Original paper

Soft Computing 7 (2003) 545–562 © Springer-Verlag 2003
DOI 10.1007/s00500-002-0238-y

Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions

F. Herrera, M. Lozano

Abstract The genetic algorithm behaviour is determined by the exploitation and exploration relationship kept throughout the run. Adaptive genetic algorithms, that dynamically adjust selected control parameters or genetic operators during the evolution have been built. Their objective is to offer the most appropriate exploration and exploitation behaviour to avoid the premature convergence problem and improve the final results. One of the adaptive approaches are the adaptive parameter setting techniques based on the use of fuzzy logic controllers, the fuzzy adaptive genetic algorithms (FAGAs). In this paper, we analyse the FAGAs in depth. First, we describe the steps for their design and present an instance, which is studied from an empirical point of view. Then, we propose a taxonomy for FAGAs, attending on the combination of two aspects: the level where the adaptation takes place and the way the Rule-Bases are obtained. Furthermore, FAGAs belonging to different groups of the taxonomy are reviewed. Finally, we identify some open issues, and summarise a few new promising research directions on the topic. From the results provided by the approaches presented in the literature and the experimental results achieved in this paper, an important conclusion is obtained: the use of fuzzy logic controllers to adapt genetic algorithm parameters may really improve the genetic algorithm performance.

Keywords Adaptive genetic algorithms, Fuzzy logic controllers

1 Introduction

The behaviour of the genetic algorithms (GAs) [25] is strongly determined by the balance between exploration (to investigate new and unknown areas in a search space) and exploitation (to make use of knowledge acquired by exploration to reach better positions on the search space). The GA control parameter settings, such as mutation probability, crossover probability, and population size, are key factors in the determination of the exploitation versus exploration tradeoff. It has long been acknowledged that they have a significant impact on GA performance [28]. If poor settings are used, the exploration/exploitation balance may not be reached in a profitable way; the GA performance shall be severely affected due to the possibility of premature convergence.

Finding robust control parameter settings is not a trivial task, since their interaction with GA performance is a complex relationship and the optimal ones are problem-dependent [4]. Furthermore, different control parameter values may be necessary during the course of a run to induce an optimal exploration/exploitation balance. For these reasons, adaptive GAs (AGAs) have been built that dynamically adjust selected control parameters or genetic operators during the course of evolving a problem solution. Their objective is to offer the most appropriate exploration and exploitation behaviour [1, 19, 22, 31, 47, 53].

Fuzzy logic controllers (FLCs) [17] provide a tool which can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. They are particularly suited to model the relationship between variables in environments that are either ill-defined or very complex.

The adaptation of GA parameters is one such complex problem that may benefit from the use of FLCs, getting the called fuzzy AGAs (FAGAs). The Rule-Bases of FLCs facilitate the capture and representation of a broad range of adaptive strategies for GAs (so, they may be the support for the automatic learning of such strategies). The main idea of FAGAs 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 shall be used by the GA. Figure 1 shows this process.

The goal of this paper is to report on an extensive study of FAGAs, according to the following three points of view:

- First, we describe the steps for their design, which are then applied to build an instance of FAGA proposed in [32]. It adapts the mutation probability during the run using an FLC. We have considered the adaptation for this particular parameter since it can determine directly the degree of population diversity, which is the main factor to avoid the premature convergence problem. We carry out an empirical study of the instance, where we compare its results with the ones for other (non fuzzy) methods to control the mutation probability. The analysis of the results give us an excellent knowledge on the FAGA behaviour.

Published online: 14 July 2003

F. Herrera (✉), M. Lozano
Department of Computer Science
and A.I. University of Granada, 18071 – Granada, Spain
e-mail: herrera@decsai.ugr.es

This research has been supported by DGICYT PB98-1319.

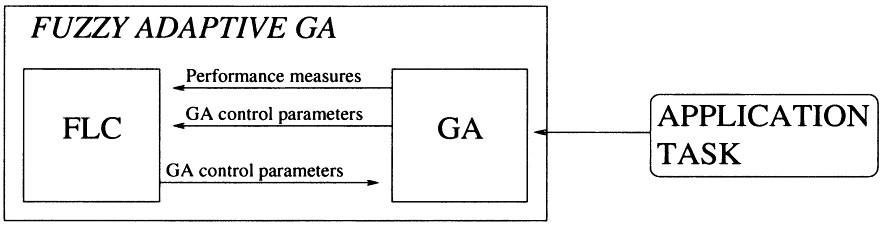

Fig. 1. FAGA model

- Second, we present a taxonomy that groups FAGAs in different categories, attending on the level where the adaptation takes place (population-level and individual-level) and the way the Rule-Bases are derived (through the knowledge of GA experts, using an Offline learning mechanism, or by means of an Online learning method). Furthermore, we review some representative approaches that belong to the different categories of FAGAs.
- Finally, we attempt to identify some open issues, and summarise a few new promising research directions on the topic.

The paper is set up as follows. In Sect. 2, we introduce the main features of FLCs. In Sect. 3, we explain the design steps of FAGAs and outline the instance of FAGA. In Sect. 4, we present a taxonomy for FAGAs and review some representative approaches presented in the literature (Appendix C includes an additional description of other FAGA instances). In Sect. 5, we attempt to identify some open issues and summarise a few new promising research directions for this type of GA adaptation. Finally, in Sect. 6, we point out some concluding remarks.

2 Preliminaries: fuzzy logic controllers

The essential part of the FLCs is a set of IF-THEN linguistic control rules, whose antecedents and consequents are composed of fuzzy statements, related by the dual concepts of fuzzy implication and the compositional rule of inference. The methodology of the FLCs appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, inexactly, or uncertainly.

An FLC is composed by a Knowledge Base, that includes the information given by the expert in the form of linguistic control fuzzy rules, a Fuzzification Interface, which has the effect of transforming crisp data into fuzzy sets, an Inference System, that uses them together with the Knowledge Base to make inference by means of a reasoning method, and a Defuzzification Interface, that translates the fuzzy control action thus obtained to a real control action using a defuzzification method. The generic structure of an FLC is shown in Fig. 2.

The Knowledge Base encodes the expert knowledge by means of a set of IF-THEN rules, which are a conditional statement with the form:

If *a set of conditions are satisfied* **Then** *a set of consequences can be inferred*

in which the antecedent is a condition in its application domain, the consequent is a control action to be applied in the controlled system and both antecedent and consequent are associated with fuzzy concepts, that is, linguistic terms (notion of fuzzy rule).

The Knowledge Base is composed of two components:

- A Data Base, containing the linguistic term sets considered in the linguistic rules and the membership functions defining the semantics of the linguistic labels. Each linguistic variable involved in the problem will have associated a fuzzy partition of its domain representing the fuzzy set associated with each of its linguistic terms.
- A Rule-Base, comprised of a collection of linguistic rules that are joined by the also operator. In other words, multiple rules can fire simultaneously for the same input.

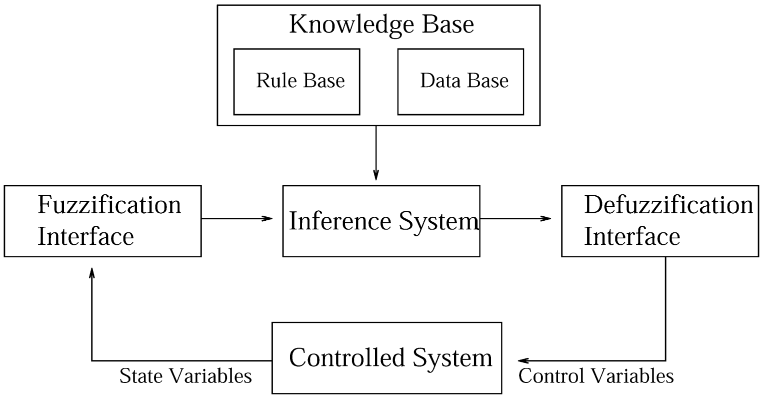

Fig. 2. Generic structure of an FLC

For more information about fuzzy systems and FLC, the following books may be consulted: [13] and [17].

3 Designing FAGAs

In this section, we shortly describe the issues that should be tackled in order to build the FLC used by an FAGA. They include the choice of the inputs and outputs, the definition of the Data Base associated with them (Sect. 3.1), and the specification of the Rule-Base (Sect. 3.2) [31]. Furthermore, we define and study empirically an FAGA instance (Sect. 3.3).

3.1 Inputs, outputs, and Data Base

Inputs. They should be robust measures that describe GA behaviour and the effects of the genetic setting parameters and genetic operators. Some possible inputs may be: diversity measures, maximum, average, and minimum fitness, etc. The current control parameters may also be considered as inputs.

Outputs. They indicate values of control parameters or changes in these parameters. In [58], the following outputs were reported: mutation probability, crossover probability, population size, selective pressure, the time the controller must spend in a target state in order to be considered successful, the degree to which a satisfactory solution has been obtained, etc.

Data Base. Each input and output should have an associated set of linguistic labels. The meaning of these labels is specified through membership functions of fuzzy sets, the fuzzy partition, contained in the Data Base. Thus, it is necessary that every input and output have a bounded range of values in order to define these membership functions over it.

3.2 Rule-Base

After selecting the inputs and outputs and defining the Data Base, the fuzzy rules describing the relations between them should be defined. They facilitate the capture and representation of a broad range of adaptive strategies for GAs.

Although, the experience and the knowledge of GA experts may be used to derive Rule-Bases, many authors have found difficulties to do this. In this sense, the following three reflections were quoted by different authors:

"*Although much literature on the subject of GA control has appeared, our initial attempts at using this information to manually construct a fuzzy system for genetic control were unfruitful*" [40].

"*Statistics and parameters are in part universal to any evolutionary algorithm and in part specific to a particular application. Therefore it is hard to state general fuzzy rules to control the evolutionary process*" [58].

"*The behaviour of GAs and the interrelations between the genetic operators are very complex. Although there are many possible inputs and outputs for the FLCs, frequently fuzzy rule-bases are not easily available: finding good fuzzy rule bases is not an easy task*" [31].

Automatic learning mechanisms to obtain Rule-Bases have been introduced to avoid this problem. By using these mechanisms, relevant relations and membership functions may be automatically determined and may offer insight to understand the complex interaction between GA control parameters and GA performance [40]. Two types of Rule-Base learning techniques have been presented: the *Offline* learning technique [40, 41] and the *Online* learning technique [33]:

- The *Offline* learning mechanism is an evolutionary algorithm that is executed once, before the operation of the FAGA, however it has associated a high computational cost. It works considering a fixed set of test functions, following the same idea as the meta-GA of Grefenstette [28]. Unfortunately, the test functions may have nothing to do with the particular problem to be solved, which may limit the robustness of the Rule-Bases returned.
- In the *Online learning* technique, the Rule-Bases used by the FLCs come from an evolutionary process that interacts concurrently with the GA to be adapted. The learning technique underlying this approach only takes into account the problem to be solved (in contrast to the previous one, which never considers it). In this way, the Rule-Bases obtained shall specify adaptation strategies particularly appropriate for this problem.

3.3 An instance of FAGA: the fuzzy adaptive mutation probability

The objective of this section is to show, by means of a simple and descriptive example, that the use of FLCs may really improve GA performance. We describe an FAGA instance that integrates an FLC to adapt the mutation probability (p_m) during the run [32]. This system is called *fuzzy adaptive mutation probability* and denoted as GA-FAMP in the experiments. We tackled the issue of adapting this parameter since it determines directly the operation of the mutation operator, which is responsible for the generation of the population diversity, arising as an important element to solve the premature convergence problem.

3.3.1 Main ideas

The FLC is fired every G generations and p_m is fixed over the generations in these time intervals. It takes into account the p_m value used during the last G generations and the improvement achieved on f_b (fitness for the best element found so far). Then, it computes a new value for p_m, which shall be used during the next G generations. Its goal is to observe the effects of a p_m value on the GA performance during G generations, and produce a new p_m value that properly replies against a possible poor rate of convergence, or 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).

3.3.2
Inputs, output, and Data Base

The FLC proposed has two inputs:

- The current mutation probability, p_m^o, which shall 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. $p_m = 0.001$ [15] and $p_m = 0.01$ [28]).
- A convergence measure (minimization is assumed):$CM = f_b^c/f_b^o$, 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 shall 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 Fig. 3b. The set of linguistic labels for CM is $\{Low, High\}$. Their meanings are shown in Fig. 3a.

The output of the FLC is the new p_m value, $p_m^n \in [0.001, 0.01]$, which shall 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 Fig. 3c.

3.3.3
Rule-Base

Fuzzy rules describe the relation between the inputs and output. Table 1 shows the Rule-Base used by the FLC presented.

There are two heuristics associated with this Rule-Base:

- The heuristic of 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.
- 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 to obtain better individuals. Fuzzy rule 3 was included to deal with this circumstance, since it proposes the use of a low p_m value when CM is high and p_m^o is high.

Table 1. 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

3.3.4
Experiments and analysis of the results

Minimisation experiments on the test suite, described in Appendix A and summarised in Table 2, have been carried out in order to study the behaviour of the fuzzy adaptive mutation probability. We compare its results with the ones for other (non fuzzy) methods appeared in the GA literature to control p_m, which are reviewed in Appendix B.

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 ($\eta_{\min} = 0.5$) [7] and the sampling algorithm is the stochastic universal sampling [8]. The elitist strategy [15] is considered as well.
The features of all of the algorithms compared in the experiments are shown in Table 3.

Since we attempt to compare GA-FAMP with other techniques for controlling the mutation probability (GA-DET, GA-IL, and GA-SELF, see Table 3 and appendix B for a short description), we have considered that these techniques should handle the same range of possible p_m values ($[0.001, 0.01]$). In order to do this, for the deterministic control of p_m, 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, a transformation was made from the interval con-

Table 2. Test suite

Test function	Fitness of the opt.
Sphere model (f_{Sph})	0
Generalized Rosenbrock's function (f_{Ros})	0
Generalized Rastrigin's function (f_{Ras})	0
One-max function (f_{One})	0
Fully deceptive order-3 function (f_{Dec})	0
Royal Road (f_{RR})	0

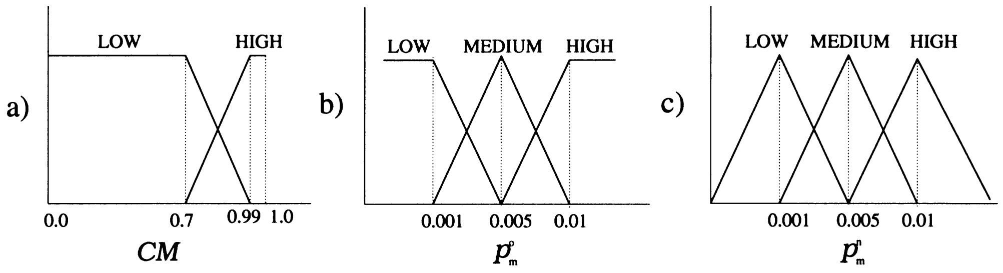

Fig. 3. Meaning of the linguistic terms associated with the inputs and output

Table 3. Algorithms for experiments

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	For each generation, p_m is chosen, at random, from $[0.001, 0.01]$
GA-DET	Deterministic control of p_m
GA-IL	Adaptive control at individual-level of p_m
GA-SELF	Self-adaptive control of p_m ($\delta = 0.001$)
GA-FAMP	Fuzzy adaptive mutation probability ($G = 50$)

sidered by this technique ($[0, 1]$) into $[0.001, 0.01]$. Finally, for the self-adaptive control of p_m, we consider $p_m^l = 0.001$ and $p_m^h = 0.01$.

The parameters of the test functions $f_{\text{Sph}}, f_{\text{Ros}}$, 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 run all the algorithms 30 times, each one with a maximum of 100, 000 evaluations.

The general features of the FLC used by GA-FAMP 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.
- The center of gravity weighted by matching strategy as the defuzzification operator is considered.

This setting was chosen from [12], where the authors study the combination of inference systems and defuzzification methods using different applications and defining a degree of behaviour. The previous combination was the most effective one, in the sense that it obtained the best behaviour for all the applications.

Table 4 shows the results obtained. The performance measures used are the following:

- **A** performance: average of the best fitness function found at the end of each run.
- **E** performance: average of the number of evaluations after which improvements in solution quality were no longer obtained.
- **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 $f_{\text{One}}, f_{\text{Dec}}$, and f_{RR}.

Moreover, a two-sided t-test (H_o : means of the two groups are equal, H_a: means of the two group are not equal) at 0.05 level of significance was applied in order to ascertain if differences in the A and E performance measures for GA-FAMP are significant when compared with the ones for the other algorithms. The direction of any significant differences is denoted either by:

- a plus sign $(+)$ for an improvement in the corresponding performance, or
- a minus sign $(-)$ for a reduction, or
- an approximate sign $(\sim)$ for non significant differences.

The places in Table 4 where these signs do not appear correspond with the performance values for GA-FAMP.

GA1, GA2, and GA3. With regards to the GA versions with fixed p_m values, we may underline a very reasonable fact:

Table 4. Results

Alg.	f_{Sph}			f_{Ros}			f_{Ras}		
	A	E	B	A	E	B	A	E	B
GA1	2.4e-10 ∼	35261 ∼	30	1.1e-01 +	99761 +	0	5.9e + 00 +	71731 ∼	0
GA2	7.7e-08 +	97336 +	0	8.5e-03 ∼	86246 ∼	7	5.1e-05 +	98339 +	0
GA3	8.0e-06 +	98210 +	0	1.2e-05 ∼	50165 −	28	6.0e-02 +	98872 +	0
GA-RAN	2.2e-07 +	98122 +	0	6.9e-04 ∼	81931 ∼	8	8.4e-05 +	98680 +	0
GA-DET	2.8e-10 ∼	97851 +	25	5.6e-05 ∼	49589 −	11	3.3e-02 ∼	97781 +	20
GA-IL	2.4e-10 ∼	52542 +	30	4.3e-02 +	79328 ∼	0	1.2e + 00 +	84610 ∼	7
GA-SELF	2.5e-10 ∼	86178 +	28	3.2e-02 +	92529 +	1	7.0e-01 +	96046 +	5
GA-FAMP	2.4e-10	38273	30	6.4e-04	81508	15	6.1e-08	77990	27
	f_{One}			f_{Dec}			f_{RR}		
Alg.	A	E	B	A	E	B	A	E	B
GA1	0.0e + 00 ∼	20988 ∼	30	9.7e + 00 +	1950 −	0	4.2e + 01 +	89118 +	0
GA2	8.7e + 00 +	91511 +	0	5.1e + 00 +	55417 ∼	3	3.3e + 01 ∼	76974 ∼	0
GA3	3.2e + 01 +	94845 +	0	0.0e + 00 −	31289 −	30	6.8e + 01 +	74819 ∼	0
GA-RAN	1.2e + 01 +	92156 +	0	3.3e + 00 +	63515 ∼	9	3.4e + 01 ∼	76784 ∼	0
GA-DET	6.7e-02 ∼	96342 +	28	1.4e + 00 +	30704 −	13	3.0e + 01 ∼	86493 +	0
GA-IL	4.3e + 01 +	57891 +	0	8.6e + 00 +	27396 −	0	5.2e + 01 +	77529 ∼	0
GA-SELF	3.3e-01 +	63370 +	21	9.4e + 00 +	28699 −	1	2.1e + 01 −	82022 ∼	1
GA-FAMP	0.0e + 00	21709	30	6.7e-01	68655	22	3.4e + 01	80097	0

the best A and B measures for each test function are reached with different p_m values.

GA-FAMP vs. GA1, GA2, and GA3. For most test functions, GA-FAMP returns A results that are very similar (or superior) to the ones for the most successful GAs with fixed p_m values (see the t-test results).

Therefore, we may consider that GA-FAMP achieves a robust operation, in the sense that it obtains a significant performance for each one of the test functions, which have different difficulties. In fact, we may underline that none of the remaining algorithms allows a better operation to be achieved.

GA-FAMP vs. GA-DET, GA-IL, and GA-SELF. Now, we compare the results for GA-FAMP with the ones for the other techniques to control p_m. In general, GA-FAMP improves the A and B results for GA-IL and GA-SELF. Only GA-DET offers a significant resistance against the results for GA-FAMP. The t-test results for the A measure indicate that no significant differences exist between GA-DET and GA-FAMP, with regards to this performance measure. However, the t-test results for the E measure shows that GA-FAMP clearly outperforms GA-DET. This means that although these algorithms reach the same solution quality, GA-FAMP achieves a higher speed of search than GA-DET. Furthermore, we may observe that GA-FAMP obtains a better B measure than GA-DET for most functions.

Adaptive Control of p_m in GA-FAMP. In order to ascertain whether GA-FAMP achieves its robust operation due to the adaptive control of p_m, and not to the application of different p_m values during the run, we compare its results with the ones for GA-RAN. This algorithm works in this way, since it selects, at random, a p_m value for each generation. We observe that GA-FAMP improves the A, E, and B performances for GA-RAN on most functions. This indicates that the adaptation ability of GA-FAMP is responsible for the performance improvement.

Conclusions. The principal conclusions derived from this empirical study are the following:

- The fuzzy adaptive mutation probability is the most effective technique to control p_m as compared with other non fuzzy techniques proposed in the GA literature, which have been considered for the experiments.
- The adaptation ability of this technique allows suitable p_m values to be used to produce a robust operation for test functions with different difficulties.

4
A taxonomy for FAGAs

In this section, we present a taxonomy for FAGAs, attending on the combination of two aspects:

- The way in which the Rule-Bases are derived:
  - Through the *expertise, experience*, and *knowledge* on GAs, which have become available as a result of empirical studies conducted over a number of years.
  - Using an *Offline learning mechanism*, which finds Rule-Bases that induce a suitable FAGA behaviour on a fixed set of test functions. It is executed before the application of the FAGA on any real problem.
  - By means of an *Online learning mechanism*, which learns Rule-Bases during the run of the FAGA on a real problem.

A discussion about these methods was included in Sect. 3.2.

- The level where the adaptation takes place in FAGAs:
  - *Population-level* adaptations adjust control parameters that apply for the entire population.
  - *Individual-level* adaptations tune control parameters that have an effect on the individual members of the population.

Most FAGAs presented in the literature involve population-level adaptation. However, adaptive mechanisms at the individual level based on FLCs may be interesting to adjust control parameters associated with genetic operators [31, 33, 69]. In this case, the control parameters will be defined on individuals instead of on the whole population. Inputs to the FLCs may be central measures and/or measures associated with particular chromosomes or sets of them, and outputs may be control parameters associated with genetic operators that are applied to those chromosomes.

A justification for this approach is that it allows for the application of different search strategies in different parts of the search space. This is based on the reasonable assumption that, in general, the search space will not be homogeneous, and that different strategies will be better suited to different kinds of sublandscapes [1, 53].

The marked places in Table 5 show the four categories of the taxonomy to which belong the particular cases of FAGAs that were presented in the GA literature.

The next sections are devoted to review representative instances of every category. In Appendix C, we describe the remaining approaches presented in the literature.

4.1
Rule-Base derivation through expert knowledge and population-level adaptation

Most FAGA instances presented in the GA literature belong to this category. In this section, we describe the approach of [52]. Other models of this type of FAGAs are reviewed in Appendix C.

4.1.1
Main ideas

The authors claimed that from experience, we know when the fitness is high, e.g., at the end of the run, low crossover probability and high mutation probability are often preferred. Also, when the best fitness is stuck at one value for a long time, the system is often stuck at a local minimum

Table 5. FAGA categories with instances in the literature

	Expert knowledge	Offline learning	Online learning
Population-level	X	X	
Individual-level	X		X

in a local neighbourhood, so the system should probably concentrate on exploring rather than exploiting; that is, the crossover probability should be decreased and mutation probability should be increased. A similar situation exists for the variance of the fitness of the population. When the variance is low, mutation should be emphasised, while when variance is high, crossover should be stressed.

4.1.2 Inputs, outputs, and Data Base

According to this kind of knowledge, in [52], an FLC was developed to adjust the crossover and mutation probabilities with best fitness (BF), number of generations for unchanged best fitness (UN), and variance of fitness (VF) as input variables.

The set of linguistic labels associated with the inputs and outputs is $L = \{Low, Medium, High\}$. The meaning of these labels are shown in Fig. 4.

4.1.3 Rule-Base

Eight fuzzy rules were used:

If BF is low **then** p_m is low and p_c is high.

If BF is medium and UN is low **then** p_m is low and p_c is high.

If BF is medium and UN is medium **then** p_m is medium and p_c is medium.

If UN is high and VF is medium **then** p_m is high and p_c is low.

If BF is high and UN is low **then** p_m is low and p_c is high.

If BF is high and UN is medium **then** p_m is medium and p_c is medium.

If UN is high and VF is low **then** p_m is high and p_c is low.

If UN is high and VF is high **then** p_m is low and p_c is low.

4.1.4 Results

A FAGA with this FLC was applied to design a fuzzy classification system for the Iris data set. In particular, the membership function shapes and types and the fuzzy rule set including the number of rules inside it were evolved using the FAGA. The experiments showed that using the fuzzy expert system to adapt p_m and p_c significantly fewer generations were required to get the same performance compared to holding these parameters constant.

4.2 Rule-Base derivation through expert knowledge and individual-level adaptation

The FAGA presented in [69] is one of the first approaches with individual-level adaptation. The authors used their knowledge about GAs to obtain the Rule-Bases.

4.2.1 Main ideas

In this FAGA, the crossover probability and the mutation probability are defined on specific individuals of the population using several FLCs that take into account fitness values of individuals and distances between individuals. The next sections present the design of the FLC that adapts the crossover probability, p_c.

4.2.2 Inputs, outputs, and Data Base

The following measures were considered as inputs to the FLC, where X and Y are two chromosomes to be crossed (maximisation is assumed).

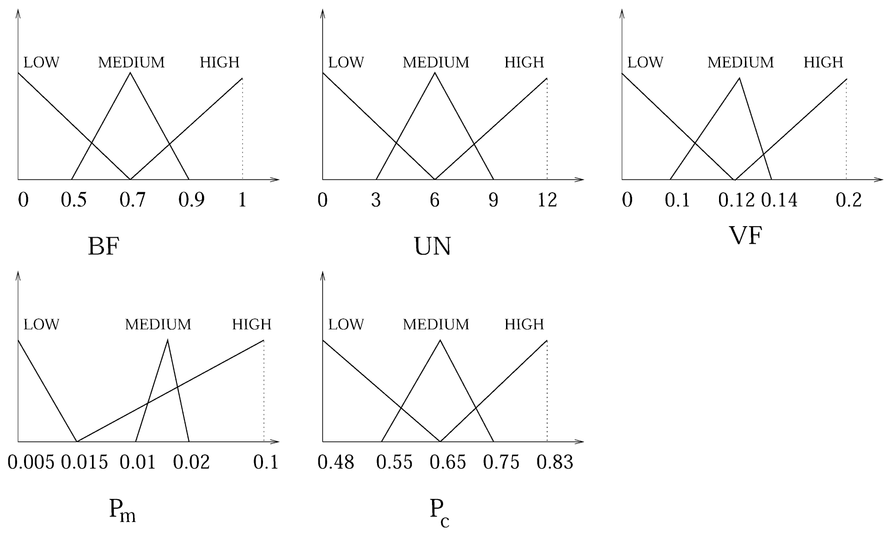

Fig. 4. Meaning of the linguistic terms associated with the inputs and outputs

- *Variance of fitness values*: $Var = \left(f_{\max} - \bar{f}\right)/(f_{\max} - f_{\min})$, where $\bar{f}$ is the average of all fitness values, and $f_{\max}$ and $f_{\min}$ the maximal and the minimal fitness values, respectively.
- *Distance between the fitness value of the best parent and* $f_{\max}$: $G = (f_{\max} - \max\{f(X), f(Y)\})/(f_{\max} - f_{\min})$.
- *Distance between X and Y*: $D = d(X, Y)$.
- *Normalised fitness values*: $f_1 = f(X)/f_{\max}$ and $f_2 = f(Y)/f_{\max}$.

Var is overall for the entire population, and G, f_1, f_2, and D are measures defined on specific samples. All the measures were included in $[0, 1]$. Their set of linguistic labels is $L = \{Small, Big\}$. For each input, the membership functions are defined in Figure 5a, where $input \in \{Var, G, f_1, f_2\}$ is any input variable of the controller and ω is the corresponding parameter obtained from the experience of the authors on GAs.

The output was p_c. The set of linguistic labels for p_c is $L = \{Small, Big\}$ (their meanings are depicted in Fig. 5b).

4.2.3 Rule-Base

Next, we show the Rule-Base considered to obtain the p_c value for each pair of parents, (X, Y).

If G is big **then** p_c is big.

If Var is small and G is small **then** p_c is small.

If D is small and f_1 is big and f_2 is big **then** p_c is big.

If D is small and $(f_1$ or $f_2)$ is small **then** p_c is small.

If D is big **then** p_c is big.

These fuzzy rules attempt to implement the following principles (where apparently there are some conflicts):

- Maintain the diversity in the population; two distant samples have more chance to be selected for crossover.
- Enhance the searching in optimal regions; two near samples with high fitness values have more chance to be selected for crossover.
- Avoid convergence to local optima; crossover operations have to be enhanced if the variance of fitness values is very small.
- Stabilise optimal populations; crossover operations have to be reduced if the specific fitness values of the samples to be selected are close enough to the maximal fitness value of the current population.

The FLC proposed to adapt the mutation probability was designed so that mutation operations can be enhanced when the GA tends to a local optimum. Furthermore, they can reduce when the current population is in strong variations or the globally optimal population is obtained.

4.2.4 Results

The results of the FAGA proposed on a multimodal test function were compared with the ones for a simple GA. On the one hand, populations in the FAGA were more diversified than those in the simple GA, on the other, it was easier for the FAGA to lead to the global maximum and its convergence behaviour was better than the simple GA.

4.3 Rule-Base derivation using offline learning and population-level adaptation

In [40, 41], an automatic Offline learning process was proposed to obtain suitable Rule-Bases along with their Data Bases.

4.3.1 Main ideas

The mechanism is based on an additional GA whose chromosomes code possible Rule-Bases together with their corresponding Data Bases. The fitness function value for a chromosome is calculated using the Online and Offline measures [15] obtained from an FAGA that uses the Rule-Base coded in such chromosome on each one of the five De Jong's test functions. After the meta-level GA completed 1000 fitness function evaluations, the best Rule-Base reached is returned. The underlying idea is very similar to the one of the meta-GA of Grefenstette [28].

4.3.2 Inputs and outputs

In order to study this mechanism, an FAGA was developed using three FLCs. All consider the following three inputs:

- Two phenotypical diversity measures (minimisation is assumed): $PD_1 = f_{\text{best}}/\bar{f}$ and $PD_2 = \bar{f}/f_{\text{worst}}$, where $\bar{f}$, f_{best}, and f_{worst} are the average, best, and the worst fitness in the current population, respectively. PD_1 and PD_2 belong to the interval $[0, 1]$. If they are near to 1, convergence has been reached, whereas if they are near to 0, the population shows a high level of diversity.
- The change in the best fitness since the last control action.

The outputs of the three FLCs are variables that control variations in the current p_m, p_c, and N (population size), respectively. For example, such output for the control of N, represents the degree to which the current N should vary. The new population size is computed by multiplying the output value by the current N.

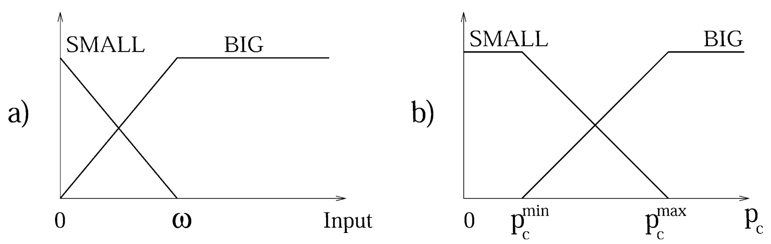

Fig. 5. Definition of membership function for input and output variables

4.3.3 Results

The automatic learning mechanism was executed to obtain both the Data Base and the Rule-Base for this example. An FAGA using the resultant FLCs was applied to a particular problem: the inverted pendulum control task. The results obtained exhibited better performance than a simple static GA. Other experiments aimed at isolating the effects of the fuzzy adaptation of N, p_c, and p_m showed that the adaptation of the mutation probability contributes most to high performance [41].

In [56], FLCs are used for dynamic scheduling of parameters (population size, crossover rate, and mutation rate) of a GA applied on an agile manufacturing application. A high-level GA was used as well to automatically identify and tune the Knowledge Base.

4.4 Rule-Base derivation using online learning and individual-level adaptation

In [33], it is proposed an Online learning mechanism to obtain Rule-Bases for FAGAs that adapt control parameters associated with the genetic operators. It is called Coevolution with Fuzzy Behaviours.

4.4.1 Main ideas

The main ideas of this proposal are:

- It incorporates genetic operator adaptation at an individual-level based on FLCs. Control parameter values for a genetic operator are computed for each set of parents that undergo it, using an FLC that considers particular features associated with the parents as inputs.
- The Rule-Bases of the FLCs applied are learnt implicitly throughout the run by means of a separate GA (denoted as FBs-GA) that *coevolves* with the one that applies the genetic operator to be controlled (denoted as main-GA). Both GAs have an influence on the other:
  - On the one hand, Rule-Bases in FBs-GA induce parameter values for the genetic operator applied to main-GA (FBs-GA $\rightarrow$ main-GA).
  - On the other, they evolve according to the performance of the operator on the elements of main-GA (main-GA $\rightarrow$ FBs-GA).

  The goal of FBs-GA is to obtain the Rule-Bases that produce suitable control parameter values to allow the genetic operator to show an adequate performance on the particular problem to be solved.

  FBs-GA does not handle Rule-Bases directly. Instead, it uses structures, called Fuzzy Behaviours (FBs), to represent them, which are more adequate to be treated as chromosomes by a GA. FBs consists of vectors with the linguistic values of the fuzzy rule consequent.

Since the learning technique underlying this approach only takes into account the problem to be solved (in contrast to the approach in [40, 41]), the Rule-Bases obtained shall specify adaptation strategies particularly appropriate for this problem.

4.4.2 Inputs, output, and Data Base

An instance was implemented for the case of the adaptation at individual-level of the d control parameter associated with fuzzy recombination (FR) [62], a crossover operator that was presented to work with *real-coded* GAs [38].

Let us assume that $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ $(x_i, y_i \in [a_i, b_i] \subset \Re,\ i = 1, \dots, n)$ are two real-coded chromosomes to be crossed, then FR generates an offspring, $Z = (z_1, \dots, z_n)$, where z_i is obtained from a distribution $\Phi(z_i) \in \{\phi_{x_i}, \phi_{y_i}\}$ in which ϕ_{x_i} and ϕ_{y_i} are triangular probability distributions having the following features ($x_i \leq y_i$ is assumed):

Triangular prob. dist.	Minimum value	Modal value	Maximum value
ϕ_{x_i}	$x_i - d \cdot \lvert y_i - x_i \rvert$	x_i	$x_i + d \cdot \lvert y_i - x_i \rvert$
ϕ_{y_i}	$y_i - d \cdot \lvert y_i - x_i \rvert$	y_i	$y_i + d \cdot \lvert y_i - x_i \rvert$

In particular, the range of d was constrained to the interval $[0, 1]$.

The features considered for each pair of parents, X and Y, were their index in the population, $Index(X)$, $Index(Y) \in \{1, \dots, N\}$ (N is the population size). The index of the best chromosome is N, and the one of the worst chromosome is 1 (the fitter elements have larger indexes). The set of linguistic labels associated with $Index(X)$ and $Index(Y)$ is $L = \{Low, High\}$. The meanings of these labels are depicted in Fig. 6a. The set of linguistic labels for d is $L_d = \{Low, Medium, High\}$. Their meanings are shown in Fig. 6b.

4.4.3 Rule-Bases

Each FB codes a Rule-Base having fuzzy rules whose inputs are $Index(X)$ and $Index(Y)$ and whose output is d.

The fitness function associated with the FBs should take into account the performance of FR when it is applied to the parents with the d value obtained from them. But according to what criterion should we judge this performance? One possibility that has received attention is the ability of an operator to produce children of improved fitness [59]. Clearly, this is necessary for optimisation to progress (the aim of a GA is, after all, to uncover new,

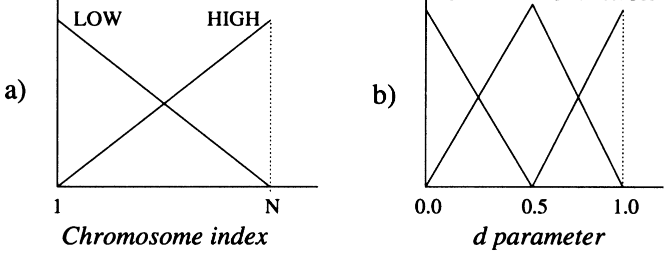

Fig. 6. Meanings for the linguistic labels considered

fitter, points in the search space). In fact, the overall performance of a GA depends upon it maintaining an acceptable level of productivity throughout the search. However, this is not enough: an efficient crossover operator should introduce the right portion of variance into the offspring population. If the variance is too large then the GA does not converge at all, whereas if it is too small then it converges prematurely [62].

Taking into account this two-fold objective, in [33], the following fitness function was proposed for each FB_i in the population of FBs-GA (minimisation is assumed):

$$Fit(FB_i) = \begin{cases} 0 & \text{if } \bar{f_O} < f(X) \le f(Y), \\ 2 - d_i & \text{if } f(X) \le \bar{f_O} < f(Y), \\ 3 & \text{if } f(X) \le f(Y) \le \bar{f_O} \ , \end{cases}$$

where $\bar{f_O}$ is the average of the fitness of the two offspring generated, $f(X)$ and $f(Y)$ are the fitness function values of the parents ($f(X) \le f(Y)$ is assumed), and d_i is the d value calculated from FB_i and $Index(X)$ and $Index(Y)$.

4.4.4 Results

An empirical study of the adaptive FR based on Coevolution with FBs was made using test functions with different difficulties from two points of view, one of performance and one of learning behaviour (based on the distributions of FBs appearing during the runs). Its results were compared with the ones from FR with fixed d values ($d = 0$, $d = 0.5$, and $d = 1$) and with the ones from the adaptation of d, at individual-level, through an FLC with a fixed Rule-Base (different ones were tried). The main conclusion was that the learning ability associated with adaptive FR allowed suitable distributions of FBs to be produced to introduce a robust operation. This means that a significant performance was obtained for each one of the test functions, which achieved their best results through Rule-Bases or fixed d values that might be different (as was observed in the experiments).

5 Future directions

Despite the recent activity and the associated progress in FAGA research [33, 52, 55], there remain many directions in which they may be improved or extended. In particular, we consider that the most promising future research areas are the following:

- Study of the aspects that may allow the behaviour of FAGAs to be improved, such as relevant inputs, the feedback between genetic operators, the FLC application frequency, and the Data Base refinement.
- Extensions for the adaptive model by Coevolution with Fuzzy Behaviours (Sect. 4.4).
- Applications of FAGAs to solve particular problems, such as constrained parameter optimisation problems and multimodal optimisation problems.
- Design of fuzzy adaptation mechanisms to control other types of evolutionary algorithms, such as distributed GAs, genetic programming, etc.
- Implementation of FAGA instances belonging to different categories of the taxonomy presented in Sect. 4.

In the following sections, we analyse these issues in deep.

5.1 Improvements for FAGAs

The future research may take into account the following issues in order to produce effective FAGAs.

Relevant inputs. Research on determining relevant input variables for the FLCs controlling GA behaviour should be explored. These variables should describe either states of the population or features of the chromosomes so that control parameters may be adapted on the basis thereof to introduce performance improvements.

Coevolution with FBs (Sect. 4.4) may be useful to discover this type of variables (for the case of adaptation at individual level). Different FBs, coding fuzzy rules with distinct number and type of inputs, may evolve together in the same population. The learning process associated with this approach shall proportionate the most significant inputs along with the fuzzy rules concerning them.

The nature of the possible input variables should be studied as well, i.e., whether they are universal to any GA or particular to a given problem. This would be useful to determine the Rule-Base learning procedure that may be applied: an Offline learning mechanism (Sect. 4.3), when fuzzy rules involve universal inputs, and an Online learning mechanism (Sect. 4.4), when they involve inputs that are particular to the problem.

Feedback between genetic operators. It may be interesting to design FLCs taking into account the action of each genetic operator in relation to the behaviour of each one of the remaining ones. The future action of an operator may be tuned depending on the repercussions of the actions of other operators (even itself). In this way, a feedback between operators must be established, allowing a suitable balance between their actions to be reached throughout the GA run.

ARGAF (Sect. C.2 in Appendix C) may be considered as a first approximation to do this, since it complements the role of the selection mechanism (either maintaining or eliminating diversity) with the role of the crossover operator (either creating or using diversity). Its significant performance shows promise in this future research front.

FLC application frequency. Usually, a fixed scheduling to fire the FLCs has been followed, i.e. every a fixed number of generations. However, the choice of the time interval between control actions is a parameter that has an influence on the final controller performance. If the controller is fired with a low time interval, the effects of previous control actions may not be achieved, whereas if the controller is fired with a high time interval, the search process may be misled by particular parameter values. A possible solution is to fire the controller when certain conditions relating to some performance measures are reached [44].

Data Base refinement. The behaviour of the FLC used by an FAGA depends on the linguistic term set and the membership functions that form the Data Base. Thus, some approaches concerning this component may be considered to improve FAGA performance. An example is the automatic Offline learning process (presented in Sect. 4.3), that may be capable of offering a suitable Data Base for the Rule-Base returned. Another possibility is to extend the definition of Fuzzy Behaviour (Sect. 4.4), with the aim of including the representation of the Data Bases along with the one of the Rule-Bases. Finally, for FAGAs that operate with Data Bases provided directly by GA experts, an attempt involves the application of a tuning process for the membership functions [13, 35].

5.2 Extensions for the adaptation model by coevolution with FBs

Different types of parameter settings were associated with genetic operators, which may be adapted by means of Coevolution with FBs. They include the following:

- *Operator probabilities*. There is a type of GAs that do not apply both crossover and mutation to the selected solutions, as in the traditional ones. Instead, a set of operators is available, each with a probability of being used, and one is selected to produce offspring. Many AGAs have been designed starting from this GA approach, which adjust the operator probabilities throughout the run [14, 59].
- *Operator parameters*. These parameters determine the way in which genetic operators work. Examples include: (1) the step size of mutation operators for real-coded GAs, which determines the strength in which real genes are mutated [45], (2) parameters associated with crossover operators for real-coded GAs, such as FR [62], BLX-α-β-γ [20], and dynamic FCB-crossovers [36], (3) the number of parents involved in multi-parent recombination operators [18], and (4) parameters associated with crossover operators for binary-coded GAs, such as n-point crossover and uniform crossover.
  The adaptation at individual-level of operator probabilities and operator parameters by Coevolution with FBs may be carried out by considering these variables as consequent of the fuzzy rules represented in the FBs. Furthermore, the appropriate features of the parents should be chosen, in the basis of which the adjustment of these variable is expressed. On the other hand, hybrid models may be built, in such a way that FBs include information for both the adaptation of operator probabilities and operator parameters. In this case, the model shall detect the operators that should be applied more frequently, along with favourable operator parameter values for them.
- *Mate selection parameters*. In mate selection mechanisms [50], chromosomes carry out the choice of mate for crossover on the basis of their own preferences (which are formulated in terms of different chromosome characteristics, such as the phenotypical distance between individuals).

Mate selection strategies may be expressed by means of FBs. In particular, given two chromosomes, an FB may induce a probability of mating depending on their characteristics. This probability determines whether or not they are crossed. Then, the process of Coevolution with FBs shall discover FBs containing mate selection strategies that encourage recombination between chromosomes that have useful information (characteristics) to exchange.
The adaptive mechanism by Coevolution with FBs may be used as well for problems where we intuit that particular features of the parents may be taken into account to allow the crossover operator behaviour to be more effective, but we do not know the precise fuzzy rules determining the relation between these features and the appropriate control actions for the operator. In this fashion, this approach allows particular knowledge about the problem to be integrated in the GA in order to improve its behaviour. Hence, this technique arises as a possible way to build hybrid GAs [14].

5.3 Applications and extensions

FAGAs may be defined to tackle particular problems such as constrained parameter optimisation problems and multimodal optimisation problems, and be extended to introduce fuzzy adaptation in other evolutionary computation models, such as distributed GAs and genetic programming. Other extensions may be directed at designing FAGA instances belonging to the categories presented in Sect. 4 that do not have a representation in the literature.

Constrained parameter optimisation problems. Different ways may be considered to apply FAGAs to these problems: (1) modify the FAGA model presented in [64] (Section C.4 in Appendix C), which deals with multiobjective optimisation problems, to solve constrained problems (the problem of satisfying a number of violated inequality constraints is clearly the multiobjective problem of optimising the associated functions until given values are reached), (2) build adaptive penalty methods based on FLCs, which may consider, as inputs to the FLCs, some measures that describe the difficulty of the constrained problem (see [46]), and (3) design adaptive genetic operators by Coevolution with FBs to deal with these problems, where the FBs take into account the degree of constraint satisfaction of the parents.

We should point out that fuzzy logic-based techniques have been used to allow evolutionary algorithms to solve these problems. In particular, in [60, 61], a evolutionary approach is presented based on the fuzzification of the constrained optimisation problems. In this method, the degrees of constraint satisfaction of the chromosomes are used as weight factors to calculate their fitness.

Multimodal optimisation problems. Given a problem with multiple solutions, a simple GA shall tend to converge to a single solution. As a result, various mechanisms have been proposed to stably maintain a diverse population throughout the search, thereby allowing GAs to identify multiple optima reliably. Many of these methods work by encouraging artificial niche formation through sharing

[24] and crowding [43], but these methods introduce one or more parameters that affect algorithm performance, parameters such as the sharing radius in fitness sharing or the crowding factor in crowding. In many problems, the uniform specification of niche size is inadequate to capture solutions of varying location and extent without also increasing the population size beyond reasonable bounds. Therefore, there remains a need to develop niching methods that stably and economically find the best niches regardless of their spacing and extent [27]. FLCs may be useful for the adaptation of parameters associated with sharing and crowding methods. Possible inputs may be: diversity measures, number of niches that are currently in the population, etc.

Fuzzy adaptive distributed GAs. The basic idea of the distributed GAs lies in the partition of the population into several subpopulations, each one of them being processed by a GA, independently from the others. Furthermore, a migration process produces a chromosome exchange between the subpopulations. Two important control parameters determine the operation of this process [11]: the migration rate, that controls how many chromosomes migrate and the migration interval, that specifies the number of generations between each migration. FLCs may be used to adapt these parameters, depending on the diversity and the convergence of the subpopulations. Furthermore, the application of Coevolution with FBs would be suitable to learn Rule-Bases that determine migration rates and migration intervals between pair of subpopulations.

Fuzzy adaptation for genetic programming (GP) [39]. GP basic distinction from GAs is the evolution of dynamic tree structures, often interpreted as programs, rather than fixed-length vectors. It would be interesting to design FLCs to control diversity and convergence in a population of genetic programs, and apply Coevolution with FBs to adapt the genetic operators that work with trees (this may be carried out by extending work appeared in [2, 29, 57]).

Design of FAGAs belonging to other categories. We may observe in Table 5 that there are groups of the taxonomy presented in Sect. 4 that have not instances in the literature. They include FAGAs with individual-level adaptations and Rule-Bases derived from an Offline learning mechanism and FAGAs that adapt parameters at population-level with an Online learning mechanism. Future research may be directed at offering FAGA instances of these categories.

Furthermore, another possible extension involves the use of FAGAs for adaptations at component-level, which associate adaptive parameters with each component of an evolving individual that determine how each component is modified during reproduction.

6 Concluding remarks

In this paper, we reviewed different aspects of FAGAs, from three points of view: design, taxonomy, and future directions.

- First, we stressed the steps for the design of the FLCs used by FAGAs. As an example, we described in depth an instance of FAGA, the fuzzy adaptive mutation probability. 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. It returns a new p_m that shall be used as an attempt to gain a better evolution quality during the next generations.
  The principal conclusions derived from the empirical study of the instance are the following:
  - The fuzzy adaptive mutation probability is the most effective technique to control p_m (with regards to the quality of the solutions returned and the speed of the search, i.e., the number of fitness function evaluations required to reach the best solutions) as compared with other non fuzzy techniques proposed in the GA literature, which were considered for the experiments.
  - The adaptation ability of this technique allows suitable p_m values to be used to produce a robust operation for test functions with different difficulties.
- Second, we categorised FAGAs according to the way in which the Rule-Bases that they use are obtained and the level where the adaptation takes place. Furthermore, we introduced the main features of instances of the different categories.
  The good performance of the approaches reviewed and the suitable results shown by the fuzzy adaptive mutation probability allow an important conclusion to be pointed out: the use of FLCs to adapt GA parameters may really improve GA performance. Clearly, this argument is based upon empirical results and is not be extensible to all classes of problems given the non free lunch theorems [66].
- Third, we discussed future directions and some challenges for FAGA research, which show that there remain many exciting research issues connected with this topic. In particular, we should emphasise the wide spectrum of possibilities that offer these algorithms to be applied and extended.

At this point, an additional aspect that should be tackled is the computational complexity of the FLCs used by FAGAs. It is important to have an idea of the trade-off between improvement in the FAGA performance versus increase in computational cost. The time required by an FLC to offer the output is relatively low. Mainly, it depends on the number of fuzzy rules and the computation of the inputs. Most FAGAs assume a small number of fuzzy rules and use inputs that are easily obtained. Moreover, FLCs are usually fired few times during the FAGA run, which means that they are not an important time wasteful element. Thus, in conclusion, the extra complexity of FAGAs becomes acceptable provided the remarkable improvements obtained by them.

Finally, we highlight the position of FAGAs in relation to an important topic of the soft computing: the integration of GAs and *fuzzy logic*. This integration has been accomplished by following two different approaches [34]: (1) the use of fuzzy logic-based techniques to improve GA behaviour, and (2) the application of GAs in optimization and search problems involving fuzzy systems [13, 35]. FAGAs are the most prolific representatives of the first approach in the literature.

Appendix A. Test suite

For the experiments (Sect. 3.3.4), we have considered six frequently used test functions:

Sphere model (f_{Sph}) [15]: $f_{\mathrm{Sph}}(\vec{x}) = \sum_{i=1}^{n} x_i^2$, where $n = 10$ and $-5.12 \leq x_i \leq 5.12$. The fitness of the optimum is $f_{\mathrm{Sph}}(x^*) = 0$. This test function is continuous, strictly convex, and unimodal.
Generalized Rosenbrock's function (f_{Ros}) [15]:
$f_{\mathrm{Ros}}(\vec{x}) = \sum_{i=1}^{n-1}(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$, where $n = 2$ and $-5.12 \leq x_i \leq 5.12$. The fitness of the optimum is $f_{\mathrm{Ros}}(x^*) = 0$. f_{Ros} is a continuous and unimodal function, with the optimum located in a steep parabolic valley with a flat bottom. This feature shall 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}) [5]:
$f_{\mathrm{Ras}}(\vec{x}) = a \cdot n + \sum_{i=1}^{n} x_i^2 - a \cdot \cos(\omega \cdot x_i)$, where $n = 10$, $a = 10$, and $\omega = 2\pi$. The fitness of its global optimum is $f_{\mathrm{Ras}}(x^*) = 0$. 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 minimisation.

Fully deceptive order-3 function (f_{Dec}) [26]. 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 6. 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.

Royal Road (f_{RR}) [23]. 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.

Table 6. Fully deceptive order-3 problem

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

Appendix B. Controlling the mutation probability

Next, we describe different adaptive mechanisms presented in the GA literature to control the mutation probability, used in Sect. 3.3.4 to compare the FAGA behaviour.

- *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. One of the most considered schedules consists in the decreasing of p_m during the GA run [6, 21]. This schedule follows the heuristic *"to protect the exploration in the initial stages and the exploitation later"*, which has been considered to design other search techniques, such as simulated annealing.
  We consider a linear function to control the decrease of p_m, following the idea presented in [6]. 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:
  $$p_m(t) = p_m^h - \frac{p_m^h - p_m^l}{T} \cdot t \quad 0 \leq t \leq T \ .$$
  The GA considered for the experiments (Sect. 3.3.4) that applies this technique is called GA-DET (Table 3).
- *Adaptive control at individual level of p_m.* In [54], 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):
  $$p_m^i = k_1 \cdot \frac{f_{\max} - f_i}{f_{\max} - \bar{f}} \text{ if } f_i \geq \bar{f}, \quad \text{and} \quad p_m^i = k_3 \text{ if } f_i < \bar{f} \ ,$$
  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 ($p_m^i = 1$). 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.
  GA-IL is the GA that uses this technique in Sect. 3.3.4 (Table 3).
- *Self-adaptive control of p_m.* 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. This gene evolves with the solution [6, 59].
  The values of p_m^i are allowed to vary from p_m^l to p_m^h. The following steps are considered to mutate 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 $[p_m^l, p_m^h]$.

The GA based on the self-adaptive control of p_m is called GA-SELF (Table 3).

Appendix C. Other FAGA approaches

This appendix is the continuation of Sect. 4.1, since it includes other FAGA approaches that use Rule-Bases derived from the knowledge of GA experts and that adapt parameters at population-level.

Furthermore, additional information about these algorithms may be found in [9, 10, 30, 65].

C.1 Approach of Xu and Vukovich

In [67, 68], the use of FLCs to control GAs is considered to solve two problems to which a standard GA may be subjected: very slow search speed and premature convergence. These problems are due to: (1) control parameters not well chosen initially for a given task, (2) parameters always being fixed even though the environment in which the GA operates may be variable, and (3) difficulties resulting from selection of other parameters such as population size and in understanding their influence, both individually and in combination, on the GA performance. FLCs were proposed to control GAs in order to: (1) choose control parameters before the GA run, (2) adjust the control parameters on-line to dynamically adapt to new situations, and (3) assist the user in accessing, designing, implementing, and validating a GA for a given task.

Experiments were carried out with an FAGA that controls p_c and p_m using two FLCs. Both of them had the same inputs: generation and population size. The Rule-Bases considered are shown in Table 7. It was claimed that part of the mechanism to create fuzzy rules to adapt p_m is that it should increasingly diminish when the GA approaches convergence to the best fitness.

The FAGA stood out as the most efficient algorithm against a standard GA in solving the TSP and other optimisation problems.

C.2 ARGAF

In [31], an fuzzy adaptive real-coded GA, called ARGAF, was proposed. Its principal features are described below.

Main ideas. ARGAF applies two different crossover operators; one with exploitation properties and another with exploration properties. A parameter, denoted as p_e, defines the frequency of application of the exploitative operator. Its value strongly influences the exploration/exploitation balance induced by the crossover operator: if p_e is low, ARGAF shall generate diversity, in this way, exploration takes effect, whereas if it is high, the current diversity shall be used to generate best elements, and so, exploitation comes into force.

Different crossover operator types were considered to build versions of ARGAF. For example, the FCB-crossover operators [37] were used as follows:
The F-crossover and S-crossover operators show exploration properties and the M-crossover operator has exploitation properties.

```
For each pair of chromosomes from a total of $p_c \cdot N$ Do
    If a random number $r \in [0,1]$ is lower than $p_e$ Then
        Generate two offspring, the result of applying two $M$-crossovers.
    Else
        Generate two offspring, the result of applying an $F$-crossover and an $S$-crossover.
    The two offspring substitute their parents in the population.
```

ARGAF uses the linear ranking selection mechanism [7]. In this selection mechanism the selective pressure is determined by the parameter $\eta_{\min} \in [0,1]$. If $\eta_{\min}$ is low, high pressure is achieved, whereas if it is high, the pressure is low.

The p_e and $\eta_{\min}$ parameters are adjusted using two FLCs. Adapting the p_e parameter, ARGAF controls the effects of crossovers, i.e., either generating diversity or using diversity, whereas adjusting the $\eta_{\min}$ parameter, it controls the effects of selection, i.e., either keeping diversity or eliminating diversity. The joint management of these parameters allows ARGAF to administer the diversity in a suitable way. For example, if useful diversity is detected by ARGAF, then it sets selection to keep diversity and crossover for using it. If the level of diversity is high and its quality is not good, then ARGAF increases the selective pressure and tries to obtain better elements

Table 7. Rule-Bases for the control of p_c and p_m, respectively

Generation	Population size			Population size		
	Small	Medium	Big	Small	Medium	Big
Short	Medium	Small	Small	Large	Medium	Small
Medium	Large	Large	Medium	Medium	Small	Very small
Long	Very large	Very large	Large	Small	Very small	Very small

by increasing exploitation by means of crossover. All these considerations are included in the Rule-Bases of the FLCs used by ARGAF. Next, we discuss the different steps in the design of these FLCs.

Inputs, outputs, and Data Base. Two diversity measures were considered as inputs. One is a genotypical diversity measure based on the Euclidean distances of the chromosomes in the population from the best one. Its definition is $GD = (\bar{d} - d_{\min})/(d_{\max} - d_{\min})$, where $\bar{d}$, $d_{\max}$, and $d_{\min}$ are the average, maximum, and minimum distances of the chromosomes in the population from the best one, respectively. The range of GD is $[0, 1]$. If GD is low, most chromosomes in the population are concentrated around the best chromosome and so convergence is achieved. If GD is high, most chromosomes are not biased towards the current best element.

The another input is a phenotypical diversity measure [40] defined as (minimisation is assumed): $PD = f_{\text{best}}/\bar{f}$, where $\bar{f}$ and f_{best} are the average and best fitness in the current population, respectively. PD belongs to the interval $[0, 1]$. If it is near to 1, convergence has been reached, whereas if they are near to 0, the population shows a high level of diversity.

GD determines the quantity of diversity in the population and PD the quality of this diversity. The linguistic label set of these inputs is $\{Low, Medium, High\}$.

The outputs are variables that control the variation on the current p_e and $\eta_{\min}$ parameters, which are kept within the range $[0.25, 0.75]$. These variables, noted as δp_e and $\delta\eta_{\min}$, represent the degree to which the current p_e and $\eta_{\min}$ values should vary, respectively. The variations shall be carried out by multiplying the δp_e and $\delta\eta_{\min}$ values, obtained by the FLCs, by the current p_e and $\eta_{\min}$ values, respectively. The action interval of δp_e and $\delta\eta_{\min}$ is $[0.5, 1.5]$ and their associated linguistic labels are $\{Small, Medium, Big\}$.

The Data Base is shown in Fig. 7. The meaning of the linguistic terms associated with GD is depicted in (a), the ones for PD in (b), and finally, the ones for δp_e and $\delta\eta_{\min}$ in (c).

Rule-Base. Table 8 shows the Rule-Bases used by the FLCs of ARGAF.

Results. Experiments on different optimisation problems of parameters with variables on continuous domains were carried out in order to study the efficacy of ARGAF. Its results were compared with the ones for other algorithms like ARGAF, but with fixed p_e and η_{min} values. Different combinations for these parameters were considered. The main conclusion was that ARGAF is a very robust GA since it adapts the p_e and η_{min} parameters to the settings that return the best results (which were different from one function to others).

C.3 Fuzzy government

In [3], it is claimed that GAs require human supervision during their routine use as practical tools for the following reasons: (1) detect the emergence of a solution, (2) tune algorithm parameters, and (3) monitor the evolution process in order to avoid undesirable behaviour such as premature convergence. It is advised as well that any attempt to develop artificial intelligence tools based on GAs should take these issues into account. The authors proposed FLCs for this task. They called fuzzy government the collection of fuzzy rules and routines in charge of controlling the evolution of the GA population.

Fuzzy government was applied to the symbolic inference of formulae problem. Genetic programming [39] was used to solve the problem along with different FLCs, which dynamically adjusted the maximum length for genotypes, acted on the mutation probability, detected the emergence of a solution, and stopped the process. The results showed that the performance of the fuzzy governed GA was almost impossible to distinguish from the performance of the same algorithm operated directly with human supervision.

C.4 FAGAs for multiobjective optimization problems

In [63], an FAGA is presented for multiobjective optimization problems. In each generation, an FLC decides what transformation of the cost components into an one-dimensional fitness function is taken.

In [64], a more complex method, called *Fuzzy Reduction* GA, is proposed. It attempts to enable a uniform approximation of the Pareto optimal solutions (those that cannot be improved with respect to any cost function without making the value of some other worse). The

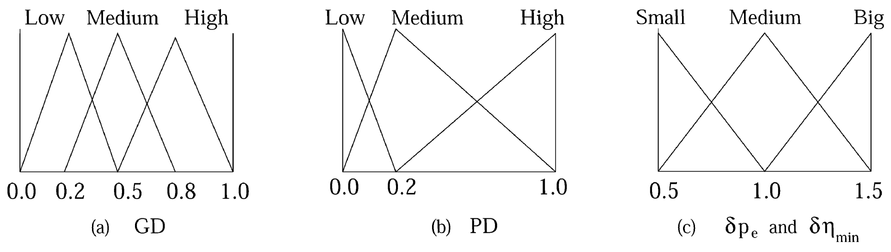

Fig. 7. Meaning of the linguistic terms associated with the inputs and the outputs

Table 8. Rule-Bases for the control of p_e and $\eta_{\min}$, respectively

GD	PD			PD		
	Low	Medium	High	Small	Medium	Large
Low	Medium	Small	Small	Small	Medium	Big
Medium	Big	Big	Medium	Small	Big	Big
High	Big	Big	Medium	Small	Small	Big

authors started by explicitly formulating desirable goals for the evolution of the population towards the target Pareto optimal solutions (which could be expressed in vague terms only). Then, they defined deviation measures of a population from these goals, which were the inputs to an FLC. Later, they fixed a set of possible actions that could serve as countermeasures to decrease the deviations. These actions are different selection mechanisms based on classical ones proposed to tackle multiobjective optimisation problems. The FLC determines activation rates for the actions. The action that should actually be taken is decided according to the activation rates found.

As an application, a timetable optimisation problem is presented where the method was used to derive cost-benefit curves for the investment into railway nets. The results showed that the fuzzy adaptive approach avoids most of the empirical shortcomings of other multiobjective GAs by the adaptive nature of the procedure.

Other models of multiobjective GA based on the use of FLCs are found in [16, 42].

C.5 Dynamic fuzzy control of GA parameter coding

In [55], an algorithm for adaptively controlling GA parameter coding using fuzzy rules is presented, which was called fuzzy GAP. It uses an intermediate mapping between the genetic strings and the search space parameters. In particular, each search parameter is specified by the following equation:

$$p_s = \left(\frac{p_g}{2^l - 1}\right) \cdot R + O \ ,$$

where p_s is the search parameter, p_g is the genetic parameter, l is the number of bits in the genetic parameter, R is a specified parameter range, and O is a specified offset. By controlling the offset and range, more accurate solutions are obtained using the same number of binary bits.

Fuzzy GAP performs a standard genetic search until the population of strings has converged. Convergence was measured by evaluating the average number of bits which differ between all the genetic strings. Each string is compared to every other string and the number of different bits are counted. If the average number of differing bits per string pair is less than a threshold, the GA has converged. After the genetic strings have converged, a new range and offset for the search parameters are determined by means of an FLC with an input that measures the distance between the center of the current range and the best solution found in the search, x_b:

$$d(x_b, O, R) = \left| 2 \cdot \left(\frac{x_b - O}{R}\right) - 1 \right| \ .$$

This measure lies in $[0, 1]$. It is 0 when the best solution is exactly in the center of the range, and 1 when the best solution is either at the lower limit or upper limit of the range. According to this measure, the authors developed different heuristic rules:

- If the best solution is near the center of the range, it makes sense that the range should be reduced in size. The best solution is the center of the range indicates that previous range adjustments were correct and the true solution is near the center.
- If the best solution is near one of the limits, the best solution is moving and the search space should be adjusted to include more of the space about the best solution. Thus, increasing the size and centering the range is reasonable.

The use of fuzzy rules allowed easy and straightforward implementation of this type of heuristic rules. After applying the FLC, the GA is executed again with the new values for the range and offset.

The performance of fuzzy GAP on a hydraulic brake emulator parameter identification problem was investigated. It was shown to be more reliable than other dynamic coding algorithms (such as the dynamic parameter encoding algorithm [51]), providing more accurate solutions in fewer generations.

C.6 Fuzzy cultural algorithms

Cultural algorithms (CAs) [48] are dual inheritance systems that consist of a social population and a belief space. The problem solving experience of individuals selected from the population space by an acceptance function is used to generate problem solving knowledge that resides in the belief space. This knowledge can be viewed as a set of beacons that can control the evolution of the population component by means of an influence function. The influence function can use the knowledge in the belief space to modify any aspect of the population component. Various evolutionary models have been used for the population component of CAs, including GAs, genetic programming, evolution strategies, and evolutionary programming.

In [49], a fuzzy approach to CAs is presented in which an FLC regulates the amount of information to be transferred to the belief space used by the CA over time. In particular, the FLC determines the number of individuals which shall impact the current beliefs. Its inputs are the individual success ratio (ratio of the number of successes to the total number of mutations) and the current generation. The basic intuition used to design the fuzzy rules was that if the current generation is early on in the evolution process and the success ratio is low then accept a medium number (around top 30%) of individuals from the population. If the current generation is early on and the success ratio is high, then accept a larger number (around 40%). If the current generation is near the end of the evolution process and the success ratio is low, then accept a smaller number (around top 20%). And, if the current generation is near the end and the success ratio is high, then accept a medium number.

A comparison was made between the fuzzy version of a CA (that used evolutionary programming as the population component) and its non fuzzy version on 34 optimisation functions. The conclusions were: (1) the fuzzy interface between the population and belief space outperformed the non fuzzy version in general, and (2) the use of fuzzy acceptance function significantly improved the success ratio and reduced CPU time.

References

1. **Angeline PJ** (1995) Adaptive and self-adaptive evolutionary computations. In: Palaniswami M, Attikiouzel Y, Markc R, Fogel D, Fukuda T (eds), Computational Intelligence: A Dynamic Systems Perspective, pp. 152–163. Piscataway, NJ: IEEE Press
2. **Angeline PJ** (1996) Two self-adaptive crossover operators for genetic programming. In: Angeline PJ, Kinnear JE, Jr. (eds), Advances in Genetic Programming 2, pp. 89–109. Cambridge, MA: MIT Press
3. **Arnone S, Dell'Orto M, Tettamanzi A** (1994) Toward a fuzzy government of genetic populations. In: Proc. of the 6th IEEE Conference on Tools with Artificial Intelligence, pp. 585–591. Los Alamitos, CA: IEEE Computer Society Press
4. **Bäck, T** (1992) The interaction of mutation rate, selection, and self-adaptation within genetic algorithm. In: Männer R, Manderick B (eds), Parallel Problem Solving from Nature 2, pp. 85–94. Amsterdam: Elsevier Science Publishers
5. **Bäck T** (1992) Self-adaptation in genetic algorithms. In: Varela FJ, Bourgine P, (eds), Proc of the First European Conference on Artificial Life, pp. 263–271. Cambridge, MA: The MIT Press
6. **Bäck T, Schütz M** (1996) Intelligent mutation rate control in canonical genetic algorithms. In: Ras ZW, Michalewicz M (eds), Foundation of Intelligent Systems 9th Int Symposium, pp. 158–167. Berlin: Springer
7. **Baker JE** (1985) Adaptive selection methods for genetic algorithms. In: Proc First Int Conf on Genetic Algorithms, pp. 101–111. Hillsdale, MA: L. Erlbaum Associates
8. **Baker JE** (1987) Reducing bias and inefficiency in the selection algorithm. In: Grefenstette JJ (ed.), Proc of the Second Int Conf on Genetic Algorithms and their Applications, pp. 14–21. Hillsdale, NJ: Lawrence Erlbaum
9. **Bastian A, Hayachi I** (1996) A proposal for knowledge-based systems using fuzzy rules and genetic algorithms, Japanese J Fuzzy Theory and Systems **8**(6): 895–907
10. **Bergmann A, Burgard W, Hemker A** (1994) Adjusting parameters of genetic algorithms by fuzzy control rules. In: Becks K-H, Perret-Gallix D (eds), New Computing Techniques in Physics Research III, pp. 235–240. Singapore: World Scientific Press
11. **Cantú-Paz E** (2000) Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers
12. **Cordón O, Herrera F, Peregrín A** (1997) Applicability of the fuzzy operators in the design of fuzzy logic controllers, Fuzzy Sets and Systems, **86**(1): 15–41
13. **Cordón O, Herrera F, Hoffmann F, Magdalena L** (2001) Genetic fuzzy systems. Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific
14. **Davis L** (1991) Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold.
15. **De Jong KA** (1975) An analysis of the behaviour of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Ann Arbor. Dissertation Abstracts International, **36**(10), 5140B (University Microfilms No 76-9381)
16. **Dozier GV, McCullough S, Homaifar A, Moore L** (1998) Multiobjective evolutionary path planning via fuzzy tournament selection. In: IEEE International Conference on Evolutionary Computation (ICEC'98), pp. 684–689. Piscataway: IEEE Press
17. **Driankow D, Hellendoorn H, Reinfrank M** (1993) An Introduction to Fuzzy Control. Berlin: Springer-Verlag
18. **Eiben AE** (1997) Multi-parent recombination. In: Bäck T, Fogel D, Michalewicz Z (eds), Handbook of Evolutionary Algorithms, pp. 25–33. IOP Publishing Ltd. and Oxford University Press
19. **Eiben AE, Hinterding R, Michalewicz Z** (1999) Parameter control in evolutionary algorithms, IEEE Trans Evolut Comput, **3**(2): 124–141
20. **Eshelman LJ, Mathias KE, Schaffer JD** (1997) Crossover operator biases: exploiting the population distribution. In: Bäck T (ed.), Proc of the Seventh Int Conf on Genetic Algorithms, pp. 354–361. San Mateo: Morgan Kaufmann
21. **Fogarty TC** (1989) Varying the probability of mutation in the genetic algorithm. In: Schaffer JD (ed.), Proc of the Third Int Conf on Genetic Algorithms, pp. 104–109. San Mateo: Morgan Kaufmann
22. **Fogel DB, Fogel GB, Ohkura K** (2001) Multiple-vector self-adaptation in evolutionary algorithms, BioSystems **61**: 155–162
23. **Forrest S, Mitchell M** (1993) Relative building block fitness and the building block hypothesis. In: Whitley LD (ed.), Foundations of Genetic Algorithms-2, pp. 109–126. San Mateo: Morgan Kaufmann
24. **Goldberg DE, Richarson JJ** (1987) Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette JJ (ed.), Proc of the Second Int Conf on Genetic Algorithms and their Applications, pp. 28–31. Hillsdale, NJ: Lawrence Erlbaum
25. **Goldberg DE** (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley
26. **Goldberg DE, Korb B, Deb K** (1989) Messy genetic algorithms: motivation, analysis, and first results, Complex Systems **3**: 493–530
27. **Goldberg DE, Wang L** (1997) Adaptive niching via coevolutionary sharing. In: Quagliarella et al. (eds), Genetic Algorithms in Engineering and Computer Science, pp. 21–38. John Wiley and Sons Ltd
28. **Grefenstette JJ** (1986) Optimization of control parameters for genetic algorithms, IEEE Trans Systems, Man, and Cybernetics **16**: 122–128
29. **Iba H, de Garis H** (1996) Extending genetic programming with recombinative guidance. In: Angeline PJ, Kinnear JE, Jr. (eds), Advances in Genetic Programming 2, pp. 69–88. Cambridge, MA: MIT Press
30. **Herrera F, Herrera-Viedma E, Lozano M, Verdegay JL** (1994) Fuzzy tools to improve genetic algorithms. In: Proc of the European Congress on Intelligent Techniques and Soft Computing, pp. 1532–1539
31. **Herrera F, Lozano M** (1996) Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In: Herrera F, Verdegay JL (eds), Genetic Algorithms and Soft Computing, pp. 95–125. Physica-Verlag
32. **Herrera F, Lozano M** (2000) Adaptive control of the mutation probability by fuzzy logic controllers. In: Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo JJ, Schwefel H-P (eds), Parallel Problem Solving from Nature VI, pp. 335–344. Berlin: Springer
33. **Herrera F, Lozano M** (2001) Adaptive genetic operators based on coevolution with fuzzy behaviours, IEEE Trans on Evolut Comput **5**(2): 1–18
34. **Herrera F, Lozano M, Verdegay JL** (1995) Tackling fuzzy genetic algorithms. In: Winter G, Periaux J, Galan M, Cuesta P (eds), Genetic Algorithms in Engineering and Computer Science, pp. 167–189, John Wiley and Sons
35. **Herrera F, Lozano M, Verdegay JL** (1995) Tuning fuzzy logic controllers by genetic algorithms, Int J Approx Reasoning, **12**: 299–315
36. **Herrera F, Lozano M, Verdegay JL** (1996) Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convengence of real-coded genetic algorithms, Int J Intelligent Systems **11**: 1013–1041
37. **Herrera F, Lozano M, Verdegay JL** (1997) Fuzzy connectives based crossover operators to model genetic algorithms population diversity, Fuzzy Sets and Systems **92**(1): 21–30
38. **Herrera F, Lozano M, Verdegay JL** (1998) Tackling real-coded genetic algorithms: operators and tools for the behavioural analysis, Artificial Intelligence Reviews **12**(4): 265–319

39. **Koza JR** (1992) Genetic Programing: on the Programming of Computers by Means of Natural Selection. Cambridge: The MIT press
40. **Lee MA, Takagi H** (1993) Dynamic control of genetic algorithms using fuzzy logic techniques. In: Forrest S (ed.), Proc of the Fifth Int Conf on Genetic Algorithms, pp. 76–83. San Mateo: Morgan Kaufmann
41. **Lee MA, Takagi H** (1994) A framework for studying the effects of dynamic crossover, mutation, and population sizing in genetic algorithms. In: Furuhashi T (ed.), Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms, pp. 111–126. Lecture Notes in Computer Science 1011, Berlin: Springer-Verlag
42. **Lee MA, Esbensen, H** (1997) Fuzzy/multiobjective genetic systems for intelligent systems design tools and components. In: Pedrycz W (ed.), Fuzzy Evolutionary Computation, pp. 57–80. Boston: Kluwer Academic Publishers
43. **Mahfoud SW** (1992) Crowding and preselection revised. In: Männer R, Manderick B (eds), Parallel Problem Solving from Nature 2, pp. 27–36. Amsterdam: Elsevier Science Publishers
44. **Matousek R, Osmera P, Roupec J** (2000) GA with fuzzy inference system. In: Proc 2000 IEEE Int Conf on Evolutionary Computation, pp. 646–651. Piscataway, NJ: IEEE Press
45. **Michalewicz Z** (1992) Genetic Algorithms + Data Structures = Evolution Programs, New York: Springer-Verlag
46. **Michalewicz Z, Schoenauer M** (1996) Evolutionary algorithms for constrained parameter optimization problems, Evolut Comput **4**(1): 1–32
47. **Oyama A, Obayashi S, Nakamura T** (2001) Real-coded adaptive range genetic algorithm applied to transonic wing optimization, Appl Soft Compting **16**: 1–9
48. **Reynolds RG** (1994) An introduction to cultural algorithms. In: Sebald AV, Fogel LJ (eds), Proc of the 3rd Annual Conference on Evolutionary Programming, pp. 131–139. River Edge, NJ: World Scientific
49. **Reynolds RG, Chung C-J** (1997) Regulating the amount of information used for self-adaptation in cultural algorithms. In: Bäck T (ed.), Proc of the Seventh Int Conf on Genetic Algorithms, pp. 401–408. San Francisco: Morgan Kaufmann Publishers
50. **Ronald E** (1993) When selection meets seduction. In: Forrest S (ed.), Proc of the Fifth Int Conf on Genetic Algorithms, pp. 167–173. San Mateo: Morgan Kaufmann
51. **Schraudolph NN, Belew RK** (1992) Dynamic parameter encoding for genetic algorithms, Machine Learning **9**: 9–21
52. **Shi Y, Eberhart R, Chen Y** (1999) Implementation of evolutionary fuzzy systems, IEEE Trans Fuzzy Systems **7**(2): 109–119
53. **Smith JE, Fogarty TC** (1997) Operator and parameter adaptation in genetic algorithms. Soft Computing **1**(2): 81–87
54. **Srinivas M, Patnaik LM** (1994) Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Trans Systems, Man, and Cybernetics **24**(4): 656–667
55. **Streifel RJ, Marks II RJ, Reed R, Choi JJ, Healy M** (1999) Dynamic fuzzy control of genetic algorithm parameter coding, IEEE Trans Systems, Man, and Cybernetics – Part B: Cybernetics **29**(3): 426–433
56. **Subbu R, Sanderson AC, Bonissone PP** (1998) Fuzzy logic controlled genetic algorithms versus tuned genetic algorithms: an agile manufacturing application. In: Proc of the IEEE SIC/CIRA/ISAS '98 Conference
57. **Teller A** (1996) Evolving programmers: the co-evolution of intelligent recombination operators. In: Angeline PJ, Kinnear JE, Jr. (eds), Advances in Genetic Programming 2, pp. 45–68. Cambridge, MA: MIT Press
58. **Tettamanzi AG** (1995) Evolutionary algorithms and fuzzy logic: a two-way integration. In: 2nd Joint Conference on Information Sciences, pp. 464–467, Wrightsville Beach, NC
59. **Tuson AL, Ross P** (1998) Adapting operator settings in genetic algorithms. Evolut Comput **6**(2): 161–184
60. **Van Le T** (1995) A fuzzy evolutionary approach to solving constraint problems. In: Proc 1995 IEEE Int Conf on Evolutionary Computation, pp. 317–319. Piscataway, NJ: IEEE Press
61. **Van Le T** (1996) A fuzzy evolutionary approach to constrained optimisation problems. In: Proc 1996 IEEE Int Conf on Evolutionary Computation, pp. 274–278. Piscataway, NJ: IEEE Press
62. **Voigt HM, Mühlenbein H, Cvetkovic D** (1995) Fuzzy recombination for the breeder genetic algorithm. In: Eshelman L (ed.), Proc of the Sixth Int Conf on Genetic Algorithms, pp. 104–111. San Francisco: Morgan Kaufmann Publishers
63. **Voget S** (1996) Multiobjective optimization with genetic algorithms and fuzzy-control. In: Proc of the Fourth European Congress on Intelligent Techniques and Soft Computing, pp. 391–394
64. **Voget S, Kolonko M** (1998) Multidimensional optimization with a fuzzy genetic algorithm, J Heuristic **4**(3): 221–244
65. **Wang PY, Wang GS, Hu ZG** (1997) Speeding up the search process of genetic algorithm by fuzzy logic. In: Proc of the European Congress on Intelligent Techniques and Soft Computing, pp. 665–671
66. **Wolpert DH, Macready WG** (1997) No free lunch theorems for optimization, IEEE Trans Evolut Comput **1**(1): 67–82
67. **Xu HY, Vukovich G** (1993) A fuzzy genetic algorithm with effective search and optimization. In: Proc of 1993 International Joint Conference on Neural Networks, pp. 2967–2970
68. **Xu HY, Vukovich G, Ichikawa Y, Ishii Y** (1994) Fuzzy evolutionary algorithms and automatic robot trajectory generation. In: Michalewicz Z, Schaffer JD, Schwefel H-P, Fogel DB, Kitano H (eds), Proceeding of the First IEEE International Conference on Evolutionary Computation, pp. 595–600. Piscataway, NJ: IEEE Press
69. **Zeng X, Rabenasolo B** (1997) A fuzzy logic based design for adaptive genetic algorithms. In: Proc of the European Congress on Intelligent Techniques and Soft Computing, pp. 660–664