. Fixed membership functions for the fuzzification module were adopted, as carried out in references (**4**) and (**5**). The defuzzification strategy was the centre of area strategy (**4**, **5**). It was assumed that **R** had dimensions 7×11 and each element of **R** could be digitized as an 8-bit number (that is $l_g = 8$). Thus, a possible solution $\mathbf{R}_i$ was able to be represented by a chromosome of length equal to $7 \times 11 \times 8$ bits. An example of a chromosomal representation of $\mathbf{R}_i$ is given in Fig. 6.

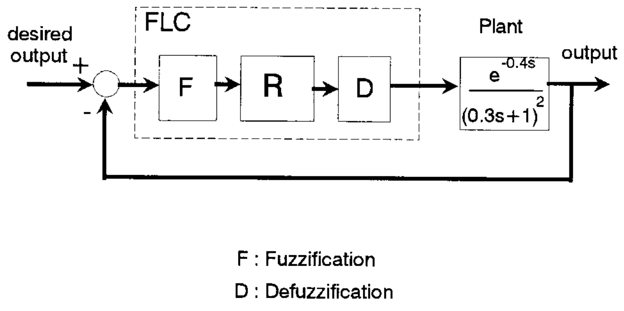


Fig. 4 A fuzzy logic control system

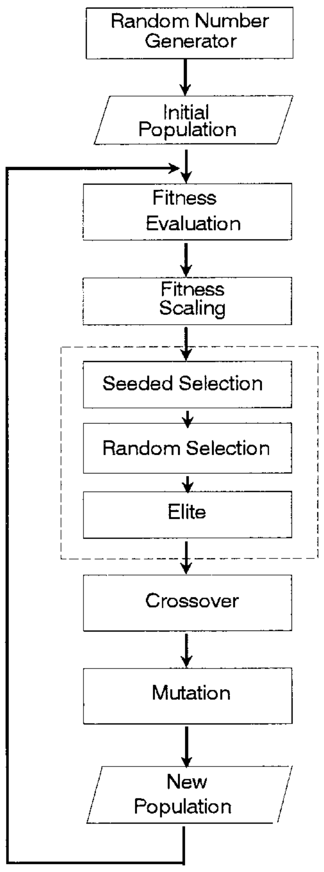


Fig. 5 Flowchart of the genetic algorithm used in this study

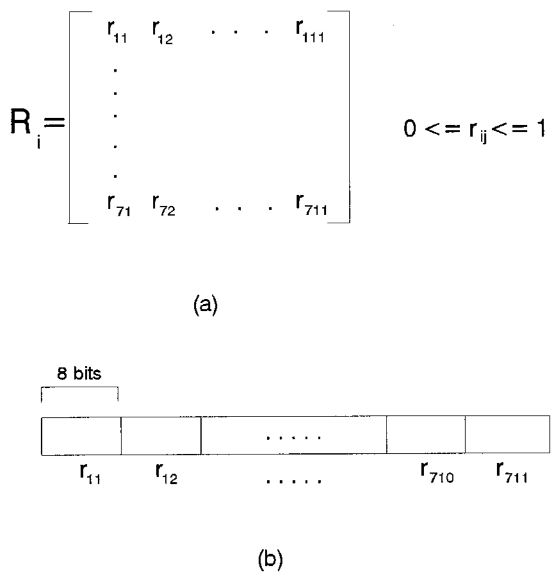

$R_i = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{111} \\ \vdots & & & \\ r_{71} & r_{72} & \cdots & r_{711} \end{bmatrix}$
$0 <= r_{ij} <= 1$
(a)
8 bits
r_{11}
r_{12}
.....
r_{710}
r_{711}
(b)

Fig. 6 (a) A fuzzy relation matrix $\mathbf{R}_i$. (b) A chromosomal representation of $\mathbf{R}_i$

The fitness of each chromosome was evaluated by reconstructing the corresponding matrix $\mathbf{R}_i$, using $\mathbf{R}_i$ in the FLC to control the given plant, obtaining the step response of the control system and computing the ITAE (integral of time multiplied by absolute error) value. The unscaled fitness value of the chromosome was taken as the inverse of the computed ITAE value. The fitness value was then normalized in the range 0.0–1.0 using the minimum fitness value obtained in the current generation (**9**). This normalization or scaling process was implemented to aid differentiation between good and better chromosomes towards the end of evolution.

Except for the mutation rate when a variable mutation rate strategy was employed, all control parameters and operating conditions for the GA were kept constant: the maximum number of generations (maxgen) was 400, the population size for a generation was 100, the crossover rate was 0.85 and a combined 'seeded' and random selection strategy was adopted, where 95 per cent of the chromosomes for a new generation were chosen on the basis of their fitness values and 5 per cent were randomly picked from the current generation. As shown in Fig. 5, the elite strategy was also used. This means that the best chromosome in one generation was always retained for the next generation.

Prior to investigating the effectiveness of the mutation rate adaptation strategies described above, simulation experiments were conducted to determine suitable ranges of mutation rates that could be employed in a GA with a fixed mutation rate regime. The rates obtained also served as guidelines for choosing the maximum and minimum limits needed for the proposed variable rate strategies. The results of the experiments are summarized in Fig. 7, which plots the fitness value of the best chromosome in each generation throughout the evolution process for different mutation rates. It can be seen that a suitable range of mutation rates for a fixed mutation rate regime is from 0.001 to 0.01. Decreasing the mutation rate to 0.0 caused the process to be trapped at local optima, yielding low fitness values. Increasing the rate to 0.05 disrupted the process and produced a poor final performance. However, that rate accelerated the search process at the beginning of evolution. Thus, in the variable mutation rate experiments, the range of mutation rates, $\text{mutrate}_{\text{min}}$ to $\text{mutrate}_{\text{max}}$, was chosen as 0.001–0.05.

Figures 8a and 8b give the results obtained with variations 1 and 2 of the first strategy. The threshold nrep_{max} used was 20. The corresponding time responses of the control system using the FLC produced at the end of

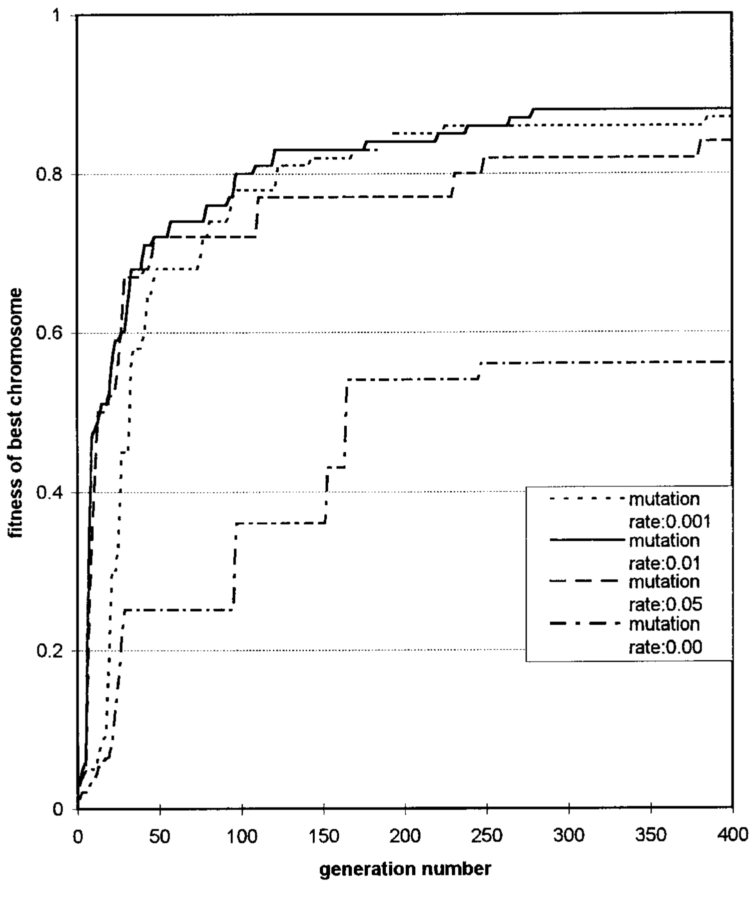
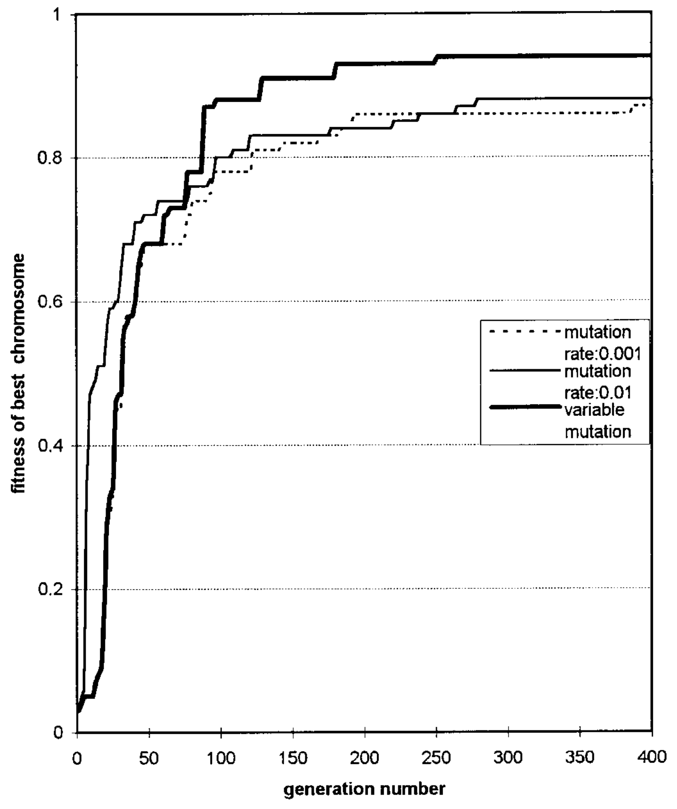
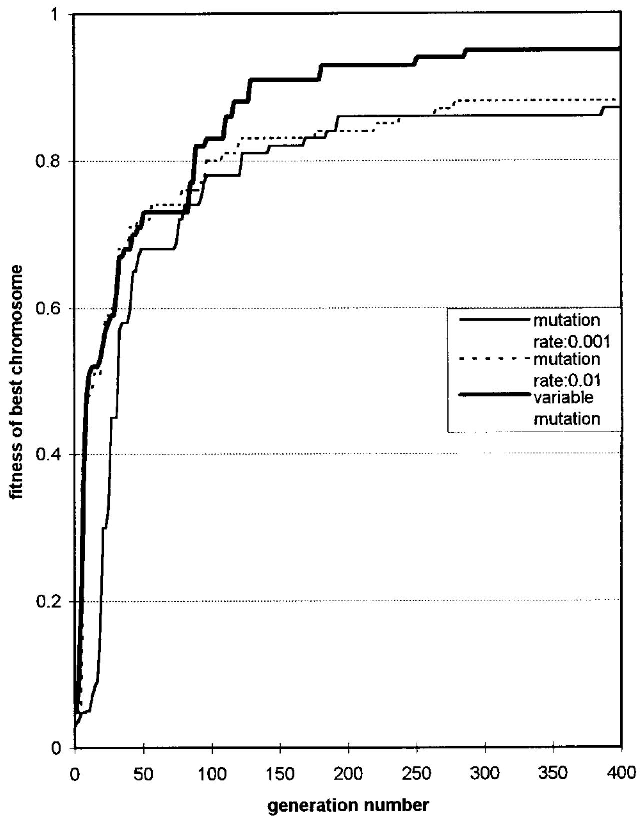


Fig. 7 Performance of a GA with different fixed mutation rates

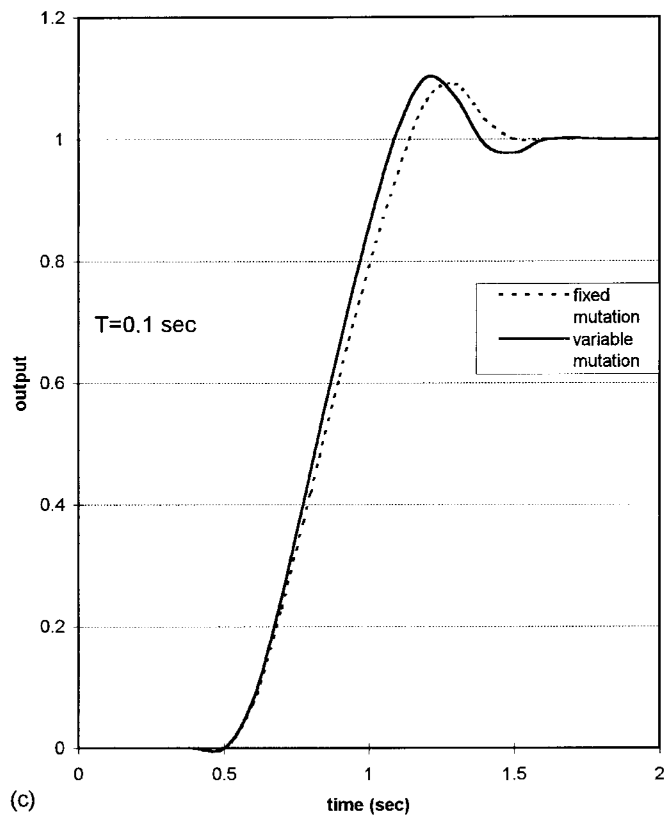


(a)



(b)

Fig. 8 (continued over)



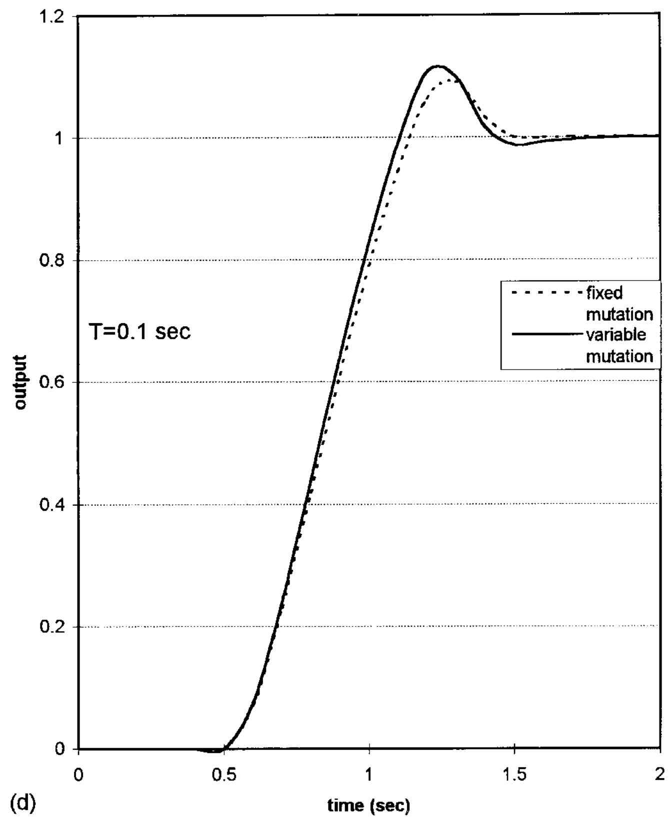


Fig. 8 Performance of a GA with variable mutation rates (first strategy) and fixed mutation rates: (a) first strategy (variation 1); (b) first strategy (variation 2); (c) responses of control system using the best FLC obtained in generation 400: first strategy (variation 1) and fixed mutation rate of 0.01; (d) responses of control system using the best FLC obtained in generation 400: first strategy (variation 2) and fixed mutation rate of 0.01 (T = sampling period)

generation 400 are plotted in Figs 8c and 8d. The responses obtained with varying mutation rates are quicker than those for a constant mutation rate. However, the overshoot is slightly higher. This is because the performance index employed in the calculation of fitness values is the ITAE parameter. The aim of the GA is simply to minimize the ITAE parameter without considering the overshoot.

Figures 9 and 10 show the results for the second strategy and the third strategy respectively. For comparison, in all cases, the results obtained using fixed mutation rates of 0.001 and 0.01 are also provided. The time responses obtained with strategies 2 and 3 are similar to those shown in Figs 8c and 8d.

Table 2 gives the results for different tests when a fixed mutation rate strategy was employed. In Tables 3, 4, 5 and 6, the results for different tests using the suggested variable mutation rate strategies are presented. It can be seen from the results obtained that the variable mutation rate strategies all produced fitter solutions than the fixed mutation rate strategy. Although the third strategy did not yield as good a final solution as the second strategy for the particular example tested, convergence was achieved after fewer generations and the performance was more consistent.

In summary, the convergence speed of the standard GA and the quality of the solution finally achieved are strongly dependent on the mutation rate selected, as seen from Fig. 7. The modified algorithm always converges to fitter solutions than those produced by the standard GA without requiring trial-and-error experiments to determine appropriate mutation rates. Although not observed in this work,

Table 2 Results obtained for different mutation rates

Test number	Mutation rate	ITAE
1	0.001	3.41
2	0.01	3.42
3	0.02	3.13
4	0.04	3.35
5	0.05	3.24

Table 3 Results obtained using the first strategy in different tests (variation 1)

Test number	ITAE
1	3.05
2	3.10
3	2.98
4	3.04
5	3.11

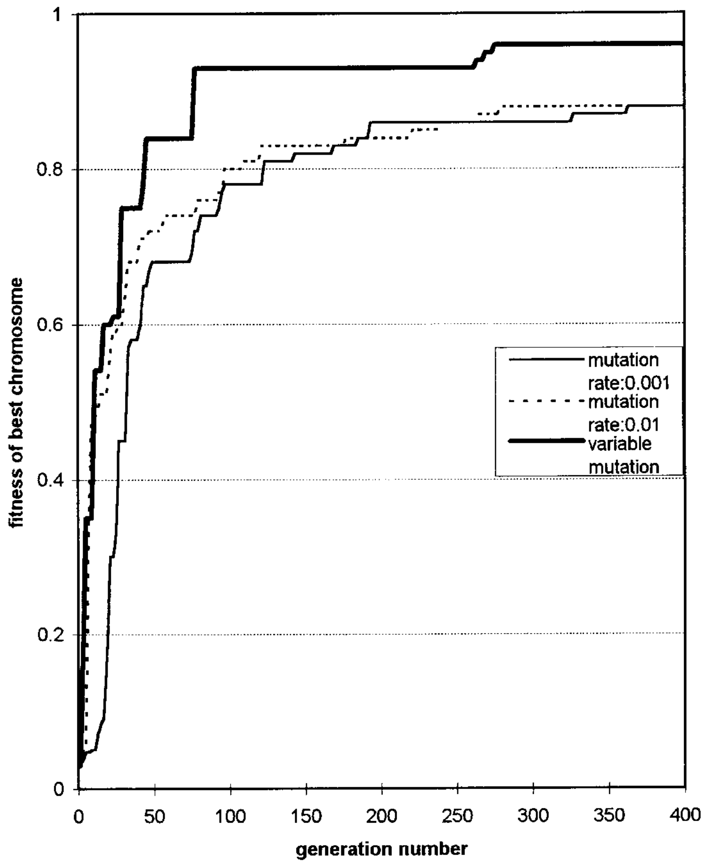


Fig. 9 Performance of a GA with variable mutation rates (second strategy) and fixed mutation rates

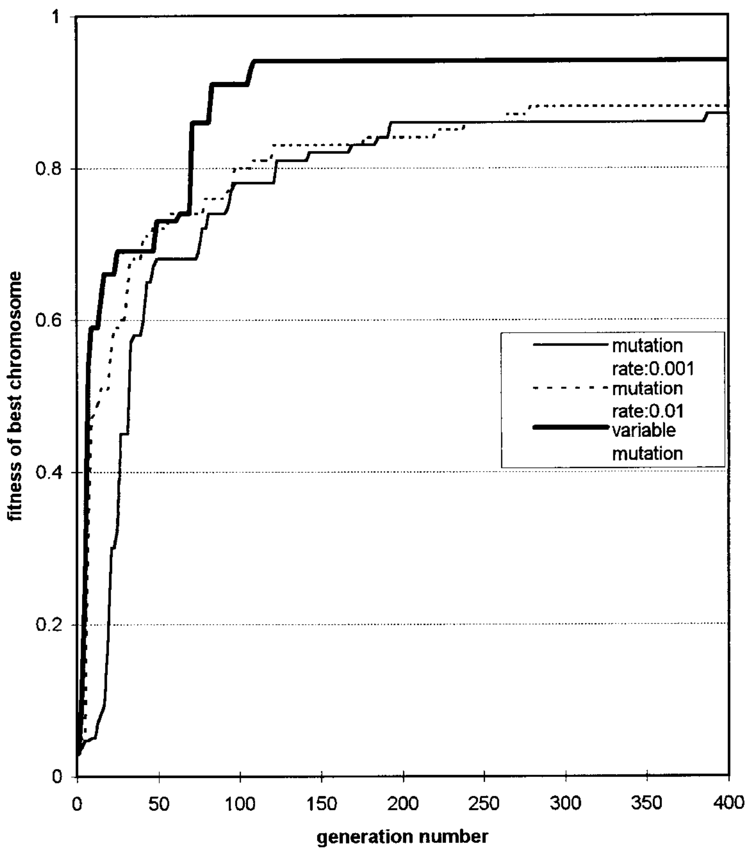


Fig. 10 Performance of a GA with variable mutation rates (third strategy) and fixed mutation rates

Table 4 Results obtained using the first strategy in different tests (variation 2)

Test number	ITAE
1	3.01
2	3.03
3	3.12
4	3.00
5	3.06

Table 5 Results obtained using the second strategy in different tests

Test number	ITAE
1	2.99
2	3.11
3	3.02
4	3.10
5	3.00

Table 6 Results obtained using the third strategy in different tests

Test number	ITAE
1	3.11
2	3.05
3	3.00
4	3.05
5	3.01

in spite of extensive testing, it is possible that after many more iterations the standard GA might produce solutions approaching the fitness levels achieved by the proposed GA. This makes the standard GA not as suitable as the proposed GA for the on-line tuning of FLCs, which is the subject of another investigation (**18**).

5 CONCLUSION

This paper has described three simple strategies for automatically adapting the mutation rate in a GA. All three strategies gave better results than the commonly adopted fixed rate strategy. The first strategy is computationally the most efficient as the same mutation rate is applied to all chromosomes. The third strategy is the most involved and should be considered for the more difficult multivariable optimization problems. It is well known that the mutation operation is most important to the performance of a GA when searching a highly complex solution space as it helps the search process to escape from local minima and find better solutions. Therefore, the authors believe that the benefits of the proposed strategies would be even more evident when designing FLCs for more complicated and non-linear plants or when solving complex optimization problems.

ACKNOWLEDGEMENTS

The authors would like to thank the Higher Education Funding Council for Wales, the CVCP and Erciyes University, Turkey, for their support of the work described in this paper.

REFERENCES

1 **Holland, J. H.** *Adaptation in Natural and Artificial Systems*, 1975 (University of Michigan Press, Ann Arbor, Michigan).

2 **Goldberg, D. E.** *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1989 (Addison-Wesley, Reading, Massachusetts).

3 **Davis, L.** *Handbook of Genetic Algorithms*, 1991 (Van Nostrand Reinhold, New York).

4 **Pham, D. T.** and **Karaboga D.** A new method to obtain the relation matrix of fuzzy logic controllers. In Proceedings of Sixth International Conference on *Artificial Intelligence in Engineering*, Oxford, July 1991, pp. 567–581.

5 **Pham, D. T.** and **Karaboga, D.** Optimum design of fuzzy logic controllers using genetic algorithms. *J. Systems Engng*, 1991, **1**(2), 114–118.

6 **Pham, D. T.** and **Yang, Y.** Optimisation of multi-modal discrete functions using genetic algorithms. *Proc. Instn Mech. Engrs, Part D*, 1993, **207**(D1), 53–59.

7 **De Jong, K. A.** Analysis of the behavior of a class of genetic adaptive systems. PhD dissertation, Department of Computer and Communication Science, University of Michigan, Ann Arbor, Michigan, 1975.

8 **Schaffer, J. D., Caruana, R. A., Eshelman, L. J.** and **Das, R.** A study of control parameters affecting on-line performance of genetic algorithms for function optimisation. In Proceedings of Third International Conference on *Genetic Algorithms and Their Applications*, George Mason University, Fairfax, Virginia, June 1989, pp. 51–61.

9 **Grefenstette, J. J.** Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, 1986, **SMC-16**(1), 122–128.

10 **Goldberg, D. E.** Optimal initial population size for binary-coded genetic algorithms. TCGA report 8510001, Department of Engineering Mechanics, University of Alabama, Alabama, 1985.

11 **Davis, L.** Adapting operator probabilities in genetic algorithms. In Proceedings of Third International Conference on *Genetic Algorithms and Their Applications*, George Mason University, Fairfax, Virginia, June 1989, pp. 61–69.

12 **Whitely, D.** and **Hanson, T.** Optimising neural networks using faster, more accurate genetic search. In Proceedings of Third International Conference on *Genetic Algorithms and Their Applications*, George Mason University, Fairfax, Virginia, June 1989, pp. 370–374.

13 **Fogarty, T. C.** Varying the probability of mutation in the genetic algorithm. In Proceedings of Third International Conference on *Genetic Algorithms and Their Applications*, George Mason University, Fairfax, Virginia, June 1989, pp. 104–109.

14 **Karr, C. L.** Design of an adaptive fuzzy logic controller using a genetic algorithm. In Proceedings of Fourth International Conference on *Genetic Algorithms* (Eds R. Belew and L. Booker), 1989, pp. 450–457 (Morgan Kauffman Publishers, Cambridge, Massachusetts).

15 **Thrift, P.** Fuzzy logic synthesis with genetic algorithms. In Proceedings of Fourth International Conference on *Genetic Algorithms* (Eds R. Belew and L. Booker), 1991, pp. 509–513 (Morgan Kauffman Publishers, Cambridge, Massachusetts).

16 **Linkens, D. A.** and **Nyongesa, H. O.** Genetic algorithms for fuzzy control. Part 1: offline system development and application. *IEE Proc., Control Theory Applic.*, 1995, **142**(3), 161–176.

17 **Linkens, D. A.** and **Nyongesa, H. O.** Genetic algorithms for fuzzy control. Part 2: online system development and application. *IEE Proc., Control Theory Applic.*, 1995, **142**(3), 177–185.

18 **Pham, D. T.** and **Karaboga, D.** Design of an adaptive fuzzy logic controller. Technical report, Intelligent Systems Laboratory, School of Engineering, University of Wales, College of Cardiff, 1995.