Modular and Hierarchical Evolutionary Design of Fuzzy Systems

Myriam Delgado
DCA - UNICAMP
13083-970 - Campinas-SP - Brazil
myriam@dca.fee.unicamp.br

Fernando Von Zuben
DCA - UNICAMP
vonzuben@dca.fee.unicamp.br

Fernando Gomide
DCA - UNICAMP
gomide@dca.fee.unicamp.br

Abstract

The paper introduces a modular, hierarchical evolutionary method to design fuzzy systems. The method uses genetic algorithms to evolve a population of rule-based fuzzy models. For this purpose, three hierarchical modules are defined namely, partition, population of granules, and population of fuzzy graphs. The evolutionary process is designed to find the best set of fuzzy rules induced by the granularity required, the form of the membership functions and the inference procedure. This hierarchical configuration guides to the implementation of an effective process of cooperation and competition among rules, responsible for the small cardinality of the resulting set of rules. Simulation results show that the method does increase flexibility to design fuzzy models, preserves the computational tractability, and improves the descriptive nature of the final solution.

1 INTRODUCTION

In fuzzy set theory, relations induced by if-then rules are the basic building blocks to represent and process information [Klir and Forger, 1988]. Key design issues must be carefully considered to develop efficient rule-based fuzzy models. They include the choice of relevant input/output variables, granularity of the universes, number of rules to be used, shape of the membership functions, and inference method (see Pedrycz and Gomide [1998] for details). To find the optimal values of all parameters involved in the design of fuzzy systems is a very demanding task, even when expert knowledge on problem domain exists. The situation becomes more complicated in the case of complex engineering problems because arbitrary specifications may result in undesirable solutions.

Genetic algorithms, neural networks, and more recently hybrid systems, have been successfully used to design fuzzy systems [Pedrycz and Gomide, 1998]. Automatic design of fuzzy systems constitutes a promising and challenging research area in which evolutionary computation has a significant role. The use of genetic operators, like reproduction, mutation and crossover [Holland, 1992], appears as a powerful procedure to automatically find fuzzy model parameters, as indicated in Thrift [1991] and Karr [1991]. For instance, in Thrift [1991], genetic algorithms are used to select adequate fuzzy sets for the consequent part of if-then fuzzy rules. In this case, each individual represents a set of fuzzy rules. The shape of the membership functions is fixed and evolution techniques define which rules should be part of the model. In Karr [1991], membership functions and fuzzy rules are both determined by genetic algorithms. First, mutation and crossover are applied to discover the number of fuzzy rules, with the shape of each membership function maintained fixed. Only at a second stage the shape of the parameters are tuned using GA. The problem of membership function definition has also been considered in Nomura et al. [1992] and Murata et al. [1998]. Here again, the number and the shape of membership functions are found via evolution. In this case, each individual is binary-coded to represent a set of fuzzy rules, but fuzzy sets overlapping is fixed. Concerning both rules and membership functions, Karr and Gentry [1993] applied GA to obtain a set of production rules for pH control, whereas Takagi and Lee [1993] used TSK class of fuzzy models as a template model. In Karr and Gentry [1993], binary-coded chromosomes represent membership functions that, evolved by GA techniques, are used by a conventional fuzzy controller. In Takagi and Lee [1993], triangular membership functions are codified on a three-value string: left base, right base, and the distance between centers. The output is a linear relationship on the inputs. The chromosome codification is binary and each individual represents a solution, i.e., a set of rules that models the system.

Differently, in Valenzuela-Rendon [1991], each individual represents one fuzzy rule. Antecedents and consequents have a binary coding, which means that the number of bits determines the number of fuzzy sets for each variable. The shape of the membership functions are previously defined, and overlapping between them are allowed. In an alternative approach, Homaifar and McCormick [1995] used integer values strings. After decoding, these strings define the linguistic values of the input variables and associated membership functions. The modal value of membership functions and dispersion (or base) of output variables are all fixed. Membership function overlapping is allowed with a guaranteed minimal value.

The common feature among these works is the fact that the codification method treats all model parameters at the same level. Therefore, it is impossible to evaluate (via a fitness measure) the average performance of any specific parameter. An alternative approach is discussed in Hoffmann and Pfister [1995] where messy genetic algorithm techniques [Goldberg et al., 1989] are used to design a fuzzy controller with hierarchical prioritized structures [Yager, 1993]. Two different levels (individual rules and prioritized set of rules) are used in the evolutionary process.

Another important restriction of most of the works discussed above concerns the consideration of a small part of the design parameters. In this paper, genetic algorithms are emphasized as powerful design tool to find an optimal set of parameters for fuzzy systems models. A modular, hierarchical method is proposed to evolve a large set of design parameters of fuzzy models. The method provides the flexibility required to address complex design problems within the realm of fuzzy set theory. It also provides the designer the option to select optimization of the set, or a subset of the parameters only. Simulation results are included to show the usefulness of the approach proposed.

2 The EVOLUTIONARY DESIGN METHOD

The Modular and Hierarchical Evolutionary Fuzzy System (MHEFS) method aims at higher flexibility to design the fuzzy rules that best describe a given system. Its modular and hierarchical nature was inspired by an application of GA introduced in Moriarty and Miikkulainen [1998] to train one hidden layer neural networks.

The MHEFS considers simultaneously many important aspects that must be addressed when designing a fuzzy system:

- the system is automatically designed as a whole by means of GA techniques, providing a way to decide on : membership function parameters (type, shape and location), overlapping between fuzzy sets, relation defined by each individual rule, and the number of rules of the resulting fuzzy model.
- the evolutionary process is structured in a hierarchical configuration composed of the following modules: partition set, individual fuzzy rules, and sets of fuzzy rules.

The basic idea is to develop a fuzzy system as composed of modules (distinct populations) that interacts during the evolutionary process. The objective is to obtain, at the final generation, the best set of fuzzy rules and the associated model parameters. The modular and hierarchical structure is depicted in figure 1. The partition set provides the atomic elements (statements or membership functions) to be combined and used to produce a population of individual rules (granules). The population of individual rules produces, in turn, a population of sets of rules (fuzzy graphs).

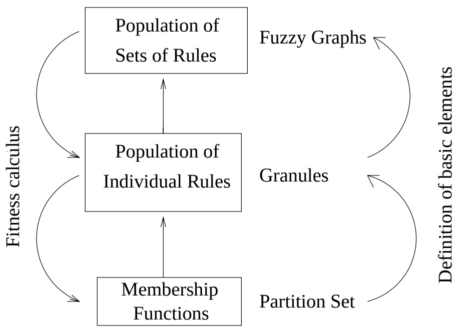

Population of Sets of Rules
Fuzzy Graphs
Population of Individual Rules
Granules
Membership Functions
Partition Set
Fitness calculus
Definition of basic elements

Figure 1: Hierarquical relation between different modules

Function approximation is used as a test problem to verify the efficiency of the proposed method. Learning an input-output compact mapping from a set of examples can be regarded as synthesizing an approximation of an unknown multidimensional function $G(.) : \Omega \subset \Re^m \rightarrow \Re^r$, with Ω a compact region. For $l = 1, ..., N$, let $\mathbf{x}_l \in \Re^m$ be the input variable, $\mathbf{s}_l \in \Re^r$ be the respective output, and $\varepsilon \in \Re^r$ be an additive random variable, with zero mean and known variance. A sampling process given by $\mathbf{s}_l = \mathbf{G}(\mathbf{x}_l)+\varepsilon, l = 1, ..., N$ produces an input matrix $\mathbf{X} \in \Re^{N\mathrm{x}m}$ and a corresponding output matrix $\mathbf{S} \in \Re^{N\mathrm{x}m}$, with $\mathbf{x}_l^T$ and $\mathbf{s}_l^T$, $(l = 1, ..., N)$ as their rows. The goal is to use the matrices $\mathbf{X}$ and $\mathbf{S}$ to construct the best approximation model $\hat{\mathbf{G}}(.)$ and adopt this model to obtain an estimate $\hat{\mathbf{s}}$, given any $\mathbf{x} \in \mathbf{\Omega}$, such that $\hat{\mathbf{s}} = \hat{\mathbf{G}}(\mathbf{x})$. The numerical example to be considered assumes $m = 2$, $r = 1$, $N = 225$ and $\Omega = [-4, 4]\mathrm{x}[-4, 4]$. Figure 8(a) shows the function to be approximated. Notice that this problem of function approximation represents a generic description of the mathematical formulation for:

- control problem: the input-output mapping represents the control surface to be approximated;
- pattern recognition problem: the input-output model represents the discriminant surface to be approximated.

The fuzzy relations associated with each rule are assumed to be fuzzy conjunctions. Therefore, **each individual rule** is a **granule** and the fuzzy relation, defined by a **specific set of rules**, is a **fuzzy graph** [Pedrycz and Gomide, 1998]. Figures 2 and 3 show the ideas of a granule and fuzzy graph, in the context of function approximation.

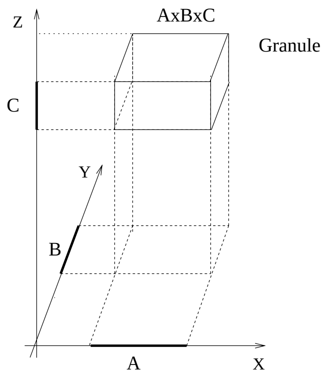


Figure 2: Fuzzy Point or Granule

Each granule represents only one fuzzy rule that relates antecedent and consequent parts. In function approximation, the antecedent has the variables x and y ($m = 2$) and the consequent has the variable z ($r = 1$). The i^{th} rule R_i is defined by: R_i: IF x is A_i AND y is B_i THEN z is C_i, and the set of rules $[R_1...R_n]$ represents a fuzzy graph (fig. 3) in $\Re^3$ space.

The partition set, formed by only one chromosome,

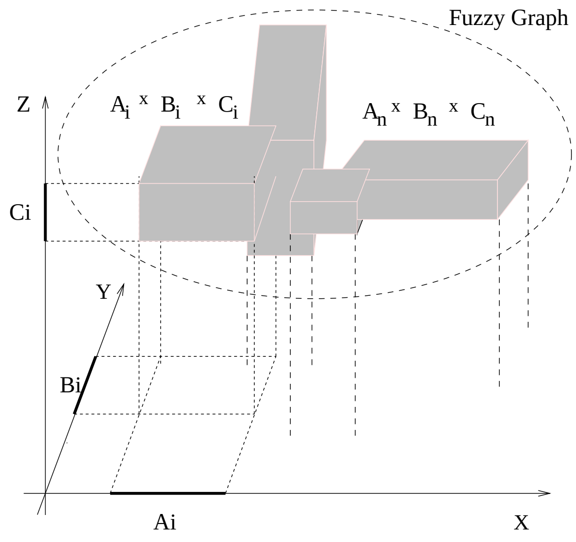


Figure 3: Fuzzy Graph

evolves by means of simple mutation and its modification influences granules and graphs. Each individual in the population of granules represents a proposition. So, the population of granules will produce different arrangements of linguistic terms according to the variables involved. The population of graphs aggregates individuals that render specific arrays of rules. The fitness measure of an individual graph is calculated based on the MSE produced by the presentation of the input data set. The fitness of each granule is based on the mean values of the fitness associated with the graphs in which it takes part. Updating the fitness associated with the population of graphs results in modifications on the fitness of the population of granules and so for.

The evolutionary process uses generic purpose operations for mutation and crossover, and specific purpose mutation operations as well. Generic purpose operations employ genetic material chosen by the roulette wheel (RW) procedure. In this case, the selection is elitist: the individuals with worst fitness are replaced by the offsprings generated. In specific purpose operations, parents are selected from a group of best individuals and replacements are performed on each individual chromosome. Hence, generic purpose operators simulate the standard process of classic evolution. Specific purpose mutation changes the original chromosome and the process of adaptation occurs along the life of each chromosome. In spite of the type of operation (generic or specific purpose) used, in the mechanism of mutation the value of a chosen allele is changed by another value taken from a set of valid values. Specifically, in granules and graphs “bad values"(elements with bad fitness measures) may be substituted for "good values". Uniform crossover is used to manipulate the population of granules, and simple crossover to manipulate the population of graphs [Michalewicz, 1996]. Details concerning the implementation of the algorithm will be given in section 3.

The hierarchical method enables the fitness storage of the population to be performed separately in each module. Therefore, the replacement of granules depends on the mean performance, not on the current performance. Thus, it reduces the risk of suppressing a granule only because it takes part on an upper-level population element (graph) with a bad performance.

The details of the structural codification are discussed in what follows. All linguistic terms (statements) are represented by membership functions encoded by strings (genes) of 5 elements. Here, each position (allele) represents different data of a specific gene or membership function. Figure 4 provides an example of the genotype and phenotype of a membership function.

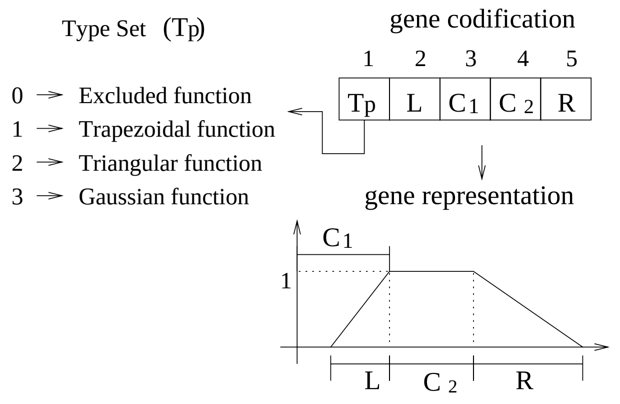


Figure 4: Membership function codified as a gene

In figure 4 the value of allele 1 indicates the type of membership function encoded: trapezoidal, triangular, and Gaussian functions. Strings or genes of type 0 (excluded function) indicates that the associated linguistic term is not able to appear in any rule of the whole set of rules. It may be observed that, except the value of allele 1, all the remaining values are relative to the pivot element C_1, which in turn depends on the respective origin of the universe. Once defined the string, a decoding process is required to generate absolute parameters for the membership functions. For trapezoidal functions, only one step is necessary: alleles 2 to 5 are converted to their absolutes values. For triangular and Gaussian functions some additional steps are necessary. In the case of triangular functions, the center is achieved by the average of C_1 and C_2, and the remaining alleles are just converted to their absolute values. For Gaussian function, the mean is calculated in the same way as in the triangular case but another modification is required to calculate standard deviation: maximum deviation is given by the following formula: $\Delta = (L + R)/2$, and standard deviation is given by $\sigma = \Delta/3$, so that the function limits will not exceed 3σ.

The partition set, codified in a single chromosome, is formed by linking all the possible membership functions (encoded according to the above description) defined in the associated universes. At the population of granules, each individual (rule) that aggregates antecedents and consequents is represented by a collection of integers (indexes concerning terms A_i, B_i and C_i, for the case of two input variables and one output variable) representing membership functions that take part in each rule. The population of graphs is also defined by a set of chromosomes, where each allele is an index i that identify the granule (rule) which takes part in that graph.

Tables 1, 2 and 3 show an example of the genotype configuration for individuals belonging to the three modules that compose the system. This is only an illustrative example, in the real implementation to be presented in section3, larger population sizes are considered.

Table 1: Genotype Configuration for three Fuzzy Graphs

Systems					
Graph	Granules or Rules				
1	1	7	3	2	5
2	6	7	0	9	10
3	10	6	4	6	9

Table 2: Genotype Configuration for ten Granules

Individual Rules										
	Rules' Indexes									
Term	1	2	3	4	5	6	7	8	9	10
A_i	1	1	3	0	4	2	3	2	4	2
B_i	3	2	3	3	1	0	1	3	2	3
C_i	4	1	4	2	1	4	2	3	4	1

Null elements indicate parameter exclusion, adopted to produce three levels of compression: exclusion of a

Table 3: Genotype Configuration for the Partition Set (with four membership functions associated with each variable)

VAR	Ling. Term	T_p	L	C_1	C_2	R
x	1	1.0	2.7	1.3	0.5	2.7
	2	3.0	0.6	4.0	0.1	3.1
	3	2.0	2.7	5.0	0.4	2.8
	4	1.0	2.7	8.0	0.5	2.7
y	1	1.0	2.4	1.0	0.8	2.7
	2	3.0	2.2	4.0	0.5	2.7
	3	1.0	2.7	6.7	0.5	2.7
	4	0.0	2.7	9.3	0.5	2.7
z	1	1.0	2.2	1.1	0.4	2.2
	2	1.0	2.2	3.3	0.4	2.2
	3	0.0	1.2	5.4	0.4	2.2
	4	1.0	2.2	6.0	0.4	2.2

membership function, variable elimination (indicating irrelevance of the corresponding variable in a specific individual rule), and exclusion of a rule. The total of relevant elements in each chromosome is computed in a different way according to its population level. For the population of fuzzy graphs, the number of rules in each graph is determined by computing the total of elements different from the null one and with the subtraction of repeated indexes. For example, at Table 1 the graph 1 is a full graph with 5 rules (no elimination or repetition), graph 2 has a total of 4 rules (one rule has been eliminated) and graph 3 is formed by 4 rules (rule 6 appears twice). The variable elimination is obtained by setting an index of the antecedent part term to zero. As we can see in Table 2, the 4th rule indicates the irrelevance of the variable **x** in this rule. So we have a compressed rule of the form: *If* **y** is "3" *then* **z** is "2". Finally, at partition set, we have a kind of compression that permits the elimination of a specific linguistic term from the partition set. As shown in Table 3, the linguistic terms 4 and 3 are not able to appear in the partition set of universes Y and Z, respectively. Notice that exclusion in lower levels determines indirect elimination in higher levels. For examples, if a specific term has been excluded from the partition set, all the rules that refer to it perform as compressed rules, although the index has not been changed to zero. In the case of granules, if the antecedent part of a specific rule has null values, this rule has no effect in any graph although its index remain fixed in the graph population.

Figure 5 illustrates the details concerning graph 1 of Table 1.

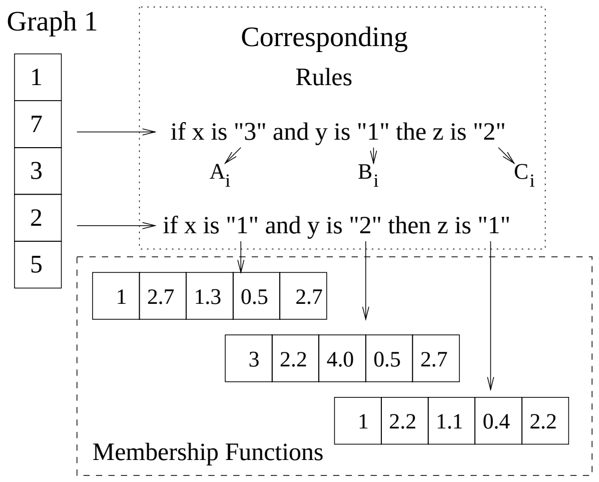


Figure 5: Hierarchical Relation Between Modules in the Fuzzy System

Figure 6 shows the phenotype of genes or linguistic terms presented in the locus of variable y, in the partition set illustrated in Table 2. It should be noted that the linguistic term "4", corresponding to the most right term is not presented since it is an excluded function.

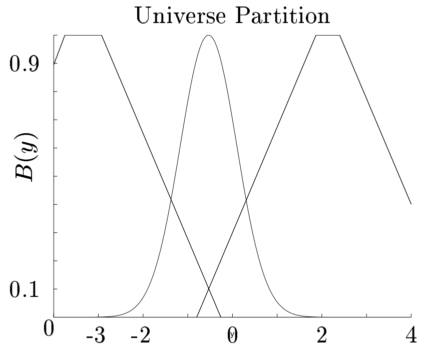


Figure 6: Partition Decoding for Variables y in the example of Table 3

3 IMPLEMENTATION AND RESULTS

The evolutionary algorithm uses an intermediate method between the Michigan and the Pitt approach. The final solution is made up from some elements (memberhip functions) from the partition set (neither the whole set, nor an individual element), some elements from the population of granules, that will join to form an individual of the population of fuzzy graphs.

The algorithm presented bellow summarize the main steps.

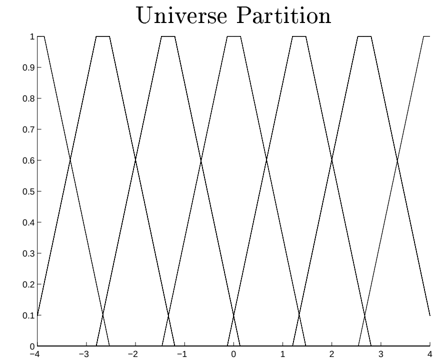
```
         ALGORITHM

1. randomly  inicialize populations
2. Eval Fitness of all modules
WHILE Best Graph Fit < F  AND  Generation < G

  set operation = specific (ind = s)
     execute steps (3.1, 4.1 and 5.1)
  set operation = generic (ind = g)
     execute steps (3, 4 and 5)
  execute step 6

  3. operations over Partition Set
    3.1 Partiton set mutation
      a) choose membership fun.(MF)
      b) SubPop = mutation(MF)
      c) MedPopPart = POPPart - MF + SubPop
      d) Eval fitness of Graphs for MedPopPart
      e) IF ExecutionConditions == OK
            NewPopPart = MedPopPart
            Update Fitness of all modules

  4. operations over Granules Population
    4.1 Granules mutation
      a) choose set of rules (RL)
      b) SubPop = mutation(RL)
      c) MedGn(s) = POPGn - RL + SubPop
         MedGn(g) = POPGn - worst_rules + SubPop
      d) Eval fitness of Graphs for MedGn(ind)
      e) IF ExecutionConditions == OK
            NewPopGn = MedGn(ind)
            Update Fitness of all modules

    4.2 granules crossover
      a) choose rules(RL1,RL2)
      b) SubPop = crossover(RL1,RL2)
      c) MedGn = POPGn - worst_rules + SubPop
      d) Eval fitness of Graphs for MedGn
      e) IF ExecutionConditions == OK
            NewPopGn = MedGn
            Update Fitness of all modules

  5. operations over Graphs Population
    5.1 Graph mutation
         idem 4.1
    5.2 Graph crossover
         idem 4.2

  6. Generation = Generation + 1

END
```

As show in the algorithm, the first step performed in each generation is the operation of specific purpose mutation over the partition set and the population of granules and graphs. This type of search looks for individuals with the highest fitness measure. After that, the algorithm performs a less specific search using generic purpose operations. This type of search looks for better offsprings, generated from individuals selected by roulette wheel. It is important to note that, in spite of the type of operations performed (specific or generic), modifications are only carried out if some conditions of execution are satisfied. There are two types of conditions: condition type 1, where the best graph does not degenerate (the fitness of the new best graph is equal or higher then the fitness of the old best graph); and condition type 2, where the fitness average of the involved individuals does not degenerate.

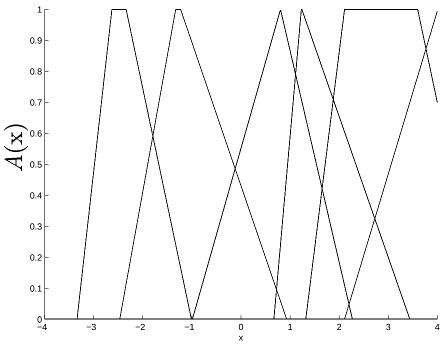
Several tests have been performed, fixing some parameters and varying others, such as population size, initial granularity, probabilities, and the type of execution test. Best results have been achieved with the following initial parameters:

- population size: 80 graphs, with a maximum of 35 rules in each graph, 340 granules and 7 linguistic terms for each variable, all of the same type and uniformly distributed over the corresponding universe.
- probabilities associated with the genetic operators:
  1. specific purpose mutation: graph(0.01), granules(0.05), partition set(0.07).
  2. generic purpose mutation: graph(0.08), granules(0.08)
  3. crossover: graphs(0.2), granules(0.1).
- execution test: condition 1 and 2 applied together
- compression taxes: graph(0,2), granules(0), partition set(0).
- inference mechanism: 1) Antecedent aggregation operator: $AND = min$, 2) Meaning of each rule:min (Mamdani Conjunction function) 3) Rule Aggregation: max 4) Decoding Process (Defuzzyfication): CoG. The defuzzyfied output is used to calculate the MSE for each graph, and the fitness measure is: $fit = 1/MSE$.

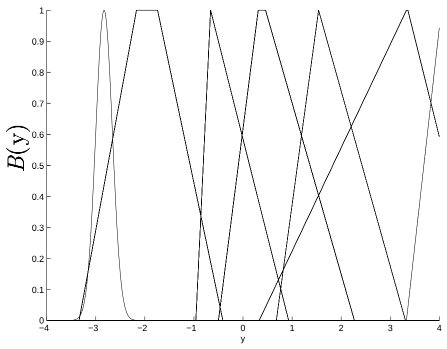
At the last generation the best individual results in a graph built up from 30 fuzzy rules based on a partition set illustrated in Figure 7, that brings a comparison of the partitions initially defined to the final partitions obtained. Figure 8 illustrates the approximation performance. Figure 8(a) shows the original function to be approximated, figure 8(b) illustrates the final surface obtained at the last generation, and figure 8(c) shows the date evolution along generations. Since the best graph is preserved or improved from generation to generation, the fitness measure never decreases.



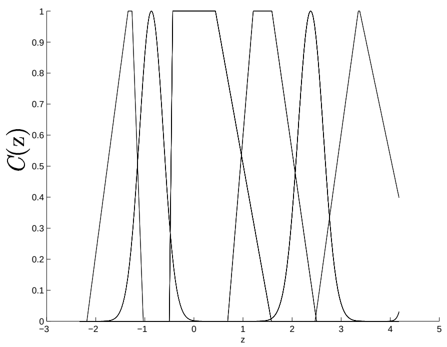
(a)



(b)

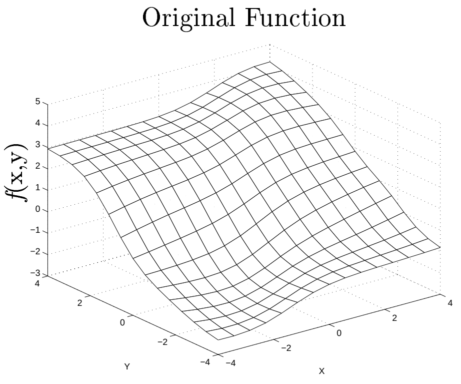


(c)

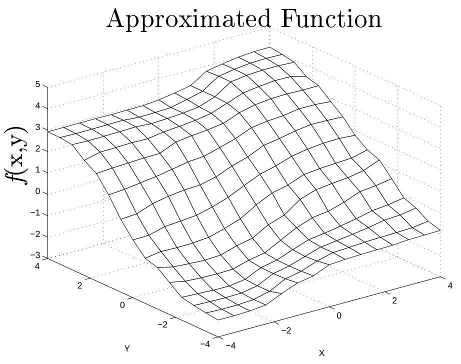


(d)

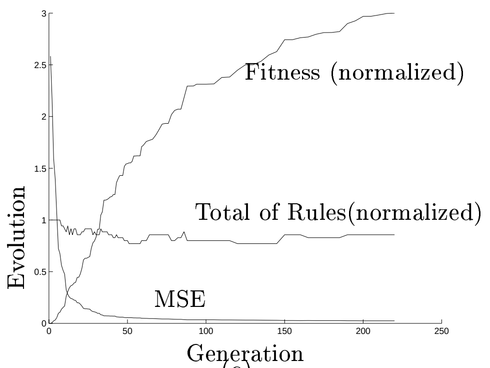
Figure 7: Partition Decoding, (a)Initial Partition (b,c,d)Final Partition Set for X,Y and Z Universes, respectively



(a)



(b)



(c)

Figure 8: (a)Original Function, (b)Final Approximation, (c) MSE, Fitness and Compression Tax of Best Chromosome in the Graph Population (Best Set of Rules)

4 CONCLUSIONS

The paper has introduced a modular, hierarchical evolutionary method to design fuzzy systems using genetic algorithms to evolve a population of rule-based fuzzy models. The method solved a function approximation example to demonstrate its potential in system modeling problems. Simulation results show that the method provides a computationally tractable procedure to design fuzzy systems. Specific purpose operators are implemented to control population diversity along generations.

The hierarchical structure is a fundamental aspect of the design because it provides an effective way to define the fitness of low-level modules. That is, the fitness value measures the potential of a module to contribute to the final solution, not the current performance of the module. Therefore, if a high-level module has a poor performance as a candidate to represent a fuzzy model, this does not necessarily mean a bad fitness for all the rules involved, because they may be part of other high-level modules of better performance. This is a form of evolutionary memory. A low-level module has a small (high) probability of survival if its performance, in average, decreases (increases). Despite its potential to deal with systems of high complexity, the method can obviously be useful to refine existing solutions to simpler engineering problems.

Acknowledgments

Myriam Delgado acknowledges CEFET/PR and CAPES/PICDT, Fernando Von Zuben the CNPq grant 300910/96-7, and Fernando Gomide the CNPq grant 300729/86-3 for their support.

References

D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms motivation analysis and first results. *Complex Systems*, 3:493–530, 1989.

F. Hoffmann and G. Pfister. A new learning method for the design of hierarchical fuzzy controllers using messy genetic algorithms. In *Proceedings of the 6th IFSA World Congress*, volume 1, pages 249–252, São Paulo, Brazil, 1995.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 2nd edition, 1992.

A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controller using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2):129–139, 1995.

C. L. Karr. Applying genetics to fuzzy logic. *AI Expert*, 6(3):38–43, 1991.

C. L. Karr and E. J. Gentry. Fuzzy control of ph using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 1(1):46–53, 1993.

G. Klir and T. Forger. *Fuzzy Sets, Uncertainty and Information*. Prentice Hall, 1988.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

D. E. Moriarty and R. Miikkulainen. Hierarchical evolution of neural networks. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 428 – 433, Alaska, 1998.

T. Murata, H. Ishibuchi, and M. Gen. Adjusting fuzzy partitions by genetic algorithms and histograms for pattern classification problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 9 – 14, Alaska, 1998.

H. Nomura, I. Hayashi, and N. Wakami. A self–tuning method of fuzzy reasoning by genetic algorithm. In *Proceedings of the 1992 International Fuzzy Systems and Intelligent Control Conference*, pages 236 – 245, Louisville, USA, 1992.

W. Pedrycz and F. Gomide. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, 1998.

H. Takagi and M. Lee. Neural networks and genetic algorithms approaches to auto design of fuzzy systems. In *Proceedings of Fuzzy Logic in Artificial Intelligence*, Linz, Austria, 1993.

P. Thrift. Fuzzy logic synthesis with genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 509 – 513, 1991.

M. Valenzuela-Rendon. The fuzzy classifier system: A classifier system for continuously varying variables. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 346 – 353, 1991.

R. R. Yager. On a hierarchical structure for fuzzy modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1189–1197, 1993.