Comparison between Multiobjective GA and PSO for Parameter Optimization of AT2-FLC for a real application in FPGA

Yazmin Maldonado  
Division of Graduate Studies  
Tijuana Institute of Technology  
Tijuana, B.C. Mexico  
yaz.maldonado@gmail.com

Oscar Castillo  
Division of Graduate Studies  
Tijuana Institute of Technology  
Tijuana, B.C. Mexico  
ocastillo@tectijuana.mx

Abstract—This paper describes the design of a type-2 average fuzzy system on FPGAs and its optimization using multiobjective Particle Swarm Optimization (PSO) and a multiobjective Genetic Algorithm (GA) for the regulation of speed of a DC motor. Based on the concept of evolution, the PSO and GA algorithms are applied to membership functions parameter optimization of type-2 average fuzzy inference systems. Implementations and simulations are carried out in FPGA using the Xilinx system generator. The optimization methods were coded in Matlab. The results of comparison PSO with GA were analyzed statistically.

Keywords—AT2-FLC, PSO, GA, T2-MF, FPGA, ReSDCM

I. INTRODUCTION

Type-2 fuzzy inference systems (T2-FIS) are used successfully in many application areas, and there is an increasing interest in the research and implementations of T2-FIS because they offer bigger advantages in handling uncertainty with respect to type-1 fuzzy systems. Fuzzy logic systems are based on rules, and these rules incorporate linguistic variables, linguistic terms and fuzzy rules. The acquisition of these rules is not an easy task for the expert and is of vital importance in the operation of the controller. The process of adjusting these linguistic terms and rules is usually done by trial and error, which implies a difficult task, and for this reason there have been methods proposed to optimize those elements that over time have taken importance, such as genetic algorithms and particle swarm optimization [2][4][5][6].

A fuzzy inference system (FIS) consists of three stages: Fuzzification, Inference and Defuzzification [3][10][29][30]. Type-1 fuzzy systems (T1-FIS) have exact membership functions (MF), while interval type-2 fuzzy systems (IT2-FIS) are described by membership functions with uncertainty [9][18].

Most of the fuzzy logic applications with physical systems require a real-time operation, and higher density programmable logic devices such as field programmable gate array (FPGA) can be used to integrate large amounts of logic in a single integrated circuit.

This paper proposes the optimization of type-2 membership functions (T2-MF) using GA and PSO for hardware applications such as Regulation Speed of a Direct Current Motor (ReSDCM). The optimization involves taking only certain points of the T2-MF in order to give greater efficiency to the algorithm. The PSO and GA have been tested in an average type-2 fuzzy logic system (AT2-FLC) for ReSDCM.

This paper is organized as follows. In section 2 we present the basic concepts of type-2 fuzzy inference systems and the basic concepts of FPGA, in section 3 we present the optimization methods such as PSO and GA, in section 4 we show the design of AT2-FLC for FPGA. The results and optimization of AT2-FLC for ReSDCM are shown in Section 5. Finally, Section 6 offers conclusions about this work.

II. TYPE-2 FUZZY INFERENCE SYSTEMS

The IT2-FIS consists of four stages: Fuzzification, Inference, Type Reduction and Defuzzification. We describe below these stages.

II.1. Fuzzification: The fuzzification maps a numeric value, \( x = (x_1, x_2, ... x_l) \in X_1 \times X_2 \times ... X_l \equiv X \), into a type-2 fuzzy set \( \tilde{A}_i \) in X. \( \tilde{A}_i \) is a singleton fuzzy set if \( \mu_{\tilde{A}_i}(x) = 1/1 \) for \( x = x^i \) and \( \mu_{\tilde{A}_i}(x) = 0 \) for all others \( x \neq x^i \) [17][19][27].

II.2. Inference: Fuzzy reasoning consists of two blocks, the rules and the inference engine, it works the same way as for type-1 fuzzy systems, except the antecedents fuzzy sets and the consequent are represented by type-2 fuzzy sets. The process consists of combining the rules and maps the input to the output (interval type-2 fuzzy sets), using the Join and Meet operations [19]. For a FIS-IT2 with \( p \) inputs \( x_1 \in X_1, x_2 \in X_2, ... x_p \in X_p \) and one output \( y \in Y \), it is assumed that there are \( M \) rules, the \( l \)th rule in a FIS-IT2 can be written as:

\[
R^l: \text{If } x_1 \text{ is } F_{i1}^l \text{ and } ... \text{ and } x_p \text{ is } F_{ip}^l, \text{ Then } y \text{ is } G^l
\] (1)

where \( l=1,...,M \). Once we have the rules it is necessary to calculate the Join and Meet operations as well as sup-star composition[★][17].

II.3. Type Reductors: The type reductor is used to convert all type-2 fuzzy sets to type-1 fuzzy intervals on the output. There are several methods to calculate the reduced set, such as joint center, center of sums, height, center joint, among others [19]. Equation (2) shows the center of sums type reductor.
III. OPTIMIZATION METHODS

A. Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a bio-inspired optimization method. PSO finds the optimal solution by simulating social behavior. PSO is basically developed through simulation of birds that more in two dimensional space, each particle position and speed.

Since its introduction in 1995, PSO has been improved several times and different applications have been achieved. Most of the modifications to the basic PSO are aimed at improving the convergence and the increasing diversity of the swarm.

A PSO algorithm maintains a swarm of particles, where each particle represents a possible solution. In analogy with the paradigms of evolutionary computation, the particles are transported through a multidimensional search space, where the position of each particle is adjusted according to its own experience and its neighbors. xi (t) represents the position of particle i in the search space at time t, r denotes the discrete time. The position of the particle is modified by the addition of a velocity vi (t), i.e. the current position [2][4], Equation (6) shows the position of the particle.

\[ x_i(t+1) = x_i(t) + v_i(t+1) \]

where \( x_i(0) \sim U(x_{min}, x_{max}) \). The velocity vector reflects both the experimental knowledge of the particle and exchanged the social information. The experimental knowledge of a particle is often referred to as the cognitive component, which is proportional to the distance of the particle from its best position (referred to as the personal best position of the particle) found from the beginning.

PSO can be described as follows; each swarm knows the best position of the particle ('Pbest') and the best global position of the swarm ('Gbest'). The speed of each particle can be calculated using Equation (7) [4].

\[ v_i(t+1) = v_i(t) + c_1 r_1(t) [x_i(t) - x_i(t)] + c_2 r_2(t) [y_i(t) - x_i(t)] \]
where $v_i(t)$ is the velocity of the particle $i$ from $j = 1, ..., n$, at time $t$, $x_i(t)$ is the position of particle $i$ in dimension $j$ at time $t$, $c_1$ and $c_2$ are the positive constants of acceleration used for the cognitive and social components respectively, $r_1(t)$, $r_2(t) \sim U(0,1)$ are random values in the range $[0,1]$. These random values of the algorithm introduce stochastic elements. $y_i$ is the $P_{best}$, is associated with the particle $i$, is the best position of the particle, is the best global position of the particle swarm $P_{gbest}$.

B. Genetic Algorithm

A Genetic Algorithm (GA) [5][11] is a stochastic search and optimization technique based on the mechanics of natural selection. From a principle proposed by Holland [10], the GA has been used successfully to manage a wide variety problems such as control, search, etc [8][12][13][14]. A population of candidate solutions (chromosomes) is held and interacts over a number of iterations to produce better solutions.

The combination of survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search provides a good optimization method. In every generation, a new set of artificial creatures is created using bits and pieces of the fittest of the old individuals; an occasional new part is also tried for good measure.

The GA has applications in a wide variety of fields to develop solutions to complex problems [26], including optimization of fuzzy systems, offering them learning and adaptation capabilities, and in this case, they are commonly called genetic fuzzy systems or fuzzy system hybrids.

IV. DESIGN OF AT2-FLC FOR FPGA

We used the average approximation of interval type-2 fuzzy systems (see Figure 1) codified into VHDL. We need two parallel T1-FIS to obtain the average and for this reason it is sufficient explain one T1-FIS. The AT2-FIS has three stages: fuzzification, inference and defuzzification, and below the behavior of these stages are explained.

A. Fuzzification

There are microelectronic implementations of the fuzzification stage, such as [3]. We present an algorithm that works with the calculation of the slopes of the triangular and trapezoidal MF unlike other methods. The procedure of the algorithm is summarized in three steps: calculate the slope value, calculate the degrees of membership and send to inference stage the degrees of membership and the linguistic terms [22].

The fuzzification stage has inputs as reset, clock(clk), clock enable(ce), two inputs (error($e=x_1$) and change of error($de=x_2$)), each with three membership functions (Negative Big(NB=“01”), Zero (Z=“10”), Positive Big (PB=“11”)), two trapezoidal MFs and one triangular MF, the universe of discourse as the degree of membership are designed for 8 bits however, by simply changing one variable in the VHDL code would increase or decrease the number of bits. Fuzzification stage has outputs as degree and linguistic terms for the error input $(g\_e1,g\_e2,g\_e3,e1,e2,e3)$ and the change of error input $(g\_de1,g\_de2,g\_de3,de1,de2,de3)$. The GA and PSO previously estimated parameters of T2-MF, the procedure of these is explained in section V.

B. Inference

The inference stage receives data sent from the fuzzification stage, which are labels and the membership degree of each input are: $g\_e1$, $g\_e2$, $g\_e3$, $e1$, $e2$, $e3$, $g\_de1$, $g\_de2$, $g\_de3$, $de1$, $de2$, $de3$, so that multiplexes labels and evaluates the rule base illustrated in Table I.

<table>
<thead>
<tr>
<th>TABLE I. RULE MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>BD=01</td>
</tr>
</tbody>
</table>

An example of rules using this codification is: If $e$ is “10” and $de$ is “10” then $C$ (consequent) is $H$. For each of these rules the max-min operation of labels is calculated (8) $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$ and the forces of shooting (gc1, gc2, gc3, gc4, gc5, gc6, gc7, gc8, gc9) are sends to the defuzzification stage.

C. Defuzzification

The Defuzzification stage is calculated using the Height method as shown in Equation (8) [10].

$$y_{(e)} = \frac{\sum C_{m}o_{m}}{\sum o_{m}}$$  \hspace{1cm} (8)

where $C$ is the consequent (firing forces) and $o$ is the consequent tags (labels). Once the consequent is calculated using Equation (8) the defuzzification stage sent to the output the crisp value.

The three stages are targeted on a FPGA Xilinx Spartan 3A XC3S700A device, in Table II shows the device utilization summary for these stage.

<table>
<thead>
<tr>
<th>TABLE II. DEVICE UTILIZATION SUMMARY FOR FUZZIFICATION(F), INFERENCE(I) AND DEFUZZIFICATION(D) STAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Utilization</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Number of Slices</td>
</tr>
<tr>
<td>Number of 4 Input LUTs</td>
</tr>
<tr>
<td>Number of Bonded IOBs</td>
</tr>
</tbody>
</table>

Table II we see that after having synthesized in VHDL the AT2-FIS there is space available on the FPGA.

V. RESULTS AND OPTIMIZATION OF THE AT2-FLC FOR THE REGULATION OF SPEED OF A DC MOTOR

We optimized the T2-MF for ReSDCM with PSO and GA, and we use triangular and trapezoidal T2-MF. Fig. 2 shows the design for the inputs and output of AT2-FLC.
Figure 2. Parameters of the inputs and output T2-MF

Fig. 2 shows the design of the input and output of the AT2-FLC with fixed and variable parameters, where the universe of discourse and the degree of membership are divided into 8 bits. The blue points are fixed, the red dots represent the parameter \(a_2\), the green dots are fixed (\(b_1\)) and the yellow dots represent the parameter \(a_1\). The parameters of these T2-MFs are moved according to the constraints in Table III.

<table>
<thead>
<tr>
<th>TABLE III. BOUNDARY PARAMETERS OF THE CHROMOSOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Parameters U ((P_U))</td>
</tr>
<tr>
<td>(0 &lt; a_{2,U} &lt; 128)</td>
</tr>
<tr>
<td>(b_{1,U} = 128)</td>
</tr>
<tr>
<td>(128 &lt; a_{1,U} &lt; 255)</td>
</tr>
<tr>
<td>Parameters L ((P_L))</td>
</tr>
<tr>
<td>(0 &lt; a_{2,L} &lt; 128)</td>
</tr>
<tr>
<td>(b_{1,L} = 128)</td>
</tr>
<tr>
<td>(128 &lt; a_{1,L} &lt; 255)</td>
</tr>
</tbody>
</table>

where \(P_U\) correspond to the ranges of the upper T2-MF and \(P_L\) correspond to the ranges of the lower T2-MF. We can see in Figure 2 that the GA and PSO only need to move 4 points (\(a_{2\,U}, a_{2\,L}, a_{1\,U}\) and \(a_{1\,L}\)) for each input and output instead of 17 points for each input and output. This gives greater speed and efficiency to the genetic algorithm.

After we designed the shape of the input, output and range is necessary to define the objective function for the GA and PSO. The PSO and GA are of multiojective type [16], because they are based on evaluating three characteristics:

a) Minimum overshoot

\[ \text{if } y(t) > r(t) \rightarrow a_1 = \min(y(t)) - r(t) \]  

(9)

b) Minimum undershoot

\[ a_2 = \left| \min(y(t)) - r(t) \right| \]  

(10)

c) Minimum output steady state error (sse)

\[ sse = \sum_{i=1}^{1000} (y(t) - r(t)) \]  

(11)

where \(y(t)\) is the output of the system and the reference signal \(r(t)\) is given by \([0,70]\) revolution per minute (rpm). Once the AT2-FIS is designed, we propose an AT2-FLC for ReSDCM, in this case is a fuzzy incremental controller [7]. Fig. 3 shows the block diagram of AT2-FLC for ReSDCM.

The control objective of AT2-FLC is:

\[ \lim_{t \to \infty} \left| y(t) - r(t) \right| = 0 \]  

(12)

where \(t\) is the sampling time. The AT2-FLC has the following inputs, error \((e(t))\) and change of error \((e'(t))\), and the output is the control signal \((y(t))\). The inputs are calculated as follows:

\[ e(t) = r(t) - y(t) \]  

(13)

\[ e'(t) = e(t) - e(t-1) \]  

(14)

where \(t\) is the sampling time. The uncertainty is given by:

\[ y'(t) = y(t) + x \cdot \text{rand} \]  

(15)

where \(x\) is the uncertainty level factor \([0,1]\). To test the AT2-FLC we implemented the controller on a DC motor Pittman GM9236S025-R1 of 12V, the PSO and GA was running in line with the DC motor.

A. PSO for the AT2-FLC for ReSDCM

Fig. 4 shows the PSO process for the AT2-FLC.

The initial particles are created randomly with respect to the ranges of the T2-MFs in Table III. The position of the particles is calculated using Equation (6), the particle velocity is obtained from Equation (7). For all experiments we used 10 particles. Fig. 5 shows some results errors of simulation using PSO for AT2-FLC for ReSDCM in Matlab software.
B. Genetic Optimization for the FLC-AT2 for ReSDCM

For the optimization of the AT2-FLC using a GA, one must define the chromosome that represents the information of the individual, which in this case is related to the universe of discourse and the linguistic terms. Fig. 6 shows the chromosome used for the GA.

![GA chromosome for the AT2-FIS](image)

Fig. 6. GA chromosome for the AT2-FIS

Fig. 7 shows the GA process for the AT2-FLC.

![Optimization GA for AT2-FLC](image)

Fig. 7. Optimization GA for AT2-FLC

Fig. 8 shows the errors obtained in simulation using the GA for AT2-FLC for ReSDCM in Matlab.

![Errors for different experiments for GA optimization of the AT2-FLC in simulation](image)

Fig. 8. Errors for different experiments for GA optimization of the AT2-FLC in simulation

Fig. 9 shows the runtime graphs of the GA and PSO respectively, giving an average runtime of the GA of 26.1382 minutes versus 20.1132 minutes of PSO, this in a simulation environment.

![Runtime graphs PSO versus GA in simulation environment](image)

Fig. 9. Runtime graphs PSO versus GA in simulation environment

Table IV shows the comparison of results on simulation versus implementation of the optimization between PSO and GA for the AT2-FLC with different levels of uncertainty.

<table>
<thead>
<tr>
<th>No.</th>
<th>Optimization algorithm</th>
<th>Uncertainty level (x)</th>
<th>Error (Simulation)</th>
<th>Error (Implementation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PSO</td>
<td>0</td>
<td>0.0390</td>
<td>0.8555</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0</td>
<td>0.1172</td>
<td>0.8063</td>
</tr>
<tr>
<td>2</td>
<td>PSO</td>
<td>0.001</td>
<td>0.0566</td>
<td>0.9646</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.001</td>
<td>0.1198</td>
<td>0.7753</td>
</tr>
<tr>
<td>3</td>
<td>PSO</td>
<td>0.005</td>
<td>0.0314</td>
<td>0.9674</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.005</td>
<td>0.1447</td>
<td>0.6441</td>
</tr>
<tr>
<td>4</td>
<td>PSO</td>
<td>0.008</td>
<td>0.0314</td>
<td>1.0602</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.008</td>
<td>0.1250</td>
<td>0.3967</td>
</tr>
<tr>
<td>5</td>
<td>PSO</td>
<td>0.05</td>
<td>0.0930</td>
<td>0.8459</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.05</td>
<td>0.1745</td>
<td>0.5901</td>
</tr>
<tr>
<td>6</td>
<td>PSO</td>
<td>0.08</td>
<td>0.0924</td>
<td>0.9526</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.08</td>
<td>0.4900</td>
<td>1.0036</td>
</tr>
<tr>
<td>7</td>
<td>PSO</td>
<td>0.1</td>
<td>0.1044</td>
<td>0.9774</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.1</td>
<td>0.7031</td>
<td>1.0423</td>
</tr>
<tr>
<td>8</td>
<td>PSO</td>
<td>0.5</td>
<td>0.9946</td>
<td>2.6583</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.5</td>
<td>0.7750</td>
<td>3.0130</td>
</tr>
<tr>
<td>9</td>
<td>PSO</td>
<td>0.8</td>
<td>1.1742</td>
<td>4.1339</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.8</td>
<td>1.3175</td>
<td>4.4775</td>
</tr>
</tbody>
</table>

We analyze statistically the performance of the two controllers using the t-student test; this was not overcome because there is a minimum error of difference between the two controllers. Fig. 10 and Fig. 11 show the speed DC motor for AT2-FLC optimized with PSO and GA, with uncertainty (x=0 and x=0.5).

![Behavior of PSO of the AT2-FLC comparison with GA of the AT2-FLC for ReSDCM without uncertainty level (x=0)](image)

Fig. 10. Behavior of PSO of the AT2-FLC comparison with GA of the AT2-FLC for ReSDCM without uncertainty level (x=0)

![Behavior of PSO of the AT2-FLC comparison with GA of the AT2-FLC for ReSDCM with uncertainty level (x=0.5)](image)

Fig. 11. Behavior of PSO of the AT2-FLC comparison with GA of the AT2-FLC for ReSDCM with uncertainty level (x=0.5)
VI. CONCLUSIONS

The AT2-FIS is a design in VHDL for FPGA, for the fuzzification stage calculating the slope of the corresponding membership function; the inference stage uses the max-min, and the defuzzification stage use of the heights method.

We described the design of the particle swarm optimization and genetic optimization of AT2-FLC for the ReSDCM, where three triangular and trapezoidal membership functions for the two inputs and one output are used in the optimization. The PSO and GA only optimize some parameters of the membership functions, but the rules are not optimized because we are interested in the speed of the algorithm. The objective function of the PSO and GA considers three characteristics: overshoot, undershoot and steady state error.

For the experiments of the PSO and GA we present simulation results. For the implementation results, these were running in line with the DC motor. Comparisons were made between the PSO versus GA of the AT2-FLC in VHDL code ReSDCM. The PSO and GA were implemented in Matlab.

The best AT2-FLC with PSO was obtained with $c_1 = 0.305$, $c_2 = 0.25$ with a runtime of 98.445 minutes, the best AT2-FLC with GA was obtained with a 0.6 Selection rate (SUS), 0.1 of mutation rate in a runtime of 121.73 minutes, and both controllers were implemented in the FPGA.

Once the best AT2-FLC was obtained some level of uncertainty was applied and these results were evaluated with t-student statistical test, from which we can say that PSO is not better than GA for the problem of ReSDCM in FPGA. Both controllers are targeted to a Xilinx Spartan 3A XC3S700A device using Xilinx Foundation Environment. The simulation was carried out using Xilinx System Generator (XSG).

ACKNOWLEDGMENT

We would like to express our gratitude to CONACYT, and Tijuana Institute of Technology for the facilities and resources granted for the development of the research reported in this paper.

REFERENCES