Bio-inspired Optimization of Interval Type-2 Fuzzy Controllers

Oscar Castillo and Patricia Melin

Tijuana Institute of Technology, Division of Graduate Studies,
Calzada Tecnologico s/n, 22379, Tijuana, Mexico
{ocastillo,pmelin}@tectijuana.mx

Abstract. A review of the optimization methods used in the design of type-2 fuzzy systems, which are relatively novel models of imprecision, has been considered in this paper. The fundamental focus of the work has been based on the basic reasons of the need for optimizing type-2 fuzzy systems for different areas of application. Recently, bio-inspired methods have emerged as powerful optimization algorithms for solving complex problems. In the case of designing type-2 fuzzy systems for particular applications, the use of bio-inspired optimization methods have helped in the complex task of finding the appropriate parameter values and structure of the fuzzy systems. In this paper, we consider the application of genetic algorithms, particle swarm optimization and ant colony optimization as three different paradigms that help in the design of optimal type-2 fuzzy systems. We also provide a comparison of the different optimization methods for the case of designing type-2 fuzzy systems.

Keywords: Intelligent Control, Type-2 Fuzzy Logic, Interval Fuzzy Logic.

1 Introduction

Uncertainty affects decision-making and appears in a number of different forms. The concept of information is fully connected with the concept of uncertainty [17]. The most fundamental aspect of this connection is that the uncertainty involved in any problem-solving situation is a result of some information deficiency, which may be incomplete, imprecise, fragmentary, not fully reliable, vague, contradictory, or deficient in some other way. Uncertainty is an attribute of information [24]. The general framework of fuzzy reasoning allows handling much of this uncertainty and fuzzy systems that employ type-1 fuzzy sets represent uncertainty by numbers in the range [0, 1]. When something is uncertain, like a measurement, it is difficult to determine its exact value, and of course type-1 fuzzy sets make more sense than using crisp sets [14]. However, it is not reasonable to use an accurate membership function for something uncertain, so in this case what we need is higher order fuzzy sets, those which are able to handle these uncertainties, like the so called type-2 fuzzy sets [14]. So, the amount of uncertainty can be managed by using type-2 fuzzy logic because it offers better capabilities to handle linguistic uncertainties by modeling vagueness and unreliability of information [5] [23].

Recently, we have seen the use of type-2 fuzzy sets in Fuzzy Logic Systems (FLS) in different areas of application [1] [2] [6] [10] [12]. In this paper we deal with the application of interval type-2 fuzzy control to non-linear dynamic systems [3] [4] [5] [15] [19]. It is a well known fact, that in the control of real systems, the instrumentation elements (instrumentation amplifier, sensors, digital to analog, analog to digital converters, etc.) introduce some sort of unpredictable values in the information that has been collected [20]. So, the controllers designed under idealized conditions tend to behave in an inappropriate manner [11].

2 Fuzzy Logic Systems

In this section, a brief overview of type-1 and type-2 fuzzy systems is presented. This overview is considered to be necessary to understand the basic concepts needed to develop the methods and algorithms presented later in the paper.

2.1 Type-1 Fuzzy Logic Systems

Soft computing techniques have become an important research topic, which can be applied in the design of intelligent controllers, which utilize the human experience in a more natural form than the conventional mathematical approach [16, 18]. A FLS, described completely in terms of type-1 fuzzy sets is called a type-1 fuzzy logic system (type-1 FLS). In this paper, the fuzzy controller has two input variables, which are the error $e(t)$ and the error variation $\Delta e(t)$,

$$e(t) = r(t) - y(t) \tag{1}$$

$$\Delta e(t) = e(t) - e(t-1) \tag{2}$$

so the control system can be represented as in Figure 1.

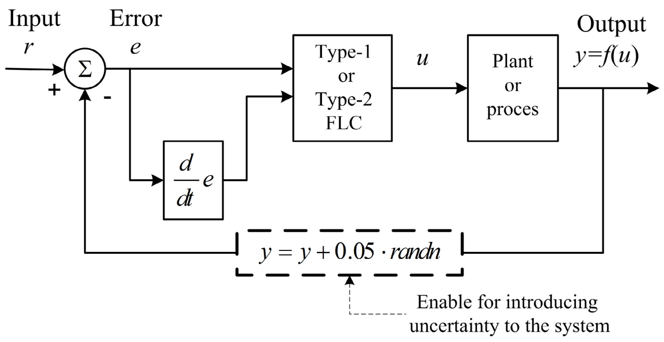

Fig. 1. System used for obtaining the experimental results

2.2 Type-2 Fuzzy Logic Systems

If for a type-1 membership function, as in Figure 2, we blur it to the left and to the right, as illustrated in Figure 3, then a type-2 membership function is obtained. In this case, for a specific value x', the membership function (u'), takes on different values, which are not all weighted the same, so we can assign an amplitude distribution to all of those points.

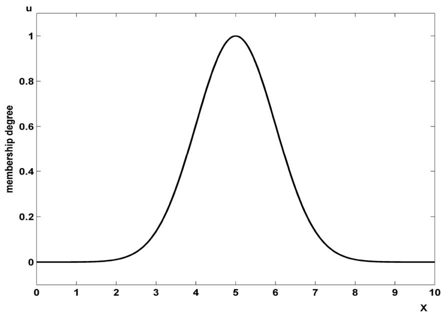

Fig. 2. Type-1 membership function

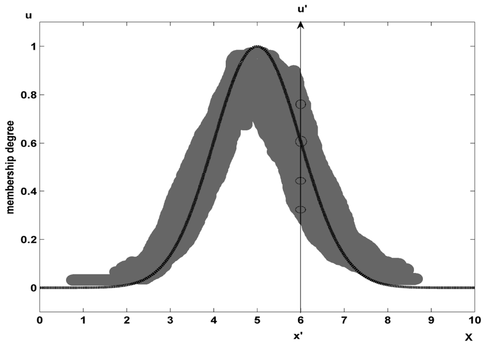

Fig. 3. Blurred type-1 membership function

A type-2 fuzzy set $\tilde{A}$, is characterized by the membership function [14, 17]:

$$\tilde{A} = \left\{ \left((x,u), \mu_{\tilde{A}}(x,u) \right) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \right\} \tag{3}$$

in which $0 \le \mu_{\tilde{A}}(x,u) \le 1$. Another expression for $\tilde{A}$ is,

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u) / (x,u) \qquad J_x \subseteq [0,1] \tag{4}$$

Where $\int\int$ denotes the union over all admissible input variables x and u. For discrete universes of discourse $\int$ is replaced by $\sum$. In fact $J_x \subseteq [0,1]$ represents the primary membership of *x*, and $\mu_{\tilde{A}}(x,u)$ is a type-1 fuzzy set known as the secondary set. Hence, a type-2 membership grade can be any subset in [0,1], the primary membership, and corresponding to each primary membership, there is a secondary membership (which can also be in [0,1]) that defines the possibilities for the primary membership. Uncertainty is represented by a region, which is called the footprint of uncertainty (FOU). When $\mu_{\tilde{A}}(x,u)=1, \forall u \in J_x \subseteq [0,1]$ we have an interval type-2 membership function, as shown in Figure 4. The uniform shading for the FOU represents the entire interval type-2 fuzzy set and it can be described in terms of an upper membership function $\bar{\mu}_{\tilde{A}}(x)$ and a lower membership function $\underline{\mu}_{\tilde{A}}(x)$.

A FLS described using at least one type-2 fuzzy set is called a type-2 FLS. Type-1 FLSs are unable to directly handle rule uncertainties, because they use type-1 fuzzy sets that are certain [14]. On the other hand, type-2 FLSs, are very useful in circumstances where it is difficult to determine an exact membership function, and there are measurement uncertainties [7, 8, 15].

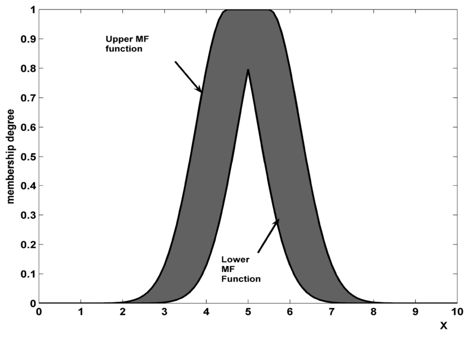

Fig. 4. Interval type-2 membership function

A type-2 FLS is again characterized by IF-THEN rules, but its antecedent or consequent sets are now of type-2. Similar to a type-1 FLS, a type-2 FLS includes a fuzzifier, a rule base, fuzzy inference engine, and an output processor, as we can see in Figure 5. The output processor includes type-reducer and defuzzifier; it generates a type-1 fuzzy set output (type-reducer) or a crisp number (defuzzifier).

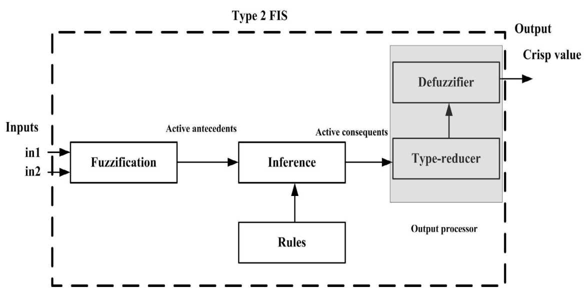

Fig. 5. Type-2 Fuzzy Logic System

2.2.1 Fuzzifier

The fuzzifier maps a crisp point $\mathbf{x}=(x_1,\ldots,x_p)^T \in X_1 x X_2 x \ldots x X_p \equiv \mathbf{X}$ into a type-2 fuzzy set $\tilde{A}_x$ in **X** [17], interval type-2 fuzzy sets in this case. We will use type-2 singleton fuzzifier, in a singleton fuzzification, the input fuzzy set has only a single point on nonzero membership [14]. $\tilde{A}_x$ is a type-2 fuzzy singleton if $\mu_{\tilde{A}_x}(x) = 1/1$ for **x=x'** and $\mu_{\tilde{A}_x}(x) = 1/0$ for all other **x≠x'**[17].

2.2.2 Rules

The structure of rules in a type-1 FLS and a type-2 FLS is the same, but in the latter the antecedents and the consequents will be represented by type-2 fuzzy sets. So for a type-2 FLS with p inputs $x_1 \in X_1,\ldots,x_p \in X_p$ and one output $y \in Y$, Multiple Input Single Output (MISO), if we assume there are M rules, the lth rule in the type-2 FLS can be written as follows [14]:

$$R^l\text{: IF } x_1 \text{ is } \tilde{F}_1^l \text{ and } \cdots \text{ and } x_p \text{ is } \tilde{F}_p^l \text{, THEN } y \text{ is } \tilde{G}^l \qquad l=1,\ldots,M \tag{5}$$

2.2.3 Inference

In the type-2 FLS, the inference engine combines rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. It is necessary to compute the join $\sqcup$, (unions) and the meet $\sqcap$ (intersections), as well as extended sup-star compositions (sup star compositions) of type-2 relations [14]. If $\tilde{F}^l{}_1 \times \cdots \times \tilde{F}^l{}_p = \tilde{A}^l$, equation (5) can be re-written as

$$R^l : \tilde{F}^l{}_1 \times \cdots \times \tilde{F}^l{}_p \rightarrow \tilde{G}^l = \tilde{A}^l \rightarrow \tilde{G}^l \qquad l=1,\ldots,M \tag{6}$$

R^l is described by the membership function $\mu_{R^l}(\mathbf{x}, y) = \mu_{R^l}(x_1,\ldots,x_p, y)$, where

$$\mu_{R^l}(\mathbf{x}, y) = \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y) \tag{7}$$

can be written as [14]:

$$\mu_{R^l}(\mathbf{x}, y) = \mu_{\tilde{A}^l \rightarrow \tilde{G}^l}(\mathbf{x}, y) = \mu_{\tilde{F}_1^l}(x_1) \sqcap \cdots \sqcap \mu_{\tilde{F}_p{}^l}(x_p) \sqcap \mu_{\tilde{G}^l}(y)$$

$$= [\sqcap_{i=1}^{p} \mu_{\tilde{F}^l{}_i}(x_i)] \sqcap \mu_{\tilde{G}^l}(y) \tag{8}$$

In general, the p-dimensional input to R^l is given by the type-2 fuzzy set $\tilde{A}_x$ whose membership function is

$$\mu_{\tilde{A}_x}(\mathbf{x}) = \mu_{\tilde{x}_1}(x_1) \sqcap \cdots \sqcap \mu_{\tilde{x}p}(x_p) = \sqcap_{i=1}^{p} \mu_{\tilde{x}i}(x_i) \tag{9}$$

where $\tilde{X}_i (i = 1,\ldots, p)$ are the labels of the fuzzy sets describing the inputs. Each rule R^l determines a type-2 fuzzy set $\tilde{B}^l = \tilde{A}_x \circ R^l$ such that [14]:

$$\mu_{\tilde{B}^l}(y) = \mu_{\tilde{A}_x \circ R^l} = \sqcup_{x \in \mathbf{X}} \left[\mu_{\tilde{A}_x}(\mathbf{x}) \sqcap \mu_{R^l}(\mathbf{x}, y) \right] \qquad y \in Y \; l=1,\ldots,M \tag{10}$$

This equation is the input/output relation in Figure 5 between the type-2 fuzzy set that activates one rule in the inference engine and the type-2 fuzzy set at the output of that engine [14]. In the FLS we used interval type-2 fuzzy sets and meet under product t-norm, so the result of the input and antecedent operations, which are contained in the firing set $\sqcap_{i=1}^{p} \mu_{\tilde{F}_{i_i}}(x'_i \equiv F^l(\mathbf{x}')$, is an interval type-1 set [14],

$$F^l(\mathbf{x}') = \left[\underline{f}^l(\mathbf{x}'), \overline{f}^l(\mathbf{x}') \right] \equiv \left[\underline{f}^l, \overline{f}^l \right] \tag{11}$$

where

$$\underline{f}^l(\mathbf{x}') = \underline{\mu}_{\tilde{F}_1^l}(x_1') * \cdots * \underline{\mu}_{\tilde{F}_p^l}(x_p') \tag{12}$$

$$\overline{f}^l(\mathbf{x}') = \overline{\mu}_{\tilde{F}_1^l}(x_1') * \cdots * \overline{\mu}_{\tilde{F}_p^l}(x_p') \tag{13}$$

where * is the product operation.

2.2.4 Type Reducer

The type-reducer generates a type-1 fuzzy set output, which is then converted in a crisp output through the defuzzifier. This type-1 fuzzy set is also an interval set, for the case of our FLS we used center of sets (cos) type reduction, $Y_{\cos}$ which is expressed as [14]:

$$Y_{\cos}(\mathbf{x}) = [y_l, y_r] = \int_{y^1 \in [y_l^1, y_r^1]} \cdots \int_{y^M \in [y_l^M, y_r^M]} \int_{f^1 \in [\underline{f}^1, \overline{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \overline{f}^M]} 1 \Big/ \frac{\sum_{i=1}^{M} f^i y^i}{\sum_{i=1}^{M} f^i} \tag{14}$$

this interval set is determined by its two end points, y_l and y_r, which corresponds to the centroid of the type-2 interval consequent set $\tilde{G}^i$ [14],

$$C_{\tilde{G}^i} = \int_{\theta_1 \in J_{y1}} \cdots \int_{\theta_N \in J_{yN}} 1 \Big/ \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i} = [y_l^i, y_r^i] \tag{15}$$

before the computation of $Y_{\cos}(\mathbf{x})$, we must evaluate equation (15), and its two end points, y_l and y_r. If the values of f_i and y_i that are associated with y_l are denoted f_l^i and y_l^i, respectively, and the values of f_i and y_i that are associated with y_r are denoted f_r^i and y_r^i, respectively, from (14), we have [14]

$$y_l = \frac{\sum_{i=1}^{M} f_l^i y_l^i}{\sum_{i=1}^{M} f_l^i} \tag{16}$$

$$y_r = \frac{\sum_{i=1}^{M} f_r^i y_r^i}{\sum_{i=1}^{M} f_r^i} \tag{17}$$

2.2.5 Defuzzifier

From the type-reducer we obtain an interval set $Y_{\cos}$, to defuzzify it we use the average of y_l and y_r, so the defuzzified output of an interval singleton type-2 FLS is [14]

$$y(\mathbf{x}) = \frac{y_l + y_r}{2} \tag{18}$$

3 Bio-inspired Optimization Methods

In this section a brief overview of the basic concepts from bio-inspired optimization methods needed for this work is presented.

3.1 Particle Swarm Optimization

Particle swarm optimization is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling [1]. PSO shares many similarities with evolutionary computation techniques such as the GA [9].

The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In the PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [16]. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest* [19].

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations [1]. In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way when compared with other methods [19]. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been considered for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

The basic algorithm of PSO has the following nomenclature:

x_z^i	-Particle position
v_z^i	-Particle velocity

w_{ij}	-Inertia weight
p_z^i	-Best "remembered" individual particle position
p_z^g	-Best "remembered" swarm position
c_1, c_2	-Cognitive and Social parameters
r_1, r_2	-Random numbers between 0 and 1

The equation to calculate the velocity is:

$$v_{z+1}^i = w_{ij} v_z^i + c_1 r_1 \left(p_z^i - x_z^i \right) + c_2 r_2 \left(p_z^g - x_z^i \right) \tag{19}$$

and the position of the individual particles is updated as follows:

$$x_{z+1}^i = x_z^i + v_{z+1}^i \tag{20}$$

The basic PSO algorithm is defined as follows:

```
1) Initialize
    a) Set constants $z_{\max}, c_1, c_2$
    b) Randomly initialize particle position $x_0^i \in D$ in $R^n$ for $i = 1,..., p$
    c) Randomly initialize particle velocities $0 \le v_0^i \le v_0^{\max}$ for $i = 1,..., p$
    d) Set Z = 1
2) Optimize
    a) Evaluate function value $f_k^i$ using design space coordinates $x_k^i$
    b) If $f_z^i \le f_{best}^i$ then $f_{best}^i = f_z^i, p_z^i = x_z^i$.
    c) If $f_z^i \le f_{best}^g$ then $f_{best}^g = f_z^i, p_z^g = x_z^i$.
    d) If stopping condition is satisfied then go to 3.
    e) Update all particle velocities $v_z^i$ for $i = 1,..., p$
    f) Update al particle positions $x_z^i$ for $i = 1,..., p$
    g) Increment z.
    h) Goto 2(a).
3) Terminate
```

3.2 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetic processes [8]. The basic principles of GAs were first proposed by John Holland in 1975, inspired by the mechanism of natural selection, where stronger individuals are likely the winners in a competing environment [9]. GA assumes that the potential solution of any problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form. A positive value, generally known as a fitness value, is used to reflect the degree of "goodness" of the chromosome for the problem, which would be highly related with its objective value. The pseudocode of a GA is as follows:

```
1. Start with a randomly generated population of n chromosomes (candidate
   solutions to a problem).
2. Calculate the fitness of each chromosome in the population.
3. Repeat the following steps until n offspring have been created:
      a. Select a pair of parent chromosomes from the current population,
         the probability of selection being an increasing function of fitness.
         Selection is done with replacement, meaning that the same chromo-
         some can be selected more than once to become a parent.
      b. With probability (crossover rate), perform crossover to the pair at a
         randomly chosen point to a form two offspring.
      c. Mutate the two offspring at each locus with probability (mutation
         rate), and place the resulting chromosomes in the new population.
4. Replace the current population with the new population.
5. Go to step 2.
```

The simple procedure just described above is the basis for most applications of GAs found in the literature [21] [22].

3.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is a probabilistic technique that can be used for solving problems that can be reduced to finding good paths along graphs. This method is inspired on the behavior presented by ants in finding paths from the nest or colony to the food source.

The S-ACO is an algorithmic implementation that adapts the behavior of real ants to solutions of minimum cost path problems on graphs [12]. A number of artificial ants build solutions for a certain optimization problem and exchange information about the quality of these solutions making allusion to the communication system of real ants [13].

Let us define the graph $G = (V, E)$, where V is the set of nodes and E is the matrix of the links between nodes. G has $n_G = |V|$ nodes. Let us define L^K as the number of hops in the path built by the ant k from the origin node to the destiny node. Therefore, it is necessary to find:

$$Q = \left\{ q_a, \ldots, q_f \middle| q_1 \in C \right\} \tag{21}$$

where Q is the set of nodes representing a continuous path with no obstacles; $q_a, \ldots, q_f$ are former nodes of the path and C is the set of possible configurations of the free space. If $x^k(t)$ denotes a Q solution in time t, $f(x^k(t))$ expresses the quality of the solution. The S-ACO algorithm is based on Equations (22), (23) and (24):

$$p_{ij}^k(t) = \begin{cases} \dfrac{\tau_{ij}^k}{\sum_{j \in N_{ij}^k} \tau_{ij}^{\alpha}(t)} & if \;\; j \in N_i^k \\ 0 & if \;\; j \notin N_i^k \end{cases} \tag{22}$$

$$\tau_{ij}(t) \leftarrow (1\text{-}\rho)\tau_{ij}(t) \tag{23}$$

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \tau_{ij}(t) \tag{24}$$

Equation (22) represents the probability for an ant k located on a node i selects the next node denoted by j, where, N_i^k is the set of feasible nodes (in a neighborhood) connected to node i with respect to ant k, τ_{ij} is the total pheromone concentration of link ij, and α is a positive constant used as a gain for the pheromone influence.

Equation (23) represents the evaporation pheromone update, where $\rho \in [0,1]$ is the evaporation rate value of the pheromone trail. The evaporation is added to the algorithm in order to force the exploration of the ants, and avoid premature convergence to sub-optimal solutions. For $\rho = 1$ the search becomes completely random.

Equation (24), represents the concentration pheromone update, where $\Delta\tau_{ij}^k$ is the amount of pheromone that an ant k deposits in a link ij in a time t.

The general steps of S-ACO are the following:

1. *Set a pheromone concentration τ_{ij} to each link (i,j).*
2. *Place a number k=1, 2,..., n_k in the nest.*
3. *Iteratively build a path to the food source (destiny node), using Equation (22) for every ant.*

- *Remove cycles and compute each route weight $f\left(x^k\left(t\right)\right)$. A cycle could be generated when there are no feasible candidates nodes, that is, for any i and any k, $N_i^k = \varnothing$; then the predecessor of that node is included as a former node of the path.*

4. *Apply evaporation using Equation (23).*
5. *Update of the pheromone concentration using Equation (24)*

6. *Finally, finish the algorithm in any of the three different ways:*
   - *When a maximum number of epochs has been reached.*
   - *When it has found an acceptable solution, with $f(x_k(t)) < \varepsilon$.*
   - *When all ants follow the same path.*

3.4 General Remarks about Optimization of Type-2 Fuzzy Systems

The problem of designing type-2 fuzzy systems can be solved with any of the above mentioned optimization methods. The main issue in any of these methods is deciding on the representation of the type-2 fuzzy system in the corresponding optimization paradigm. For example, in the case of GAs, the type-2 fuzzy systems must be represented in the chromosomes. On the other hand, in PSO the fuzzy system is represented as a particle in the optimization process. In the ACO method, the fuzzy system can be represented as one of the paths that the ants can follow in a graph. Also, the evaluation of the fuzzy system must be represented as an objective function in any of the methods.

4 General Overview of the Area and Future Trend

In this section a general overview of the area of type-2 fuzzy system optimization is presented. Also, possible future trends that we can envision based on the review of this area are presented. It has been well-known for a long time that designing fuzzy systems is a difficult task, and this is especially true in the case of type-2 fuzzy systems [5]. The use of GAs, ACO and PSO in designing type-1 fuzzy systems has become a standard practice for automatically designing this sort of systems [1] [2] [13] [21]. This trend has also continued to the type-2 fuzzy systems area, which has been accounted for with the review of papers presented in the previous sections. In the case of designing type-2 fuzzy systems the problem is more complicated due to the higher number of parameters to consider, making it of upmost importance the use of bio-inspired optimization techniques for achieving the optimal designs of this sort of systems. In this section a summary of the total number of papers published in the area of type-2 fuzzy system optimization is presented, so that the increasing trend occurring in this area can be better appreciated. Also, the distribution of papers according to the used optimization technique is presented, so that a general idea of how these different techniques are contributing to the automatic design of optimal type-2 fuzzy systems is obtained.

Figure 6 shows the distribution of the published papers in optimizing type-2 fuzzy systems according to the different bio-inspired optimization techniques previously mentioned. From Figure 6 it can be noted that the use of GAs have been decreasing recently, on the other hand the use of PSO, ACO and other methods have been increasing. The reason for the increase in use of PSO and ACO may be due to recent works in which either PSO or ACO have been able to outperform GAs for different applications. Regarding the question of which method would be the most appropriate for optimizing type-2 fuzzy systems, there is no easy answer. At the moment, what we can be sure of is that the techniques mentioned in this paper and probably newer ones that may appear in the future, would certainly be tested in the optimization of type-2 fuzzy systems because the problem of designing automatically these types of systems is complex enough to require their use.

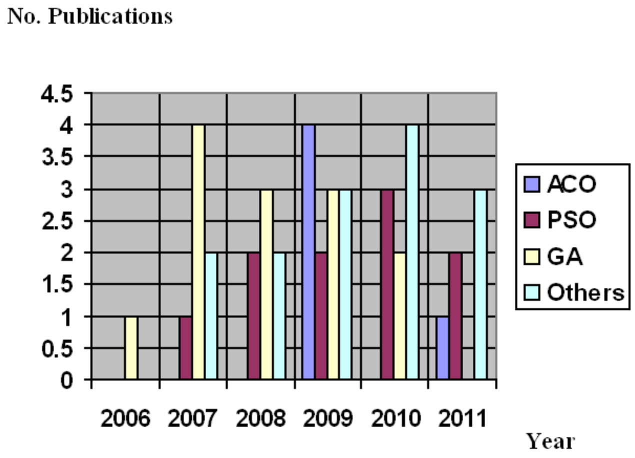

Fig. 6. Distribution of publications per area and year

There are other bio-inspired or nature-inspired techniques that at the moment have not been applied to the optimization of type-2 fuzzy systems that may be worth mentioning. For example, membrane computing, harmony computing, electromagnetism based computing, and other similar approaches have not been applied (to the moment) in the optimization of type-2 fuzzy systems. It is expected that these approaches and similar ones could be applied in the near future in the area of type-2 fuzzy system optimization. Of course, as new bio-inspired and nature-inspired optimization methods are being proposed at any time in this fruitful area of research, it is expected that newer optimization techniques would also be tried in the near future in the automatic design of optimal type-2 fuzzy systems.

5 Conclusions

In the previous sections we have presented a representative account of the different optimization methods that have been applied in the optimal design of type-2 fuzzy systems. To the moment, genetic algorithms have been used more frequently to optimize type-2 fuzzy systems. However, more recently PSO and ACO have attracted more attention and have also been applied with some degree of success to the problem of optimal design of type-2 fuzzy systems. There have been also other optimization methods applied to the optimization of type-2 fuzzy systems, like artificial immune systems and the chemical optimization paradigm. At this time, it would be very difficult to declare one of these optimization techniques as the best for optimizing type-2 fuzzy systems, as different techniques have had success for different applications of type-2 fuzzy logic. In any case, the need for bio-inspired optimization methods is justified due to the complexity of designing type-2 fuzzy systems.

References

[1] Bingül, Z., Karahan, O.: A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control. Expert Systems with Applications 38(1), 1017–1031 (2011)

[2] Cao, J., Li, P., Liu, H., Brown, D.: Adaptive fuzzy controller for vehicle active suspensions with particle swarm optimization. In: Proceedings of SPIE-The International Society of Optical Engineering, vol. 7129 (2008)

[3] Castillo, O., Huesca, G., Valdez, F.: Evolutionary Computing for Topology Optimization of Type-2 Fuzzy Controllers. In: Castillo, O., Melin, P., Kacprzyk, J., Pedrycz, W. (eds.) Hybrid Intelligent Systems. STUD FUZZ, vol. 208, pp. 163–178. Springer, Heidelberg (2008)

[4] Castillo, O., Aguilar, L.T., Cazarez-Castro, N.R., Cardenas, S.: Systematic design of a stable type-2 fuzzy logic controller. Applied Soft Computing Journal 8, 1274–1279 (2008)

[5] Castillo, O., Melin, P., Alanis, A., Montiel, O., Sepulveda, R.: Optimization of interval type-2 fuzzy logic controllers using evolutionary algorithms. Journal of Soft Computing 15(6), 1145–1160 (2011)

[6] Castro, J.R., Castillo, O., Melin, P.: An Interval Type-2 Fuzzy Logic Toolbox for Control Applications. In: Proceedings of FUZZ-IEEE 2007, London, pp. 1–6 (2007)

[7] Castro, J.R., Castillo, O., Martinez, L.G.: Interval type-2 fuzzy logic toolbox. Engineering Letters 15(1), 14 (2007)

[8] Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets and Systems 141, 5–31 (2004)

[9] Cordon, O., Herrera, F., Villar, P.: Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. International Journal of Approximate Reasoning 25, 187–215 (2000)

[10] Dereli, T., Baykasoglu, A., Altun, K., Durmusoglu, A., Turksen, I.B.: Industrial applications of type-2 fuzzy sets and systems: A concise review. Computers in Industry 62, 125–137 (2011)

[11] Hagras, H.: Hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. IEEE Transactions on Fuzzy Systems 12, 524–539 (2004)

[12] Juang, C.-F., Hsu, C.-H.: Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control. IEEE Transactions on Industrial Electronics 56(10), 3931–3940 (2009)

[13] Juang, C.-F., Hsu, C.-H.: Reinforcement interval type-2 fuzzy controller design by online rule generation and Q-value-aided ant colony optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B Cybernetics 39(6), 1528–1542 (2009)

[14] Karnik, N.N., Mendel, J.M.: An Introduction to Type-2 Fuzzy Logic Systems, Technical Report, University of Southern California (1998)

[15] Martinez, R., Castillo, O., Aguilar, L.T.: Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. Information Sciences 179(13), 2158–2174 (2009)

[16] Martinez, R., Rodriguez, A., Castillo, O., Aguilar, L.T.: Type-2 fuzzy logic controllers optimization using genetic algorithms and particle swarm optimization. In: Proceedings of the IEEE International Conference on Granular Computing, GrC 2010, pp. 724–727 (2010)

[17] Mendel, J.M.: Uncertainty, fuzzy logic, and signal processing. Signal Processing Journal 80, 913–933 (2000)

[18] Mohammadi, S.M.A., Gharaveisi, A.A., Mashinchi, M.: An evolutionary tuning technique for type-2 fuzzy logic controller in a non-linear system under uncertainty. In: Proceedings of the 18th Iranian Conference on Electrical Engineering, ICEE 2010, pp. 610–616 (2010)

[19] Oh, S.-K., Jang, H.-J., Pedrycz, W.: A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization. Expert Systems with Applications (2011) (article in press)

[20] Sepulveda, R., Castillo, O., Melin, P., Rodriguez-Diaz, A., Montiel, O.: Experimental study of intelligent controllers under uncertainty using type-1 and type-2 fuzzy logic. Information Sciences 177(10), 2023–2048 (2007)

[21] Wagner, C., Hagras, H.: A genetic algorithm based architecture for evolving type-2 fuzzy logic controllers for real world autonomous mobile robots. In: Proceedings of the IEEE Conference on Fuzzy Systems, London (2007)

[22] Wu, D., Tan, W.-W.: Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers. Engineering Applications of Artificial Intelligence 19(8), 829–841 (2006)

[23] Yager, R.R.: Fuzzy subsets of type II in decisions. J. Cybernetics 10, 137–159 (1980)

[24] Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning. Information Sciences 8, 43–80 (1975)