Outline of a new evolutionary algorithm for fuzzy systems learning

D T Pham and M Castellani

Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 2002 216: 557
DOI: 10.1243/0954406021525340

Outline of a new evolutionary algorithm for fuzzy systems learning

D T Pham* and **M Castellani**
Manufacturing Engineering Centre, School of Engineering, University of Wales, Cardiff, Wales, UK

Abstract: This paper describes a new evolutionary algorithm for the automatic generation of the knowledge base for fuzzy logic systems. In common with other evolutionary approaches, the approach adopted is to treat the problem of knowledge base generation as that of searching for a solution of an acceptable quality by applying genetic operators to a population of potential solutions. The algorithm presented dynamically adjusts the focus of the genetic search by dividing the population into three subgroups, each concerned with a different level of knowledge base optimization. The algorithm also includes a new adaptive selection routine that aims to keep the selection pressure constant throughout the learning phase.

Keywords: evolutionary algorithms, search, fuzzy logic, fuzzy learning classifier systems

NOTATION

A_{ij}, A'_{ij}	MF of fuzzy term j of universe of discourse i
C, C'	children in a GA population
delta	distance
$f(i)$	fitness value of individual i
$f'(i)$	normalized fitness value of individual i
I_{ij}	MF of fuzzy term j of input universe of discourse i
$m(i)$	mating chance of individual i
p_{x_A}	xth anchor point of MF A
P, P'	parents in a GA population
rand	random real number
μ_f	average fitness value of population

Abbreviations

BP	back-propagation
EA	evolutionary algorithm
EP	evolutionary programming
EVS	evolution strategy
FL	fuzzy logic
FNN	fuzzy neural network
GA	genetic algorithm
KB	knowledge base
LCS	learning classifier system
MF	membership function
MRAC	model reference adaptive controller
RB	rule base
RBF	radial basis function

The MS was received on 12 April 2001 and was accepted after revision for publication on 16 January 2002.
Corresponding author: Manufacturing Engineering Centre, School of Engineering, University of Wales, Cardiff, PO Box 925, Newport Road, Cardiff CF24 OYF, Wales, UK.

1 INTRODUCTION

The use of evolutionary algorithms (EAs) [**1**] to generate fuzzy logic (FL) [**2**] systems can overcome the drawbacks of local search techniques as well as allowing the simultaneous optimization of both the structure and parameters of the solution. The amount of problem domain expertise required for the implementation of EAs is small and in some cases is limited to the definition of the optimization objectives for the implementation of the fitness evaluation function. Moreover, EAs do not require properties such as the differentiability of the logic operators, unlike other optimization algorithms including the error back-propagation (BP) rule [**3**].

The main difficulty in the evolutionary design of FL systems is due to the complexity of the learning task, which requires the optimization of a large number of mutually related parameters and variables. The encoding of the fuzzy knowledge base (KB) is not straightforward and is not suited to traditional string-like implementations. Moreover, the simultaneous learning of the rule base (RB) and the fuzzy membership functions (MFs) requires the definition of two different but concurrent learning strategies. The problem has

analogies with artificial neural network (NN) learning [**4**, **5**], where the NN structure and parameters have both to be optimized. Indeed, in the field of evolutionary fuzzy neural network (FNN) learning, the two problems are identical.

For these reasons, the implementation of evolutionary FL systems has so far relied on *ad hoc* solutions, often driven by the particular problem domain. This paper details a new EA for the automatic generation of KBs for FL systems. The FL system optimization problem was approached in a generic manner to facilitate the application of the results to different problems.

The proposed algorithm dynamically optimizes the search efforts allocated for the RB and the MF optimization by dividing the solution population into three subgroups, each concerned with a different level of knowledge base optimization. Moreover, the evolved KB is expressed in a transparent format that facilitates the understanding of the control policy.

The paper is organized as follows: Section 2 gives a brief introduction to EAs and presents a review of the relevant literature in the evolutionary generation of the KB of FL systems. Section 3 describes the proposed EA. Section 4 concludes the paper and proposes areas for further investigation.

2 EVOLUTIONARY ALGORITHMS FOR TRAINING FUZZY SYSTEMS

2.1 Evolutionary algorithms

The design and optimization of a fuzzy controller is essentially a search problem, where the solution space is represented by all the possible structures and parameters defining the system. EAs are stochastic search algorithms that aim to find an acceptable solution when time or computational requirements make it impractical to find the best one.

EAs are modelled on Darwin's theory of natural evolution. This stipulates that a species improves its adaptation to the environment by means of a selection mechanism that favours the reproduction of those individuals of highest fitness. The population is made to evolve until a stopping criterion is met. At the end of the process, the best exemplar is chosen as the solution to the problem.

In EAs, the adaptation of an individual to the environment is defined by its ability to perform the required task. A problem-specific *fitness function* is used for the quality assessment of a candidate solution. The population is driven towards the optimal point(s) of the search space by means of stochastic search operators inspired by the biological mechanisms of genetic *selection*, *mutation* and *recombination*. Problem-specific operators are often used to speed up the search process.

Historically, EAs originated in the mid-1960s with two parallel directions of work leading to the fields of evolution strategies (EVSs) [**6**] and evolutionary programming (EP) [**7**]. However, it was only ten years later that they gained popularity following the creation of genetic algorithms (GAs) [**8**, **9**]. EVSs, EP and GAs represent different metaphors of biological evolution with different representations of the candidate solutions and different genetic manipulation operators.

In the last decade, research developments in each field and the mutual exchange of ideas blurred the boundaries between the three main branches of EAs. The reader is referred to the following sources for further material:

(a) EVSs: references [**1**], [**10**], [**11**], [**12**], [**13**] and [**14**];
(b) EP: references [**1**], [**12**], [**13**], [**14**] and [**15**];
(c) GAs: references [**1**], [**12**], [**13**], [**14**], [**16**], [**17**], [**18**], [**19**] and [**20**].

2.2 EAs for membership functions learning

The first application of EAs to FL system optimization was described by Karr *et al.* [**21**] and concerned the training of a fuzzy spacecraft controller for autonomous rendezvous operations. The learning algorithm was a standard GA used to optimize the fuzzy MFs of a Mamdani-type FL system [**22**, **23**]. The learning controller was successively improved by introducing on-line adaptation of the MF parameters. Applications to the control of the pH of a laboratory acid–base solution and the cart–pole balancing problem were reported respectively in references [**24**], [**25**] and [**26**], [**27**].

Several other researchers have investigated the use of EAs for MF learning. Examples can be found in reference [**28**] (process modelling and data classification), reference [**29**], where a modified GA was designed for tuning MFs, input scaling factors and output gain of an FNN PID (proportional-integral-derivative) controller [**20**], and in references [**30**], [**31**] and [**32**], where different two-step evolutionary procedures were proposed for the automatic generation of the architecture and MFs of hierarchical FL systems (data modelling applications).

The evolutionary adjustment of fuzzy MFs is suitable for problems such as FL system optimization (as in Karr's first study) or on-line system adaptation (as in his subsequent experiments). The gross behaviour of the fuzzy mapping is in any case determined by the fuzzy rule base (RB). In the optimization case, this stresses the importance of the design of the initial system. In the case of on-line adaptation of the FL system response, the success of the evolutionary method is related to the magnitude of the changes in the external environment. The amount of adaptation is in fact limited by the static nature of the fuzzy RB. Moreover, since EAs are

relatively slow search techniques, the evolutionary method is not suitable for rapidly changing dynamic systems.

2.3 EAs for rule base generation

Shortly after Karr's first paper on FL system optimization via genetic modification of the MFs, other authors focused on evolutionary RB acquisition as a method for FL machine learning. There are two ways of evolving the RB of fuzzy production systems that have originated from earlier studies in the broader field of learning classifier systems (LCSs). The *Pittsburgh* or 'Pitt' approach, introduced by the work of Smith [**33**] at the University of Pittsburgh, builds the final RB from a population of rule sets. The *Michigan* approach, initiated by Holland and Reitman [**34**] at the University of Michigan, generates the optimal rule set from a population of rules.

The first example of a fuzzy LCS was presented in reference [**35**], where the Michigan approach was applied to the fuzzy modelling of a simulated plant. Machado and Freitas da Rocha [**36**] proposed a combination of GA-based deductive learning and Hebbian inductive learning for the automatic generation of a fuzzy neural network (FNN) classifier. The fuzzy RB was determined by the network structure and was produced through genetic evolution using the Michigan approach. Bonarini [**37**, **38**] implemented a Michigan-type fuzzy LCS for the control of an autonomous mobile robot. Rule mutation was the only genetic manipulation operator used. For this reason, the learning algorithm was conceptually similar to EP.

Pham and Karaboga [**39**, **40**] and Thrift [**41**] proposed the first evolutionary fuzzy systems generated via the Pitt method. In the first case, the implementation focused on the automatic generation of the fuzzy relation matrix of a PI-type FL controller. In the second case, the algorithm evolved a Mamdani-type controller using a standard GA.

A similar approach was followed by Kropp and Baitinger [**42**] and Hwang and Thompson [**43**], with the latter study involving the implementation of a fuzzy model reference adaptive controller (MRAC). Feldman [**44**] coded each rule as a binary string and encoded each solution using fixed-length strings built by joining a predefined number of possible rules, this number being fixed prior to training. The usual GA operators were adopted for evolving the population.

The use of fixed-length strings to encode each possible combination of the rule conditions is likely to produce a non-minimal RB. For this purpose, Buhusi [**45**] used a variable-length chromosome to encode a radial basis function (RBF) representation of a Takagi–Sugeno-type [**46**] fuzzy classifier. The length of the string depended on the number of rules, i.e. RBF basis units, represented.

The generation of the fuzzy RB allows the determination of the general behaviour of the FL system. An appropriate choice of the fuzzy MFs is in any case important for correct definition of the fuzzy mapping. The definition of a finer fuzzy partitioning for the input and the output spaces can improve the mapping accuracy of the fuzzy system. On the other hand, the size of the RB exponentially increases with the number of input variables, thus severely affecting the transparency of the system and its processing times. The optimal solution in terms of mapping accuracy and structural simplicity is therefore the automatic acquisition of both RB and MFs.

2.4 EAs for concurrent membership functions and rule base learning

One of the first studies into automatic optimization of both fuzzy RB and MFs is reported in reference [**47**]. The paper proposed a Michigan-type fuzzy LCS where each solution was encoded using a set of values defining the central point of the input and the output MFs. The population size determined the size of the final RB and its optimal value depended on the width of the support of the MFs, which was a system parameter.

Lee and Takagi [**48**] proposed a combination of FL and evolutionary techniques for the dynamic control of GA parameters. The resulting adaptive GA was applied to the automatic generation of the KB of a fuzzy controller for the inverted pendulum problem. The overall system was composed of two GA-generated FL controllers, the first used to control the inverted pendulum and the second to adjust the search parameters dynamically during the genetic optimization of the first. The main drawback of this algorithm lay in the complexity of the overall learning process that required the running of a meta-GA on a population of GA controllers.

Kinzel *et al.* [**49**] used a two-step procedure for the automatic optimization of the RB and the MFs of an FL controller. An EA was first run to define an initial coarse policy via optimization of the fuzzy RB. After the initial step, the system was finally tuned via GA optimization of the fuzzy MFs. Because of the two-step procedure adopted, the algorithm was prone to converging to suboptimal solutions. Two cascaded GAs were also proposed by Heider and Drabe [**50**] to evolve the RB and MFs of FL systems.

Buckles *et al.* [**51**] suggested a Pitt-based fuzzy LCS where the RB and the fuzzy MFs were simultaneously evolved. Data categories were clustered using ellipsoidal MFs. Variable-length genotypes were used to encode each solution by concatenating the encodings of the singular ellipses. Cooper and Vidal [**52**] proposed a similar algorithm for the simultaneous optimization of the RB and MFs of FL systems.

Liska and Melsheimer [**53**] used a three-chromosome genotype for the encoding of FL systems. The algorithm included a final gradient-descent optimization stage for the fine tuning of the MFs. Ng and co-workers [**54**, **55**] employed a modified GA for the optimization of FL controllers. In this implementation, the number of MFs and the size of the RB were fixed and not subjected to evolution. A similar algorithm was also designed by Homaifar and McCormick [**56**] and Seng *et al.* [**57**]. The latter worked on a population of Takagi–Sugeno-type FL systems that were implemented using RBF NNs.

Chiang *et al.* [**58**] developed a reinforcement learning architecture for the genetic evolution of FL controllers. The system described in reference [**59**] provided predictive reinforcement of the goodness of the proposed actions. A stochastic action modifier changed the output of the controller by an amount inversely proportional to the reinforcement signal. The learning controller was applied to the cart–pole system. Even though the combination of genetic and reinforcement learning techniques makes the application interesting, it should be pointed out that the extent of the learning was limited to the location of the output fuzzy singletons. The input space partition was fixed and the number of fuzzy rules was set to cover the exhaustive combination of the input conditions.

Shi *et al.* [**60**] designed an adaptive EA for fuzzy LCSs. The crossover and mutation rates were tuned on-line during the genetic search, the adaptation law being defined by a set of heuristic fuzzy rules. Solutions are encoded into integer-based fixed-length strings defining the fuzzy rules and the shapes and ranges of the MFs. Each rule is defined by the labels of the condition and the action terms, with a 'don't care' allele to mark irrelevant terms. A specific gene determines the maximum number of rules in the RB and only rules up to that number will be decoded into the phenotype. The ability genetically to determine the shape of each single MF and the fuzzy adaptation of the crossover and mutation rates were the interesting features of this algorithm. On the other hand, it should be noted that the RB encoding scheme is likely to generate a sizeable amount of unused genetic material. This unwanted information will still be processed by the EA, thus taking computational resources from the genetic search. Moreover, pre-setting the size of the RB is a risky operation, as it may produce a suboptimal solution.

Pre-setting some of the parameters defining the final solution and adopting a 'divide and conquer' approach, where learning is carried out at different stages, are common solutions to the problem of evolutionary FL system generation. As remarked above, these approaches can lead to suboptimal solutions. Moreover, the setting of the parameters may require a certain amount of expertise and a lengthy trial-and-error phase. On the other hand, approaches tending to enhance the power of the learning algorithm to deal with the complexity of FL learning (e.g. reference [**48**]) often result in very complex systems with long running times and poorly understood dynamics.

3 PROPOSED EVOLUTIONARY ALGORITHM

3.1 Overview

In its present configuration, the proposed EA is designed for the generation of Mamdani-type FL systems through simultaneous evolution of the RB and MFs. The kind of MFs used for partitioning the input and output spaces is fixed prior to the learning process and is not subjected to evolution. As the significance of such a parameter is limited, it has not been included in the learning process.

To increase KB transparency, no rule confidence factors are used in the fuzzy inferencing. For the sake of generality, their representation was included in the genome of the rules. Their value is set to one (i.e. full confidence, no action scaling) and is left unchanged by the evolutionary procedure.

The adopted evolution scheme is close to the Pitt approach, where a population of FL systems is the object of the optimization process. For the Michigan approach to be applied to the simultaneous optimization of the RB and the MFs, a distinct set of condition and action terms must be associated with each fuzzy rule. This situation should be avoided as it affects the transparency of the fuzzy KB and is likely to generate a non-minimal RB. Moreover, in control system and in dynamic system modelling, the overall performance is often determined by the cumulative application of several different actions. Frequently, the set of activated rules includes a substantial part of the fuzzy response surface, and alternative and competing strategies may exist at the KB level. Therefore, the Pitt approach aiming at the evolution of complete fuzzy systems seems to be more appropriate.

The algorithm uses the generational replacement reproduction scheme [**17**], a new selection operator and a set of crossover and mutation procedures dealing with different elements of the fuzzy KB. For this purpose, the population is divided into three competing subgroups to which different operators are applied. A specific integer-valued gene marks the species of each individual.

The genetic operators acting on the first subpopulation work at the level of input and output fuzzy partitions. Crossover generates two new individuals by mixing the MFs of the two parents for each variable. Each parent transmits its RB to one of the offspring. Random chromosomic mutations can introduce new fuzzy terms, delete existing ones or change the parameters defining the location and shape of an MF. Whenever genetic manipulations modify the set of MFs

over which an RB is defined, a 'repair' algorithm translates the old rule conditions and actions into the new fuzzy terms. The subpopulation of fuzzy systems undergoing this set of operations is called *species_1*.

The operators manipulating the second subpopulation search for the optimal RB. This group of solutions is named *species_2* and can be thought of as the population in a Pitt-type fuzzy LCS. Genetic crossover creates two individuals by exchanging sets of rules between the two parents. Each of the offspring inherits the input and the output space partitions from one of its parents. To accommodate the new partitions, the conditions and actions of the swapped rules are translated into new linguistic terms. The mutation operator randomly changes the action of a fuzzy rule.

The operators acting on the third subpopulation deal with all the components of the fuzzy KB. Genetic recombination swaps all the MFs and the rules contained in a randomly selected portion of the input and output spaces. Mutation can take any of the forms defined for the modification of *species_1* and *species_2* genotypes. This third group of individuals is referred to as *species_3*.

Each of the subpopulations is used to perform a different search in the space of possible solutions; *species_1* is mainly concerned with fine tuning the fuzzy response, while the other two species are used for increasingly more disruptive search approaches. Information is naturally exchanged between different species through genetic recombination, which is not forbidden or strictly regulated as in normal niching techniques [**61**]. When two individuals of different kinds are mated, the fitter solution dominates the other, determining the species of the offspring and the type of crossover applied. This mechanism adds a bias towards the most successful search approach, and its action augments the effects of the selection pressure. Competition for survival is increased among the whole population of individuals, a feature that also differs from many niching approaches.

A species mutation operator prevents one type of individual from taking over the entire population. Its action has the double purpose of allowing the resurgence of temporarily suppressed species and spreading useful genetic material over the three population groups. The possibility of reinstating declining species allows the adaptive adjustment of the search strategy. As an example, at the beginning of the optimization process it may be preferable to focus the search efforts at the RB level for a quick broad-brush definition of the behaviour of the system. At a later stage, the fine tuning of the response may require more computational resources to be allocated for MF optimization.

The possibility of rapidly diffusing advantageous genetic material to other population groups allows a quick differentiation of the search levels around promising solutions. Because of this capability for quickly spreading 'good' chromosomes, the proposed adaptive mechanism is preferred to standard migration techniques [**16**].

It is important to point out that the separation of the population into three subgroups is aimed at the dynamic adjustment of the focus of the genetic search towards different KB elements. In contrast with other niching techniques, the division of the population into species is not aimed at sustaining population diversity, which is instead pursued through the conventional balance of selection pressure and genetic mutations. Because of this fundamental difference, the proposed algorithm cannot be regarded as adopting a conventional niching technique.

The possibility of shifting the search objectives according to the size of the three subpopulations could also be used to define more limited search strategies. In an on-line application of Karr's type of self-tuning controller, adaptive MF tuning could be achieved by the sole use of *species_1* individuals. Similarly, if the input and output space partitions were straightforward, only *species_2* individuals would be initialized. A progressive adaptation mechanism could also be defined, where the KB search level is determined by the magnitude of the response error. The range of applicability of the proposed algorithm is therefore wide and covers both on-line and off-line usage.

The following subsections give a more detailed description of the proposed algorithm.

3.2 Representation scheme

The encoding of the candidate solutions is one of the deciding factors for the outcome of an evolutionary search. The representation should be compact and concise enough to direct the search effort towards relevant information and to minimize disruption by the genetic operators of the results. At the same time, the encoded data should contain all the determining elements for the optimization of the performance of the solution.

In FL system optimization, encoding the solutions using binary or real strings of the kind used in standard GAs and EVSs is likely to generate very long chromosomes. This has the undesirable effects of slowing down the EA execution speed and increasing the possibility of disruption of good genetic material. Moreover, a one-dimensional representation of the fuzzy RB breaks the behavioural links between physically overlapping rules. Adjacent rules are in fact likely to have similar consequences and to cooperate in the determination of the output of the FL system. This feature is believed to provide useful information that should not be lost in the evolutionary process.

The proposed algorithm represents candidate solutions using multichromosome genotypes. Figure 1 gives

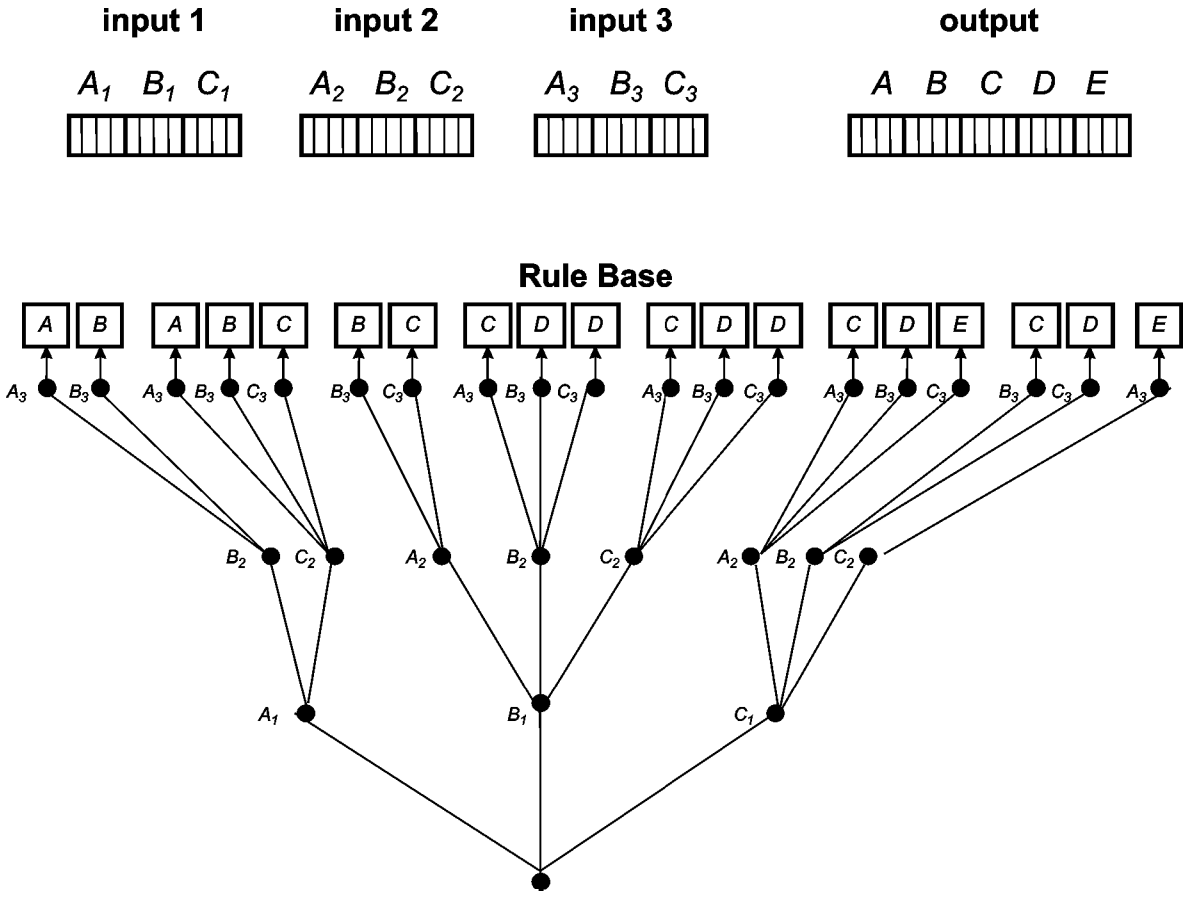

Fig. 1 Representation scheme

an example of an encoded solution for an FL system having three input variables, each partitioned into three linguistic terms, and one output variable partitioned into five fuzzy terms.

A separate chromosome is used to describe the partition of each input and output variable, each chromosome being composed of a number of genes equal to the number of linguistic terms. Each gene is a real-valued string encoding the parameters defining the location and the shape of one MF.

The shape of the fuzzy MFs has been chosen to be a trapezoid for its generality and approximation capability. Consequently, four parameters (*nucleotides*) per gene are required to describe a linguistic term completely, each parameter being related to the location of one of the anchor points of an MF. If desired, other MF shapes can be evolved by simply changing the structure of the chromosomes at the gene level. Figure 2 details the MF encoding scheme.

The fuzzy RB is represented as a multilevel decision tree, the depth of the tree corresponding to the dimensionality of the input space. The rule antecedent is encoded in the full path leading to the fuzzy consequent, each node being associated with a rule condition. The nodes at the last level of the decision tree are linked to the rule consequent, the latter containing a fuzzy action for each output variable. In the example of Fig. 1, the encoded RB is for a single-output FL system and each path leads to a single-action term label. As shown in the example, the genome of the solution contains only active rules and defines a non-extensive coverage of the input space.

The tree structure of the RB encoding allows the system to relate the genetic information quickly to its spatial location on the fuzzy response surface. In particular, different rule paths passing through nodes referring to neighbouring MFs identify groups of overlapping fuzzy rules. At the same time, the representation of the RB as a decision tree allows a simple and economical software implementation and fast processing times.

The above representation scheme defines variable-size chromosomes, the number of genes being determined by the number of fuzzy terms and rules. This type of encoding is preferred for its flexibility and for the possibility of minimizing storage space and saving computing resources. Fixed-size chromosomes in fact carry an upper limit of valid genetic information. This leads to two possible situations. In the first case, the designer is forced to determine the number of fuzzy

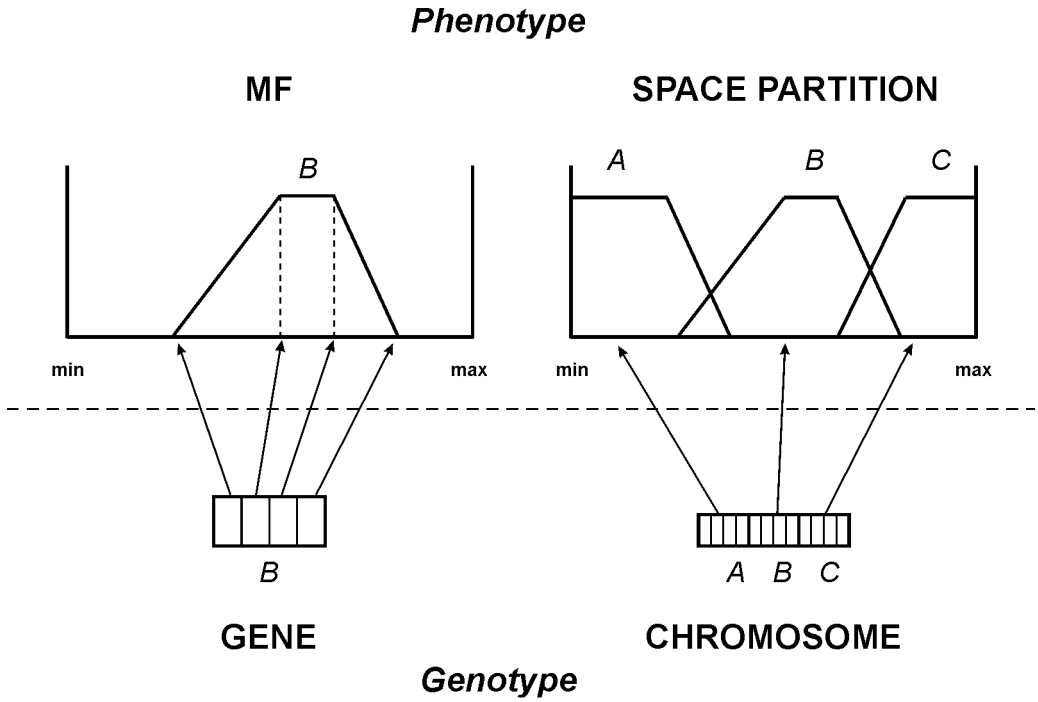

Fig. 2 MFs encoding

terms and rules prior to the learning process, this choice affecting the final result of the search. In the second case, a number of locations are defined and a special allele is used to mark unused ones. This results in the generation of a variable amount of unused genomes, diverting computational resources from the search process. The definition of large genotypes increases the occurrence of empty locations, while small genotypes may be insufficient for the purposes of learning.

3.3 Reproduction scheme

The proposed algorithm uses generational replacement [**17**] to renew the population of solutions. According to this procedure, at every *generation* (i.e. reproduction cycle) the whole population is replaced by a set of newly created individuals.

The choice has been motivated by the higher exploration capability of generational replacement and by its superior robustness to suboptimal convergence and noisy fitness evaluations. The ability to minimize the effects of inaccurate fitness measurements is particularly important where an extensive evaluation of the performance of the solutions would be impractical. This case occurs in several control applications such as the inverted pendulum problem.

The reproduction cycle repeats itself until a predefined stopping criterion is met. This may be when a certain number of generations has elapsed or some accuracy requirements have been met. At that point, the fittest individual is selected as the final solution and the EA is terminated.

3.4 Selection scheme

The purpose of the selection procedure is to pick out of the current population the set of individuals that will be used to create the next population. The choice of which solutions to reproduce determines which regions of the solution space will be sampled next. This choice is often the result of a trade-off between the necessity of exploring new areas of the search space (*exploration*) and the need to sample more accurately around the most promising solutions (*exploitation*).

The proposed algorithm uses the following adaptive scheme which comprises three main stages:

Stage 1. The fitness measure $f(i)$ of each solution is normalized in the interval $[0, 100]$. The least fit individual will be assigned fitness value 0 and the fittest individual will be assigned value 100. The normalization function is the following:

$$f'(i) = 100 \times \left[\frac{f(i) - f(w)}{f(b) - f(w)}\right] \tag{1}$$

where $f(w)$ and $f(b)$ are respectively the fitness of the worst and the fitness of the best individual. The purpose of equation (1) is to eliminate the *scaling problem* [**62**].

Stage 2. For each solution i, the *mating chance* $m(i)$ is defined by adding a random positive offset to the normalized fitness value $f(i)$. The magnitude of the added value falls into an interval equal to the difference between the maximum (100) and the average normalized fitness of the solutions:

$$m(i) = f'(i) + (100 - \mu_{f'}) \cdot rand \tag{2}$$

where $\mu_{f'}$ is the average value of the normalized fitnesses and *rand* is a uniformly distributed random real number in the interval [0, 1]. The solutions are ranked according to their mating chance and the best half of the population is selected for reproduction. The selected individuals are inserted in their ranking order into a temporary list A. The procedure is then repeated once; i.e. the individuals are assigned a new mating chance, they are ranked according to it and the half-population having the highest mating chance is included in a second temporary list B. Lists A and B form the mating pool.

The aim of the second stage is to adjust the selection pressure according to the population fitness distribution. At the beginning of the search process, the difference between the highest performing individual and the average fitness is likely to be large. For this reason, the mating chance of the lowest performing solutions can be greatly enhanced because of the large offset range. This mechanism reduces the selection pressure and encourages the exploration of the search space. Moreover, due to the construction of the mating pool, the maximum number of copies of the same individual in the pool is naturally restricted to two, thus limiting the possibility of a few superindividuals monopolizing the reproduction process. At the same time, the best performing solutions still have a good chance to be selected, ensuring that valuable genetic material will not be lost.

As the evolution proceeds, the solutions begin to converge and the difference between the maximum and the average population fitness decreases. This reduces the selection possibility of lower performing individuals and the search becomes more deterministic and exploitative. As the fitness of the individuals improves, the search therefore becomes more similar to a gradient-based optimization. This mechanism helps to overcome one of the major weaknesses of EAs, that is their slowness at converging to the optimum point once the main peak is located.

Finally, it should be noted that two kinds of adaptation are used. The first in stage 1 focuses on the scaling of the fitness to a standard window of positive values. The second kind of adaptation acts in stage 2 on the stochastic noise level in the determination of the mating pool, and it is the means through which the selection pressure is modulated.

Stage 3. At this stage, the parent couples are formed. The solutions are paired associating the first element of list A to the last element of list B, the second element of list A to the penultimate element of list B and so on, keeping on scanning the two lists in opposite directions. Following this procedure, the best individuals are likely to be paired up with those of lowest fitness as they are mostly concentrated at opposite extremes of the two lists. The aim is to grow the performance of the entire population steadily, concentrating mainly on the improvement of the average population fitness. This should act as a further policy against premature convergence of the solutions.

As a last remark, it should be noted that the proposed selection procedure requires more computation than proportional selection, fitness ranking or tournament selection. However, as pointed out in reference [**63**], in most optimization problems, the time taken to evaluate the solutions is much greater than the time spent on the genetic operations. Directing efforts towards operator time savings is therefore likely not to pay off.

3.5 Recombination operators

The purpose of the recombination operator is to mix chunks of the genetic material of the parents to generate better offspring. Crossover is therefore involved with the redistribution of genes among the population, looking for the most successful combination.

As previously mentioned, there are three ways of modifying the overall fuzzy mapping: by manipulating the input and output partitions, by changing the control policy and by performing both operations. The first case can be achieved by swapping sets of input and output MFs between the chromosomes of the parents, the second case by exchanging sets of fuzzy rules and the third case by swapping both rules and MFs. The proposed algorithm defines three different types of crossover to perform the above three operations.

Each of the proposed recombination operators behaves as a two-point crossover operator [**16**], cutting and pasting information defined into hypercubes of the input–output characteristic. For each dimension, the side of the hypercube is determined by a start and by an end fuzzy term. The procedure is as follows: for every input and output variable v_i of the first parent P, the start A_{is} and the end A_{ie} terms are randomly selected. The MFs of the second parent P' are then searched for the most similar terms to A_{is} and A_{ie}. These terms will be the corresponding start A'_{is} and end A'_{ie} terms of P'. An example of the pairing procedure applied to one variable is sketched in Fig. 3.

The first recombination operator acts on *species_1* individuals and swaps sets of input and output fuzzy

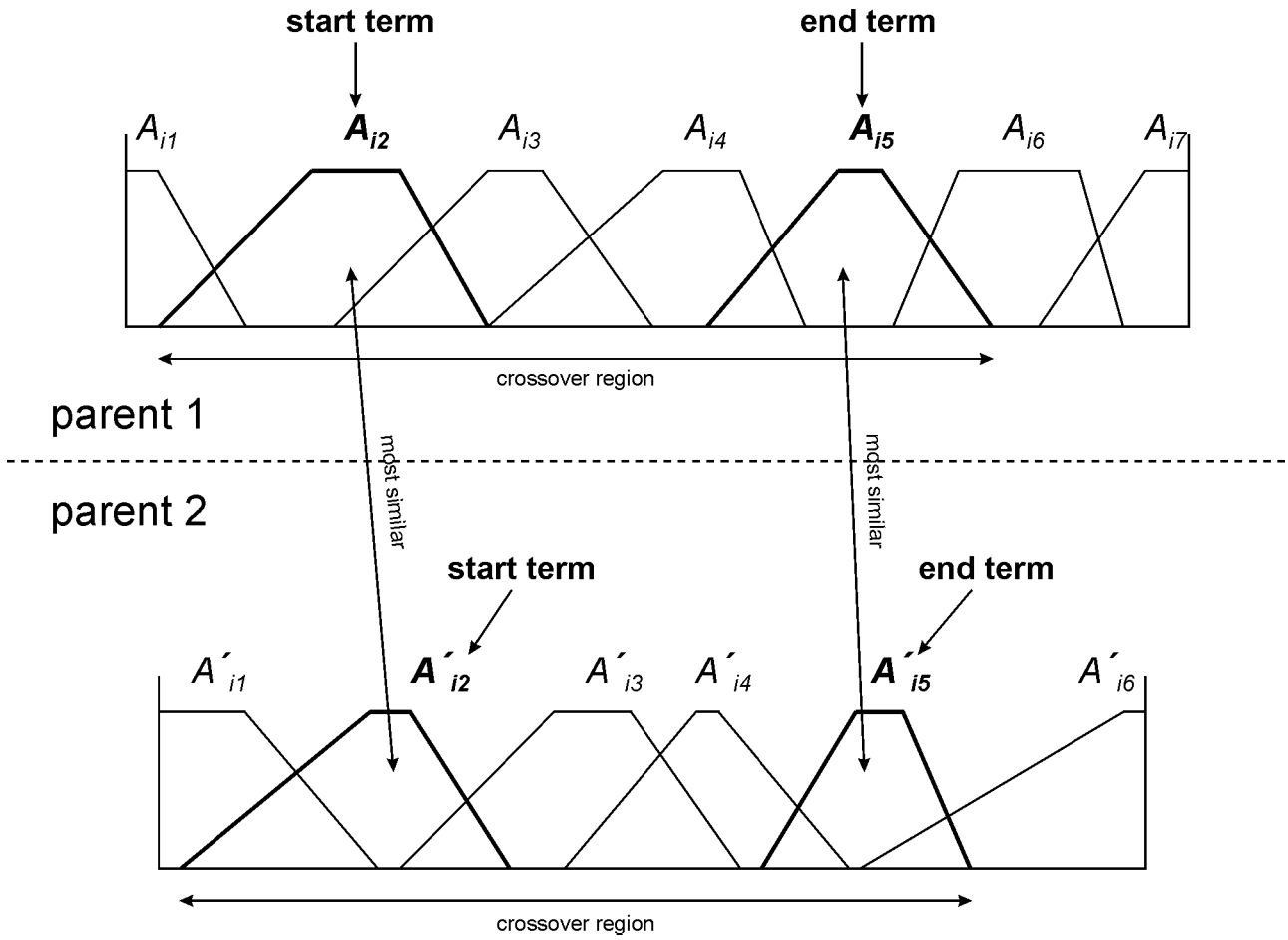

Fig. 3 Term matching

terms. For each variable, the terms between the start and the end terms are exchanged. This is equivalent to a two-point crossover applied to the chromosome encoding the fuzzy partition of the variable. Each offspring receives the RB from one of the two parents.

Each child therefore inherits from one parent part of the MFs and the whole RB. Some kind of 'translation' is needed to adjust those fuzzy rules that are defined over regions of swapped conditions (actions). For each variable, it is possible that an uneven number of fuzzy terms may be swapped or that the location of the new MFs may considerably differ from the old ones. To limit disruption to the response of the system, the proposed algorithm matches each of the new fuzzy terms with the old set of terms. The matching criterion is the same as described in Fig. 3. Each of the new terms is paired with the old fuzzy term having the most similar MF, and all the old rule conditions (actions) are re-stated using the new term. This operation just changes the labels of the fuzzy decision tree encoding the RB. Old rules (paths) containing conditions (nodes) not paired with any new term are eliminated.

Figure 4 shows an example of RB translation for a two-input one-output fuzzy system. For the sake of simplicity, only the translation of the rule conditions is described. On the left side of the figure, the pairing of the nodes is shown. New nodes A'_{ij} have been represented in bold. On the right side of the figure, the translation of the fuzzy decision table is shown. In this case, translating one condition to another means copying a row (column) from the old decision table to the corresponding row (column) of the new table. Old rows (columns) corresponding to unmatched terms are eliminated. This is the case with the row corresponding to term A_{02} and the column relating to term A_{15}. If the new decision table contains one or more extra rows (columns), each new entry is randomly picked from the neighbouring cells in the same column (row). The figure also shows the case where two new terms, A'_{01} and A'_{02} respectively, are paired with the same old term A_{01}. In this case, the second best matching term (A_{02}) is found for A_{01}. The row corresponding to A'_{01} in the new table is copied from the row corresponding to A_{01} in the old table. The row relating to A'_{02} is randomly formed by elements taken from the rows of the old table corresponding to A_{01} (best matching term) and A_{02} (second best matching term). There is no action defined for entry $A'_{02} \wedge A'_{10}$ as no action is defined for $A_{01} \wedge A_{10}$

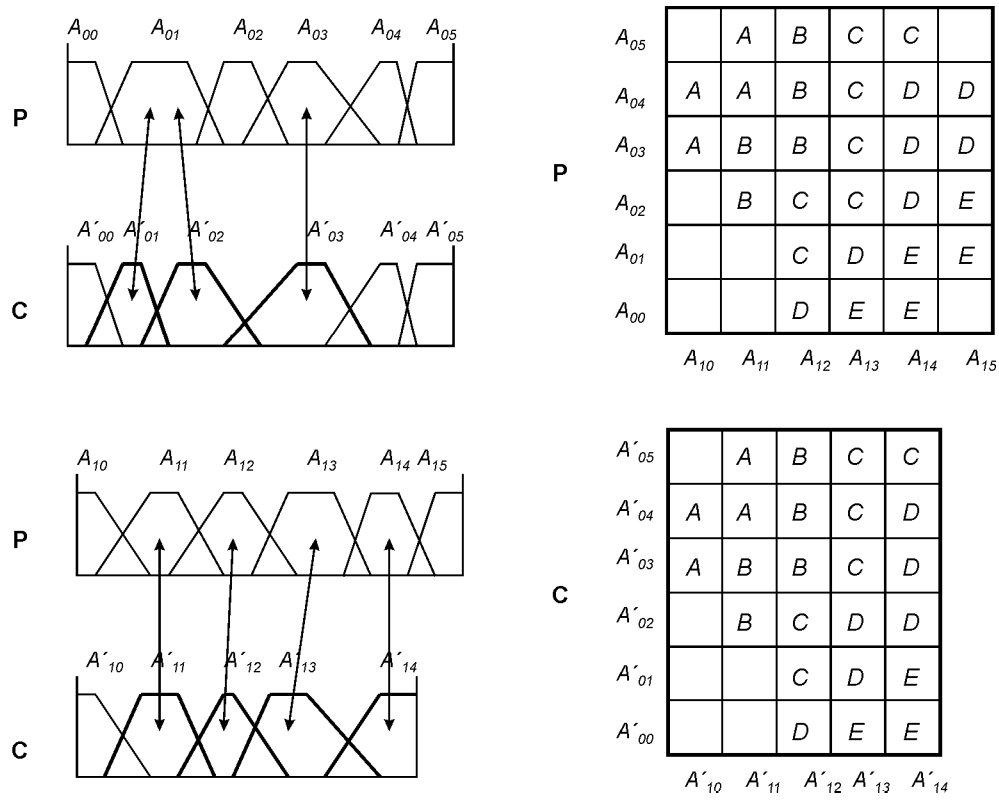

Fig. 4 Rule base translation procedure

and $A_{02} \wedge A_{10}$. The action for entry $A'_{02} \wedge A'_{11}$ is automatically taken from $A_{02} \wedge A_{11}$ as no action is defined for the antecedent $A_{01} \wedge A_{11}$. The purpose of the procedure is to fill the elements of the extra rows (columns) by blending the response policies of the two neighbouring rows (columns), thus avoiding the duplication of entire rows (columns).

The second recombination operator acts on *species_2* individuals and swaps portions of the fuzzy RB. All the rules contained in the hyperbox of the control surface delimited by the set of start and end terms are exchanged between the parent chromosomes. The fuzzy decision tree of the offspring is therefore formed by exchanging 'bunches' of branches between the two parent chromosomes. Each child inherits the chromosomes encoding the input and output MFs from one of the parents.

An offspring therefore receives from one of its parents part of the RB and the complete description of the input and output space partitions. The same procedures described above apply to the translation of the set of rules inherited from the other parent. Figure 5 shows an example of the crossover operation for a two-input one-output fuzzy system. Again, for the sake of simplicity, only the translation of the rule conditions is described. The left side of the figure summarizes the pairing of the nodes while the right side describes the crossover operation. The shaded background indicates the area where rules are swapped. Child C receives a block of 3×2 rules from parent P'. However, because of the different space partition, the portion of the RB inherited must be transformed into a 3×3 block of rules. For this purpose, the rules relating to conditions A'_{11} and A'_{12} are translated into the columns corresponding to the most similar terms A_{11} and A_{13} respectively. Three new rules are generated for the column relating to condition A_{12} by randomly picking the output from columns A_{11} and A_{13} respectively. The opposite transformation is performed for child C', where the block of 3×3 rules received from parent P is translated into a 3×2 block of rules. In this case, the rules corresponding to condition A_{12} are dropped and only two columns are copied.

The third recombination operator acts on *species_3* individuals and mixes the entire KB contained in the hyperbox delimited by the set of start and end terms. The same procedures described above are used for swapping fuzzy rules and MFs. As every fuzzy rule is inherited together with the description of its conditions, only the set of rule actions may need translation.

At the end of every crossover operation, for every input and output variable, a fuzzy partition 'repair' algorithm is run to prevent the overlap of more than two MFs at any point of the universe of discourse. The reasons for introducing this constraint are to achieve increased transparency of the KB, a reduction in the

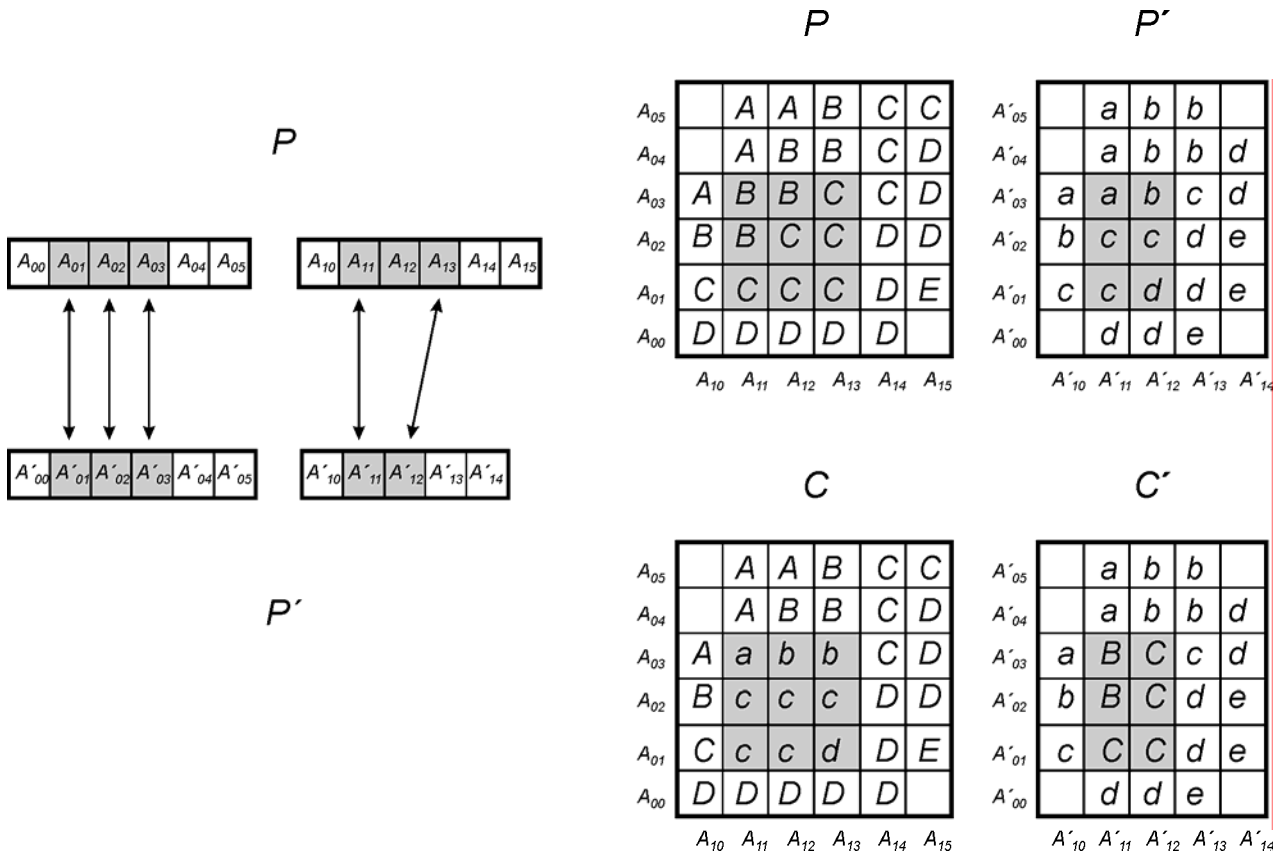

Fig. 5 Rule base recombination

number of rules fired at any input presentation and avoidance of excessive growth in the number of fuzzy terms.

3.6 Mutation operators

Genetic mutations are employed to modify the shape of the control surface randomly. The extent of the modification depends on the level of the KB at which the process takes place. The proposed algorithm uses four different mutation operators that modify the MF of one fuzzy term, create a new term in the fuzzy partition of a variable, delete a term from the partition of a variable and change the action of a rule. Each species of individuals is subjected to a different set of mutation operators, but no more than one KB modification event can occur per genotype.

The probability of a KB mutation event is specified regardless of the type of solution by the predefined system parameter *mut_rate*. Once an individual has been selected, its species will determine the type of mutation.

The KB manipulation operators acting on *species_1* individuals work at the level of input and output fuzzy partitions. The solutions belonging to the first sub-population may therefore undergo any of the first three types of mutation.

The modification of one fuzzy MF is the least disruptive and for this reason it is the most likely event. Once the gene encoding the MF of a linguistic term has been randomly selected, all its nucleotides (i.e. its anchor points) are mutated. The amount of change varies with the shape of the MF and the partition of the fuzzy space. Fuzzy MFs defined over large space intervals are varied more than narrowly localized MFs. This reflects the different requirements of granularity in the fuzzy space partition. Moreover, the magnitude of MF mutations follows a heuristic criterion, adjusting the amount of overlap between MFs according to the vagueness of their definition. The mutation of the anchor points of an MF is performed according to the following heuristic formula:

$$\Delta p_i = rand \cdot delta \cdot \left(\frac{1 + |o - f|}{3} \right) \qquad (0 \leqslant i < n) \quad (3)$$

where Δp_i is the change in the position of anchor point p_i, *rand* is a random number in $[-1, 1]$, *delta* is the smaller of the two distances from point p_i to points p_{i+1} and p_{i-1}, and n is the number of MF anchor points; f is the degree of fuzziness of the fuzzy term and is defined as the portion of the MF support interval where the truth degree is less than 1; o, the degree of overlap, is defined as the portion of the support interval that

overlaps with some other MF. The algorithm penalizes with larger mutations MFs having low fuzziness and a high degree of overlap or MFs having high fuzziness but a low degree of overlap.

The other two mutation operators acting on *species_1* individuals delete (or add) one fuzzy term from (or to) the fuzzy space partition of one randomly selected variable. To favour the generation of FL/FNN systems having a minimal number of linguistic terms, the term 'deletion operator' has been given a slightly higher probability than the term 'addition operator'. Once the fuzzy partition of a variable has been modified, the RB is translated accordingly using the same procedures as applied to the crossover operation.

The genetic operators acting on *species_2* individuals manipulate the fuzzy KB at the level of fuzzy rules. The solutions belonging to this second group may therefore undergo the fourth kind of mutation. The rule action mutation operator substitutes one action term A_i of a randomly selected fuzzy rule with the previous A_{i-1} or the next A_{i+1} term in the partition of the same fuzzy variable. Changing a rule action term with a spatially contiguous fuzzy term reduces the possibility of generating a poor offspring.

The genetic manipulations applied to *species_3* individuals may happen at all levels of the fuzzy KB. The solutions belonging to this group may therefore undergo any of the above-mentioned four mutation operations.

Finally, a fifth mutation operator has been defined to change the gene determining the species of an individual. This operator is separate from the others. It has its own rate of occurrence and may affect an individual that has already undergone KB modification. Its action is to change the species of the selected individual randomly.

4 CONCLUSIONS AND FURTHER WORK

An evolutionary algorithm for the automatic generation of the knowledge base for fuzzy logic systems has been presented in this paper. The algorithm dynamically adjusts the focus of the genetic search by dividing the population into three subgroups, each concerned with a different level of knowledge base optimization. It also includes a new adaptive selection routine that aims to optimize the selection pressure throughout the learning phase.

Further efforts should be addressed at the investigation of the effectiveness and the robustness of the proposed EA. For this purpose, the learning algorithm should be tested on different problems, including systems identification, control and data classification.

A detailed comparative analysis of the proposed evolutionary algorithm against other learning techniques is needed. A deeper investigation into the effectiveness of the proposed selection routine should also be conducted. Finally, it would be beneficial to study the internal mechanisms of the evolutionary search, with particular regard to the interactions between the three population subgroups.

ACKNOWLEDGEMENTS

This research was sponsored by Hewlett–Packard and the Welsh Assembly under the European Regional Development Fund programme.

REFERENCES

1 **Fogel, D. B.** *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd edition, 2000 (IEEE Press, New York).
2 **Zadeh, L. A.** Fuzzy sets. *Inf. and Control*, 1965, **8**, 338–353.
3 **Rumelhart, D. E.** and **McClelland, J. L.** *Parallel Distributed Processing: Exploration in the Micro-Structure of Cognition*, Vols 1 and 2, 1986 (MIT Press, Cambridge, Massachusetts).
4 **Lippmann, R. P.** An introduction to computing with neural nets. *IEEE ASSP Mag.*, 1987, 4–22.
5 **Pham, D. T.** and **Liu, X.** *Neural Networks for Identification, Prediction and Control*, 1995 (Springer-Verlag, London).
6 **Rechenberg, I.** *Cybernetic Solution Path of an Experimental Problem*, 1965, Trans. 1122 (Royal Aircraft Establishment, Farnborough, Hampshire).
7 **Fogel, L. J., Owens A. J.** and **Walsh, M. J.** *Artificial Intelligence Through Simulated Evolution*, 1966 (John Wiley, New York).
8 **Holland, J. H.** *Adaptation in Natural and Artificial Systems*, 1975 (University of Michigan Press, Ann Arbor, Michigan).
9 **De Jong, K. A.** An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, Michigan, 1975.
10 **Rechenberg, I.** Artificial evolution and artificial intelligence. In *Machine Learning Principles and Techniques* (Ed. R. Forsyth), 1989, pp. 83–103 (Chapman and Hall, London).
11 **Bäck, T., Hoffmeister, F.** and **Schwefel, H. P.** A survey of evolution strategies. In Proceedings of 4th International Conference on *GAs and Applications*, San Mateo, California, 1991, pp. 2–9.
12 **Bäck, T.** and **Schwefel, H. P.** An overview of evolutionary algorithms for parameter optimisation. *Evolutionary Comput.*, 1993, **1**(1), 1–23.
13 **Schwefel, H. P.** On the evolution of evolutionary computation. In Proceedings of 3rd International Conference of IEEE World Congress on *Computer Intelligence*, Orlando, Florida, June 1994, pp. 116–124.
14 **Bäck, T.** *Evolutionary Algorithms in Theory and Practice*, 1996 (Oxford University Press, New York).
15 **Fogel, D. B.** An analysis of evolutionary programming. In Proceedings of First International Conference on *Evolutionary Progress*, 1992, pp. 43–51.

16 **Goldberg, D. E.** *Genetic Algorithms in Search, Optimisation and Machine Learning*, 1989 (Addison Wesley).

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

18 **Michalewicz, Z.** A hierarchy of evolution programs: an experimental study. *Evolutionary Comput.*, 1993, **1**(1), 51–76.

19 **Winter, G., Periaux, J., Galan, M.** and **Cuesta P.** *Genetic Algorithms in Engineering and Computer Science*, 1995 (John Wiley, New York and Chichester).

20 **Pham, D. T.** and **Karaboga, D.** *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, 2000 (Springer-Verlag, London).

21 **Karr, C. L., Freeman, L. M.** and **Meredith, D. L.** Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm. *SPIE Intell. Contr. and Adaptive Syst.*, 1989, **1196**, 274–288.

22 **Mamdani, E. H.** Application of fuzzy algorithms for control of simple dynamic plant. *Proc. Inst. Elect. Engrs*, 1974, **121**(12), 1585–1588.

23 **Fantuzzi, C.** Bases of fuzzy control. In Proceedings of International Summer School on *FLC Advances in Methodology*, Ferrara, Italy, 1998, pp. 1–34.

24 **Karr, C. L.** Design of a cart–pole balancing fuzzy logic controller using a genetic algorithm. *SPIE Applic. of AI IX*, 1991, **1468**, 26–36.

25 **Karr, C. L.** Design of an adaptive fuzzy logic controller using a genetic algorithm. In Proceedings of 4th International Conference on *GAs and Applications*, San Mateo, California, 1991, pp. 450–457.

26 **Karr, C. L.** and **Gentry, E. J.** Fuzzy control of pH using genetic algorithms. *IEEE Trans. on Fuzzy Syst.*, 1993, **1**(1), 46–53.

27 **Karr, C. L.** Adaptive control with fuzzy logic and genetic algorithms. In *Fuzzy Sets, NNs and Soft Computers* (Eds R. R. Yager and L. A. Zadeh), 1994, pp. 345–367 (Van Nostrand Reinhold, New York).

28 **Surmann, H., Kanstein, A.** and **Goser, K.** Self-organising and genetic algorithms for an automatic design of fuzzy control and decision systems. In Proceedings of EUFIT, First European Congress on *Fuzzy and Intelligent Technology*, Aachen, Germany, September 1993, pp. 1097–1104.

29 **Jin, G.** Intelligent fuzzy logic control of processes with time delays. PhD thesis, University of Wales, Cardiff, 1995.

30 **Fukuda, T., Hasegawa, Y.** and **Shimojima, K.** Hierarchical fuzzy reasoning, adaptive structure and rule by genetic algorithms. In Proceedings of First IEEE Conference on *Evolutionary Computations (ICEC)*, Orlando, Florida, 1994, pp. 601–606.

31 **Shimojima, K., Fukuda, T.** and **Hasegawa, Y.** Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. *Fuzzy Sets and Syst.*, 1995, **71**, 295–309.

32 **Furuhashi, T., Matsushita S.** and **Tsutsui, H.** Evolutionary fuzzy modeling using fuzzy neural networks and genetic algorithms. In Proceedings of IEEE International Conference on *Evolutionary Computations*, Indianapolis, Indiana, 1997, pp. 623–627.

33 **Smith, S. F.** A learning system based on genetic adaptive algorithms. PhD thesis, Department of Computer Science, University of Pittsburgh, Pennsylvania, 1980.

34 **Holland, J. H.** and **Reitman, J. S.** Cognitive systems based on adaptive algorithms. In *Pattern-Directed Inference Systems* (Eds D. A. Waterman and F. Hayes-Roth), 1978 (Academic Press, New York).

35 **Valenzuela-Rendon, M.** The fuzzy classifier system: motivations and first results. In Proceedings of First Workshop on *Parallel Problem Solving from Nature* (*PPSN I*) (Eds H. P. Schwefel and R. Männer), Dortmund, Germany, October 1990, pp. 338–342.

36 **Machado, R. J.** and **Freitas da Rocha, A.** Evolutive fuzzy neural networks. In Proceedings of IEEE International Conference on *Fuzzy Systems*, San Diego, California, 1992, pp. 493–500.

37 **Bonarini, A.** ELF: learning incomplete fuzzy rule sets for an autonomous robot. In Proceedings of EUFIT, First European Congress on *Fuzzy and Intelligence Technology*, Aachen, Germany, September 1993, pp. 69–75.

38 **Bonarini, A.** Evolutionary learning of general fuzzy rules with biased evaluation functions: competition and cooperation. In Proceedings of First IEEE Conference on *Evolutionary Computations (ICEC)*, Orlando, Florida, 1994, pp. 51–56.

39 **Pham, D. T.** and **Karaboga, D.** A new method to obtain the relation matrix of fuzzy logic controllers. In Proceedings of 6th International Conference on *AI in Engineering*, Oxford, 1991, pp. 567–581.

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

41 **Thrift, P.** Fuzzy logic synthesis with genetic algorithms. In Proceedings of 4th International Conference on *GAs and Applications*, San Mateo, California, 1991, pp. 509–513.

42 **Kropp, K.** and **Baitinger, U. G.** Optimisation of fuzzy logic controller inference rules using a genetic algorithm. In Proceedings of EUFIT, First European Congress on *Fuzzy and Intelligence Technology*, Aachen, Germany, September 1993, pp. 1090–1096.

43 **Hwang, W. R.** and **Thompson, W. E.** Design of intelligent fuzzy logic controllers using genetic algorithms. In Proceedings of 3rd IEEE International Conference on *Fuzzy Systems*, Piscataway, New Jersey 1994, Vol. 2, pp. 1383–1388.

44 **Feldman, D. S.** Fuzzy network synthesis with genetic algorithms. In Proceedings of 5th International Conference on *GAs and Applications*, Urbana, Illinois, 1993, pp. 312–317.

45 **Buhusi, C.** Learning by simulating evolution in automatic fuzzy systems synthesis. In Proceedings of 3rd IEEE International Conference on *Fuzzy Systems*, Piscataway, New Jersey, 1994, **2**, 1308–1313.

46 **Takagi, T.** and **Sugeno, M.** Fuzzy identification of systems and its application to modelling and control. *IEEE Trans. Syst., Man and Cybernetics*, 1985, **15**, 116–132.

47 **Parodi, A.** and **Bonelli, P.** A new approach to fuzzy classifier systems. In Proceedings of 5th International Conference on *GAs and Applications*, Urbana, Illinois, 1993, pp. 223–230.

48 **Lee, M. A.** and **Takagi, H.** Dynamic control of genetic algorithms using fuzzy logic techniques. In Proceedings of 5th International Conference on *GAs and Applications*, Urbana, Illinois, 1993, pp. 76–83.

49 Kinzel, J., Klawonn, F. and **Kruse, R.** Modifications of genetic algorithms for designing and optimising fuzzy controllers. In Proceedings of First IEEE Conference on *Evolutionary Computations* (*ICEC*), Orlando, Florida, 1994, pp. 28–33.

50 Heider, H. and **Drabe, T.** Fuzzy system design with a cascaded genetic algorithm. In Proceedings of IEEE International Conference on *Evolutionary Computations*, Indianapolis, Indiana, 1997, pp. 585–588.

51 Buckles, B. P., Petry, F. E., Prabhu, D., George, R. and **Srikanth, R.** Fuzzy clustering with genetic search. In Proceedings of First IEEE Conference on *Evolutionary Computations* (*ICEC*), Orlando, Florida, 1994, pp. 46–50.

52 Cooper, M. G. and **Vidal, J. J.** Genetic design of fuzzy controllers: the cart and jointed-pole problem. In Proceedings of 3rd IEEE International Conference on *Fuzzy Systems*, Piscataway, New Jersey, 1994, Vol. 2, pp. 1332–1337.

53 Liska, J. and **Melsheimer, S. S.** Complete design of fuzzy logic systems using genetic algorithms. In Proceedings of 3rd IEEE International Conference on *Fuzzy Systems*, Piscataway, New Jersey, 1994, Vol. 2, pp. 1377–1382.

54 Ng, K. C. and **Li, Y.** Design of sophisticated fuzzy logic controllers using genetic algorithms. In Proceedings of First IEEE Conference on *Evolutionary Computations* (*ICEC*), Orlando, Florida, 1994, pp. 1708–1712.

55 Ng, K. C., Li, Y., Murray-Smith, D. J. and **Sharman, K. C.** Genetic algorithms applied to fuzzy sliding mode controller design. In Proceedings of GALESIA First IEE/IEEE International Conference on *GAs in Engineering Systems: Innovations and Applications*, Sheffield, 1995, pp. 220–225.

56 Homaifar, H. and **McCormick, E.** Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans. Fuzzy Syst.*, 1995, **3**(2), 129–139.

57 Seng, T. L., Khalid, M. B. and **Yusof, R.** Tuning of a neuro-fuzzy controller by genetic algorithm. *IEEE Trans. Syst. Man and Cybernetics*, 1999, **29**(2), 226–236.

58 Chiang, C. K., Chung, H. Y. and **Lin, J. J.** A self-learning fuzzy logic controller using genetic algorithms with reinforcements. *IEEE Trans. Fuzzy Syst.*, 1997, **5**(3), 460–467.

59 Berenji, H. R. A reinforcement learning-based architecture for fuzzy logic control. *Int. J. Approx. Reasoning*, 1992, **6**, 267–292.

60 Shi, Y., Eberhart, R. and **Chen, Y.** Implementation of evolutionary fuzzy systems. *IEEE Trans. Fuzzy Syst.*, 1999, **7**(2), 109–119.

61 Mahfoud, S. W. A comparison of parallel and sequential niching methods. In Proceedings of 6th International Conference on *GAs and Applications*, Pittsburgh, Pennsylvania, 1995, pp. 136–143.

62 Grefenstette, J. J. Optimisation of control parameters for genetic algorithms. *IEEE Trans. Syst., Man and Cybernetics*, 1986, **16**(1), 122–128.

63 Goldberg, D. E. and **Deb, K.** A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of GAs*, 1991, pp. 69–93 (Morgan Kaufmann, San Mateo, California).