Adaptive Control of the Mutation Probability by Fuzzy Logic Controllers*

Francisco Herrera and Manuel Lozano

Department of Computer Science and Artificial Intelligence
University of Granada
18071 - Granada, Spain

Abstract. A problem in the use of genetic algorithms is premature convergence, a premature stagnation of the search caused by the lack of population diversity. The mutation operator is the one responsible for the generation of diversity and therefore may be considered to be an important element in solving this problem. A solution adopted involves the control, throughout the run, of the parameter that determines its operation: the mutation probability.
In this paper, we study an adaptive approach for the control of the mutation probability based on the application of fuzzy logic controllers. Experimental results show that this technique consistently outperforms other mechanisms presented in the genetic algorithm literature for controlling this genetic algorithm parameter.

1 Introduction

Population diversity is crucial to a genetic algorithm's ability to continue the fruitful exploration of the search space. If the lack of population diversity takes place too early, a premature stagnation of the search is caused. Under these circumstances, the search is likely to be trapped in a local optimum before the global optimum is found. This problem, called *premature convergence*, has long been recognised as a serious failure mode for genetic algorithms (GAs) ([11]).

The mutation operator may be considered to be an important element for solving the premature convergence problem, since it serves to create random diversity in the population. The diversity levels introduced by this operator depend directly on the value for the mutation probability (p_m). However, finding robust values for this parameter that allow the premature convergence problem to be avoided in any problem is not a trivial task, since its interaction with the performance of a GA is complex and the optimal choice is problem dependent ([2]). Furthermore, different p_m values may be necessary during the course of a run for inducing an optimal exploration/exploitation balance. For these reasons, different techniques have been suggested that dynamically adjust p_m during the course of evolving a solution ([2, 3, 4, 12, 23, 24]). They try to offer suitable diversity levels for avoiding premature convergence and improving the results.

One of these techniques lies in the application of *fuzzy logic controllers* (FLCs) ([10]) for adapting p_m depending on either the current state of the search or

* This research has been supported by DGICYT PB98-1319.

other GA related parameters. Although much work on the adaptation of p_m by FLCs has appeared, we consider that it suffers from some important deficiencies. On the one hand, most approaches adapt p_m along with other GA parameters (population size (N), crossover probability (p_c), etc.) ([19, 22, 25, 26]), and so, the specific effects on GA performance derived from the adaptation of p_m were not studied. Only in [20], an experimental study was presented, which was aimed at isolating the effects of the adaptation by FLCs of N, p_c, and p_m. It showed that the adaptation of p_m contributes most to high performance. On the other side, there is a generalised lack of empirical comparisons between mechanisms for adapting p_m based on FLCs and other types of mechanisms proposed for controlling this GA parameter ([2, 3, 4, 12, 23, 24]).

The goal of this paper is to report on an investigation about the adaptation of p_m by means of FLCs that attempts to overcome the deficiencies previously remarked. In order to do this, we propose and study an adaptive mechanism based on an FLC that controls p_m depending on: 1) a measure that describes the advance produced by a GA over the last generations, and 2) the p_m value used during these generations. According to these two factors, the FLC returns a new p_m value that will be used during the next generations. Its objective is to control the population diversity for escaping of a possible convergence premature or for allowing a favourable evolution to be even better.

The paper is set up as follows. In Section 2, we review different techniques for controlling p_m presented in the literature. In Section 3, we specify the inputs, output, data base, and rule base of the proposed FLC. In Section 4, we compare the results of the proposal on a given test suite against the ones obtained using other mechanisms for controlling p_m and the ones reached using different fixed p_m values. Finally, some concluding remarks are offered in Section 5.

2 Techniques for Controlling the Mutation Probability

According to [8], the mechanisms presented for controlling parameters associated with GAs may be assigned to the following three categories:

- *Deterministic control.* It takes place if the values of the parameters to be controlled are altered by some deterministic rule, without using any feedback from the GA. Usually, a time-varying schedule is used.
- *Adaptive control.* It takes place if there is some form of feedback from the GA that is used to determine the direction and/or magnitude of the change to the parameters to be controlled.
  The rules for updating parameters that are used by this type of control and, by the previous one, are termed *absolute adaptive heuristics* ([1]) and, ideally, capture some lawful operation of the dynamics of the GA over a broad range of problems.
- *Self-adaptive control.* The parameters to be controlled are encoded onto the chromosomes of the individual and undergo mutation and recombination.

Next, we describe mechanisms for the control of p_m that belong to each one of these categories.

2.1 Deterministic Control of p_m

A direction followed by GA research for the variation of p_m lies in the specification of an externally specified *schedule* which modifies it depending on the time, measured by the number of generations. A time-dependency of p_m was first suggested by Holland ([18]) himself, although he did not give a detailed choice of the parameter for the time-dependent reduction of p_m. Moreover, there is a theoretical argument for introducing a time-dependent schedule of p_m ([2]).

One of the most considered schedules consists in the decreasing of p_m during the GA run ([4, 12]). This schedule follows the absolute adaptive heuristic *"to protect the exploration in the initial stages and the exploitation later"*, which has been considered for designing other search techniques, such as simulated annealing.

We consider a linear function to control the decrease of p_m, following the idea presented in [4]. It constrains $p_m(t)$ so that $p_m(0) = p_m^h$ and $p_m(T) = p_m^l$ if a maximum of T generations are used:

$$p_m(t) = p_m^h - \frac{p_m^h - p_m^l}{T} \cdot t \quad 0 \leq t \leq T.$$

2.2 Adaptive Control at Individual Level of p_m

In [23], a technique for the adaptive control at individual level of p_m was proposed, in which p_m is varied depending on the fitness values of the solutions. Each chromosome C_i has its own associated p_m value, p_m^i, which is calculated as (maximization is assumed):

$$p_m^i = k_1 \cdot \frac{f_{max} - f_i}{f_{max} - \bar{f}} \text{ if } f_i \geq \bar{f}, \text{ and } p_m^i = k_3 \text{ if } f_i < \bar{f},$$

where f_i is the chromosome's fitness, f_{max} is the population maximum fitness, $\bar{f}$ is the mean fitness, and k_1 and k_3 are 1. In this way, high-fitness solutions are protected ($p_m^i = 0$), whilst solutions with subaverage fitnesses are totally disrupted ($p_m^i = 1$). Furthermore, this technique increases p_m when the population tends to get stuck at a local optimum and decreases it when the population is scattered in the solution space.

2.3 Self-Adaptive Control of p_m

In [4, 24], a self-adaptive technique for controlling p_m is presented, in which the genotype of an individual consists of a binary-coded solution and an associated individual real-coded mutation probability, which evolves with the solution.

Next, we describe a technique that follows this idea. An extra gene, p_m^i, is added to the front of each bitstring, C_i, which represents the mutation probability for all the genes in this bitstring. The values of p_m^i are allowed to vary from p_m^l to p_m^h. The following steps are considered for mutating the genes in a chromosome C_i:

1. Apply a *meta-mutation* on p_m^i obtaining ρ_m^i. This is carried out by choosing a randomly chosen number from the interval $[p_m^i - \delta, p_m^i + \delta]$, where δ is a control parameter.

2. Mutate the genes in C_i according to the mutation probability ρ_m^i.
3. Write the mutated genes (including ρ_m^i value) back to the chromosome.

Crossover is presently applied only to the binary vector and has no impact on p_m^i. Each offspring resulting from crossover receives the p_m^i value of one of its parents. The initial p_m^i values are generated at random from $[p_m^l, p_m^h]$.

3 Adaptive Control of p_m by Fuzzy Logic Controllers

The interaction of GA control parameter settings and GA performance is generally acknowledged as a complex relationship which is not completely understood. Although there are ways to understand this relationship (for instance, in terms of stochastic behaviour), this kind of understanding does not necessarily result in a normative theory. FLCs are particularly suited to environments that are either ill-defined or are very complex. The adaptive control of GA parameters is one such complex problem that may benefit from the use of FLCs.

Applications of FLCs for the adaptive control of GA parameters are found in [16, 17, 19, 20, 22, 25, 26]. Their main idea is to use an FLC whose inputs are any combination of GA performance measures or current control parameters and whose outputs are GA control parameters. Current performance measures of the GA are sent to the FLC, which computes new control parameter values that will be used by the GA.

In this section, we present an FLC for the adaptive control of p_m for a generational GA, during its run. The FLC is fired at each G generations and p_m is fixed over the generations in these time intervals. The FLC takes into account the p_m value used during the last G generations and the improvement achieved on f_b (fitness of the best element found so far). Then, it computes a new value for p_m, which will be used during the next G generations. Its goal is to observe the effects of a p_m value on GA performance during G generations, and produce a new p_m value that properly replies against a possible poor rate of convergence, or that allows performance to be improved even more (in the case of past suitable progress). The FLC uses fuzzy rules capturing adaptive strategies that attempt to accomplish this task (an example is: use a higher value for p_m when observing no progress on f_b, with the aim of introducing diversity).

The proposed FLC has two inputs:

- The current mutation probability, p_m^o, which will be kept in the interval $[0.001, 0.01]$. This interval was chosen since it contains a wide spectrum of p_m values that were considered frequently in the GA literature (e.g. $p_m = 0.001$ ([9]) and $p_m = 0.01$ ([15])).
- A convergence measure (minimization is assumed): $CM = \frac{f_b^c}{f_b^o}$, where f_b^c is the fitness of the current best element found so far and f_b^o is the fitness of the best element found before the last G generations. If an elitist strategy is used, CM will belong to $[0, 1]$. If CM is high, then convergence is high, i.e. no progress was made during the last G generations, whereas if it is low, the GA found a new best element, which consistently outperforms the previous one.

The set of linguistic labels associated with p_m^o is $\{Low, Medium, High\}$. The meanings of these labels are depicted in Figure 1.b (for each linguistic term, there is a *triangular fuzzy set* that defines its semantic, i.e, its meaning). The set of linguistic labels for CM is $\{Low, High\}$. Their meanings are shown in Figure 1.a.

The output of the FLC is the new p_m value, $p_m^n \in [0.001, 0.01]$, which will be considered during the following G generations. The set of linguistic labels associated with p_m^n is $\{Low, Medium, High\}$. Their meanings are in Figure 1.c.

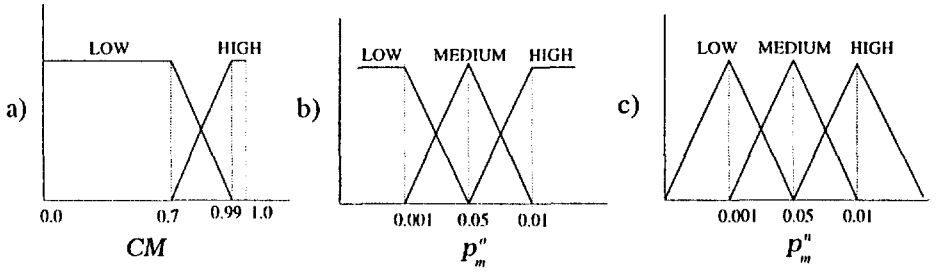

Fig. 1. Meaning of the linguistic terms associated with the inputs and the output

Fuzzy rules describe the relation between the inputs and output. Table 1 shows the fuzzy rule base used by the FLC presented.

Table 1. Fuzzy rule base for the control of p_m

Rule	CM	p_m^o	p_m^n
1	High	Low	Medium
2	High	Medium	High
3	High	High	Low
4	Low	Low	Low
5	Low	Medium	Low
6	Low	High	Medium

The absolute adaptive heuristic underling fuzzy rules 1,2, and 4-6 is: *"decrease p_m when progress is made, increase it when there are no improvements"*. If a stationary state is detected (CM high), there is a possible cause: p_m^o is too low, which induces a premature convergence, with the search process being trapped in a local optimum. With the previous heuristic, this problem would be suitably tackled, since p_m would be greater and so, more diversity is introduced with the possibility of escaping from the local optimum. However, another possible cause of a poor performance may be the use of a too high value for p_m^o, which does not allow the convergence to be produced for obtaining better individuals. Fuzzy rule 3 was included for dealing with this circumstance, since it proposes the use of a low p_m value when CM is high and p_m^o is high.

4 Experiments

Minimization experiments on the test suite, described in Subsection 4.1, have been carried out in order to study the behaviour of the proposed FLC in the

previous section for controlling p_m. In Subsection 4.2, the algorithms built in order to do this are described, and finally, in Subsection 4.3, the results are shown and analysed.

4.1 Test Suite

For the experiments, we have considered six frequently used test functions:

Sphere model (f_{Sph}) ([9]): $f_{Sph}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$, where $n = 10$ and $-5.12 \leq x_i \leq 5.12$. The fitness of the optimum is $f_{Sph}(x^*) = 0$. This test function is continuous, strictly convex, and unimodal.

Generalized Rosenbrock's function (f_{Ros})([9]): $f_{Ros}(\mathbf{x}) = \sum_{i=1}^{n-1}(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$, where $n = 2$ and $-5.12 \leq x_i \leq 5.12$. The fitness of the optimum is $f_{Ros}(x^*) = 0$. f_{Ros} is a continuous and unimodal function, with the optimum located in a steep parabolic valley with a flat bottom. This feature will probably cause slow progress in many algorithms since they must continually change their search direction to reach the optimum.

Generalized Rastrigin's function (f_{Ras}) ([3]): $f_{Ras}(\mathbf{x}) = a \cdot n + \sum_{i=1}^{n} x_i^2 - a \cdot \cos(\omega \cdot x_i)$, where $n = 10$, $a = 10$, and $\omega = 2\pi$. The fitness of its global optimum is $f_{Ras}(x^*) = 0$. This function is a scalable, continuous, separable, and multimodal, which is produced from f_{Sph} by modulating it with $a \cdot cos(\omega \cdot x_i)$.

One-Max function (f_{One}). For a string of binary digits, the fitness of a given string is the number of ones the string contains. The aim is to obtain a string containing all ones. A string length of 400 was used for the purposes of this study. To determine an individual's fitness, the value of this function is subtracted from 400 (maximum value), in order to assign a fitness of zero to the optimum, and handle the problem by means of minimization.

Fully deceptive order-3 function (f_{Dec}) ([14]). In deceptive problems there are certain schemata that guide the search toward some solution that is not globally competitive. It is due since the schemata that have the global optimum do not bear significance and so they may not proliferate during the genetic process. The deceptive problem used consists of the concatenation of 13 subproblems of length 3 (a 39-bit problem). The fitness for each 3-bit section of the string is given in Table 2. The overall fitness is the sum of the fitness of these deceptive subproblems. To obtain an individual's fitness, the value of this function is subtracted from 390 (maximum value). Therefore, the optimum has a fitness of zero.

Table 2. Fully deceptive order-3 problem

Chromosomes	000	001	010	100	110	011	101	111
Fitness	28	26	22	14	0	0	0	30

Royal Road (f_{RR}) ([13]) This is a 200-bit problem that is comprised of 25 contiguous blocks of eight bits, each of which scores 8 if all of the bits are set to one. Although there is no deception in this problem there is an amount of epistasis. Again, an individual's fitness is calculated by subtracting the value of this function from 200 (maximum value), being zero the fitness for the optimum.

4.2 Algorithms

For experiments, we have considered a generational GA model that applies a simple crossover operator and a mutation clock operator. The selection probability calculation follows *linear ranking* ([5]) ($\eta_{min} = 0.5$) and the sampling algorithm is the *stochastic universal sampling* ([6]). The *elitist strategy* ([9]) is considered as well.

The general features of the FLC are the following: the *min* operator is used for conjunction of clauses in the *IF* part of a rule, the *min* operator is used to fire each rule and the *center of gravity weighted by matching* strategy as the defuzzification operator is considered. This setting was chosen from [7], where experiments with several ones were tried, being the most effective.

Different algorithms were implemented which are differentiated according to the way followed for obtaining p_m. They are shown in Table 3.

Table 3. Algorithms

Algorithm	Features
GA1	$p_m = 0.001$ fixed during the run.
GA2	$p_m = 0.005$ fixed during the run.
GA3	$p_m = 0.01$ fixed during the run.
GA-RAN	p_m is randomly chosen from $[0.001, 0.01]$ for each generation.
GA-DET	Deterministic Control of the Mut. Prob. (Sect. 2.1).
GA-AIL	Adaptive Control at Individual-level of the Mut. Prob (Sect. 2.2).
GA-SELF	Self-Adaptive Control of the Mut. Prob ($\delta = 0.001$) (Sect. 2.3).
GA-FLC	Adaptive Control of the Mut. Prob. by FLC ($G = 50$) (Sect. 3).

Since we attempt to compare GA-FLC against other techniques for controlling p_m (GA-DET, GA-IL, and GA-SELF), we have considered that these techniques should handle the same range of possible p_m values than GA-FLC ($[0.001, 0.01]$). In order to do this, for the deterministic control of p_m (Sect. 2.1), we constrain $p_m(t)$ so that $p_m(0) = 0.01$ and $p_m(T) = 0.001$. For the adaptive control at individual-level of p_m (Sect. 2.2), a transformation was made from the interval considered by this technique ($[0, 1]$) into $[0.001, 0.01]$. Finally, for the self-adaptive control of p_m (Sect. 2.3), we consider $p_m^l = 0.001$ and $p_m^h = 0.01$.

The parameters of the test functions f_{Sph}, f_{Ros}, and f_{Ras} were encoded into bit strings using binary reflected Gray coding, with a number of binary genes assigned to each one of 20. The population size is 60 individuals and the crossover probability $p_c = 0.6$. We executed all the algorithms 30 times, each one with a maximum of 100000 evaluations.

4.3 Results

Table 4 shows the results obtained. The performance measures used are the following: 1) **A** performance: average of the best fitness function found at the end of each run, 2) **SD** performance: standard deviation, and 3) **B** performance: number of runs in which the algorithm achieved the best possible fitness value: $2.4e-10$ for f_{Sph}, $1.5e-9$ for f_{Ros}, $4.8e-8$ for f_{Ras} (they are not zero due to the use of the Gray coding), and zero for f_{One}, f_{Dec}, and f_{RR}.

Table 4. Results

Alg.	f_{Sph} A	f_{Sph} SD	f_{Sph} B	f_{Ros} A	f_{Ros} SD	f_{Ros} B	f_{Ras} A	f_{Ras} SD	f_{Ras} B
GA1	2.4e-10	0.0e+00	30	1.1e-01	2.2e-01	0	5.9e+00	2.9e+00	0
GA2	7.7e-08	1.1e-07	0	8.5e-03	2.7e-02	7	5.1e-05	5.9e-05	0
GA3	8.0e-06	2.0e-05	0	1.2e-05	6.3e-05	28	6.0e-02	2.2e-01	0
GA-RAN	2.2e-07	3.4e-07	0	6.9e-04	1.7e-03	8	8.4e-05	1.7e-04	0
GA-DET	2.8e-10	1.2e-10	25	5.6e-05	3.0e-04	11	3.3e-02	1.8e-01	20
GA-IL	2.4e-10	0.0e+00	30	4.3e-02	6.5e-02	0	1.2e+00	1.6e+00	7
GA-SELF	2.4e-10	0.0e+00	30	4.5e-02	1.3e-01	0	1.7e+00	1.9e+00	5
GA-FLC	2.4e-10	0.0e+00	30	6.4e-04	2.3e-03	15	6.1e-08	5.6e-08	27

Alg.	f_{One} A	f_{One} SD	f_{One} B	f_{Dec} A	f_{Dec} SD	f_{Dec} B	f_{RR} A	f_{RR} SD	f_{RR} B
GA1	0.0e+00	0.0e+00	30	9.7e+00	3.2e+00	0	4.2e+01	1.2e+01	0
GA2	8.7e+00	2.4e+00	0	5.1e+00	3.2e+00	3	6.2e+01	5.1e+00	0
GA3	3.2e+01	3.1e+00	0	0.0e+00	0.0e+00	30	6.8e+01	8.7e+00	0
GA-RAN	1.2e+01	2.1e+00	0	3.3e+00	3.0e+00	9	3.4e+01	8.5e+00	0
GA-DET	6.7e-02	2.5e-01	28	1.4e+00	1.5e+00	13	3.0e+01	1.0e+01	0
GA-IL	4.3e+01	2.5e+00	0	8.6e+00	3.1e+00	0	5.2e+01	8.2e+00	0
GA-SELF	2.3e-01	4.2e-01	23	9.1e+00	3.3e+00	0	2.1e+01	1.1e+01	3
GA-FLC	0.0e+00	0.0e+00	30	6.7e-01	1.2e+00	22	3.4e+01	9.3e+00	0

With regards to the GA versions with fixed p_m values (GA1, GA2, and GA3), we may underline a very reasonable fact, the best A and B measures for each test function are reached with different p_m values:

- For the easy test functions, f_{Sph} and f_{One}, with the lowest value, $p_m = 0.001$.
- For the functions with intermediate complexity, f_{Ras} and f_{RR}, with the moderate value, $p_m = 0.005$.
- For the most complex functions, f_{Ros} and f_{Dec}, with the highest value, $p_m = 0.01$.

A GA does not need too much diversity to reach the optimum of f_{Sph} and f_{One} since there is only one optimum which could be easily accessed. On the other side, for f_{Dec}, the diversity is fundamental for finding a way to lead towards the optimum. Also, in the case of f_{Ros}, diversity can help to find solutions close to the parabolic valley, and so avoid slow progress.

Now, considering the results of the GAs based on techniques for controlling p_m, we may observe that the one based on FLCs, GA-FLC, has the most robust behaviour with regards to the A and B measures, since for each function, it returns results that are very similar to the ones of the most successful GAs with fixed p_m values (the case of f_{Sph}, f_{One}, f_{Ros}, and f_{Dec}), or better than all them (the case of f_{RR} and f_{Ras}, for which it obtains very good results: 27 for B and 6.1e-8 for A). None of the remaining algorithms allows a better operation to be achieved.

In order to ascertain whether GA-FLC achieves this robust operation due to its adaptive control of p_m, and not for applying different p_m values during the run, we compare its results with the ones of GA-RAN, which works in this way (it selects, at random, a p_m value for each generation). We may observe that GA-FLC clearly improves the results of GA-RAN. This indicates that its adaptation ability is responsible for the performance improvement, i.e., it allows suitable p_m values to be used for producing a robust operation for test functions with different difficulties.

Finally, we need to point out that the GA applying a deterministic control of p_m, GA-DET, offers the most significant resistance against the results for GA-FLC. This good behaviour may be caused by the use of the heuristic *"first exploration, exploitation later"*, which, when considered for controlling other type of parameters associated with GAs, has produced successful results as well. An example is the non-uniform mutation operator for real-coded GAs ([21]). It controls the mutation steps size following this strategy, showing high performance.

5 Conclusions

This paper presented a technique for the adaptive control of p_m based on the use of FLCs. In particular, an FLC has been designed, which takes into account the p_m value used during the last generations and a measure that quantifies the progress performed by the GA during these generations, and returns a new p_m that will be used for attempting to gain a better evolution quality during the next generations.

The principal conclusions derived from the results of experiments carried out are the following:

- The technique presented is the most effective one for controlling p_m as compared with other techniques proposed in the GA literature that have been considered for the experiments.
- The adaptation ability of this technique allows suitable p_m values to be used for producing a robust operation for test functions with different difficulties.

Therefore, we may conclude that the adaptive control of p_m by means of FLCs is a suitable way for improving the results of GAs. This conclusion show promise in the use of this technique for future applications and extensions. They include the following: 1) design of FLCs for the adaptive control at the individual level of p_m, and 2) study the adaptation of p_m by FLCs for other types of GAs (real-coded GAs, genetic programming, etc.).

References

1. Angeline, P.J.: Adaptive and Self-Adaptive Evolutionary Computations. Computational Intelligence: A Dynamic Systems Perspective, M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel and T. Fukuda (Eds.), (Piscataway, NJ, IEEE Press, 1995) 152-163.
2. Bäck, T.: The Interaction of Mutation Rate, Selection, and Self-adaptation within Genetic Algorithm. Parallel Problem Solving from Nature 2, R. Männer, B. Manderick, (Eds.), (Elsevier Science Publishers, Amsterdam, 1992) 85-94.
3. Bäck, T.: Self-adaptation in Genetic Algorithms. Proc. of the First European Conference on Artificial Life, F.J. Varela, P. Bourgine, (Eds.), (The MIT Press, Cambridge, MA, 1992) 263-271.
4. Bäck, T., Schütz, M.: Intelligent Mutation Rate Control in Canonical Genetic Algorithms. Foundation of Intelligent Systems 9th Int. Symposium, Z.W Ras, M. Michalewicz (Eds.), (Springer, 1996) 158-167.
5. Baker, J.E.: Adaptive Selection Methods for Genetic Algorithms. Proc. First Int. Conf. on Genetic Algorithms, J.J. Grefenstette, (Ed.), (L. Erlbaum Associates, Hillsdale, MA, 1985) 101-111.

6. Baker, J.E.: Reducing Bias and Inefficiency in the Selection Algorithm. Proc. Second Int. Conf. on Genetic Algorithms, J.J. Grefenstette, (Ed.), (L. Erlbaum Associates , Hillsdale, MA, 1987) 14-21.
7. Cordón, O., Herrera, F., Peregrín, A.: Applicability of the Fuzzy Operators in the Design of Fuzzy Logic Controllers. Fuzzy Sets and Systems 86(1) (1997) 15-41.
8. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Trans. Evolutionary Computation 3(2) (1999) 124-141.
9. De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral Dissertation, University of Michigan (1975).
10. Driankow, D., Hellendoorn, H., Reinfrank, M.: An Introduction to Fuzzy Control, Springer-Verlag, Berlin (1993).
11. Eshelman, L.J., Schaffer, J.D.: Preventing Premature Convergence in Genetic Algorithms by Preventing Incest. Proc. of the Fourth Int. Conf. on Genetic Algorithms, R. Belew, L.B. Booker (Eds.), (Morgan Kaufmmann, San Mateo, 1991) 115-122.
12. Fogarty, T.C.: Varying the Probability of Mutation in the Genetic Algorithm. Proc. of the Third Int. Conf. on Genetic Algorithms, J. David Schaffer, (Ed.), (Morgan Kaufmann Publishers, San Mateo, 1989) 104-109.
13. Forrest, S., Mitchell, M.: Relative Building Block Fitness and the Building Block Hypothesis. Foundations of Genetic Algorithms-2, L.Darrell Whitley (Ed.),(Morgan Kaufmann Publishers, San Mateo, 1993) 109-126.
14. Goldberg D.E., Korb, B., Deb, K.: Messy Genetic Algorithms: Motivation, Analysis, and First Results. Complex Systems 3 (1989) 493-530.
15. Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Trans. on Systems, Man, and Cybernetics 16 (1986) 122-128.
16. Herrera, F., Lozano, M.: Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers. Genetic Algorithms and Soft Computing, F. Herrera, J.L. Verdegay (Eds.), (Physica-Verlag, 1996) 95-125.
17. Herrera, F., Lozano, M.: Adaptive Genetic Operators Based on Coevolution with Fuzzy Behaviors. To appear in IEEE Trans. on Evolutionary Computation.
18. Holland, J.H.: Adaptation in Natural and Artificial Systems. The MIT Press, London (1992).
19. Lee, M.A., Takagi, H.: Dynamic Control of Genetic Algorithms Using Fuzzy Logic Techniques. Proc. of the Fifth Int. Conf. on Genetic Algorithms, S. Forrest (Ed.), (Morgan Kaufmmann, San Mateo, 1993) 76-83.
20. Lee, M.A., Takagi, H.: A Framework for Studying the Effects of Dynamic Crossover, Mutation, and Population Sizing in Genetic Algorithms. Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms, T. Furuhashi (Ed.), (Springer-Verlag, 1994) 111-126.
21. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York (1992).
22. Shi, Y., Eberhart, R., Chen Y.: Implementation of Evolutionary Fuzzy Systems. IEEE Transactions of Fuzzy Systems 7(2) (1999) 109-119.
23. Srinivas, M., Patnaik, L.M.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. IEEE Trans. on Systems, Man, and Cybernetics 24(4) (1994) 656-667.
24. Tuson, A.L., Ross, P.: Adapting Operator Settings in Genetic Algorithms. Evolutionary Computation 6(2) (1998) 161-184.
25. Xu, H.Y., Vukovich, G., Ichikawa, Y., Ishii, Y.: Fuzzy Evolutionary Algorithms and Automatic Robot Trajectory Generation. Proc. of the First IEEE Conference on Evolutionary Computation, (1994) 595-600.
26. Zeng, X., Rabenasolo, B.: A Fuzzy Logic Based Design for Adaptive Genetic Algorithms. Proc. of the Second European Congress on Intelligent Techniques and Soft Computing, (1994) 1532-1539.