Particle Swarm Optimization versus Genetic Algorithms for Fitting Fuzzy Membership Functions

Ahmed Ali Abdalla Esmin[1,2], Alexandre Rasi Aoki[1], and Germano Lambert-Torres[1]
[1] Instituto de Engenharia Elétrica – Universidade Federal de Itajubá
Av. BPS, 1303 – Itajubá/MG – 37500-0903 – BRAZIL
[2] Fundação Educacional Comunitária Formiguense
Av. Dr. Arnaldo de Senna, 328 – Formiga/MG – 35570-000 – BRAZIL
{ahmed, aoki, germano}@iee.efei.br

Abstract: - The use of Fuzzy Control has been increasing considerably, and its success depends on a number of parameters, such as fuzzy membership functions, that are usually decided upon subjectively. The objective of this paper is to describe and discuss how to improve the performance of the fuzzy reasoning model through fitting fuzzy membership functions using Particle Swarm Optimization algorithm and Genetic Algorithms. An application designed to park a vehicle into a garage; beginning from any start position is used as a case study. Finally the obtained results are discussed and the performance is compared.

Key-Words: - Particle Swarm Optimization, Genetic Algorithms, Fuzzy Membership Functions.

1 Introduction

Fuzzy Sets Theory, proposed by Lofti A. Zadeh, noticed that the technologic resources available at that time, were not enough to provide the automation of the activities regarding the industrial, biologic and chemical problems, which use inherent analogical data not fitted to the computer digital handling, working with well defined numerical data, the so called discreet values.

The Computer Package for Fuzzy Logic Teaching is used as a case study [1]. The main purpose of the package is to park a vehicle into a garage, beginning from any start position. The user should first to develop a fuzzy control rules set and membership functions that will route the vehicle path. The program executes the variables fuzzing and defuzzing process with no action from user. This package provides the students a resource that shows actual happenings of every day life.

This paper presents a genetic training mode and a particle swarm optimization mode for a previously made control.

Genetics algorithms are global optimizing ones, based on natural selection and genetics mechanisms. They use a parallel procure and structured strategy, but random, aiming to reinforce searching of "high aptitude" points, that is to say, points in which the function to be minimized has relative low values. Park, Kandel and Langholz have presented an example of results achieved by this optimization [2].

Particle Swarm Optimization (PSO) finds the optimal solution by simulating such social behaviors of group as fish schooling or bird flocking [3]. That is, PSO is an optimization method that uses a principle of social behavior of a group. A group can effectively achieve the objective by using the common information of every agent, and the information owned by the agent itself.

The control is optimized through automatic fitting of existing membership functions. The PSO algorithms was made defining the subpopulation as the membership functions fitting values, whose parameters are the centers and widths of each fuzzy set. These parameters compose the particle (agent). To check the performance of the fuzzy system, it is executed from an initial set of possible parameters, this information is used to setup each subpopulation adjustment (adaptability) and make the evolution of the population.

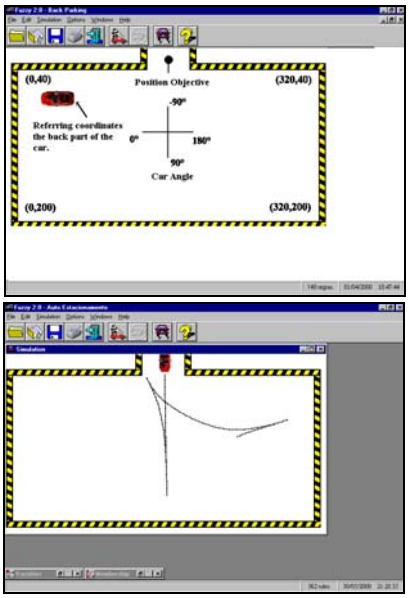
2 The Original Computing Package

The main purpose of the package is to park a vehicle in a garage, beginning from any start position. In order to make it, the user should first to develop a set of fuzzy control rules and membership functions, which will define the vehicle path. Several windows and numeric routines can be found in the program for helping the users in making the rules. The variables fuzzing and defuzzing processes are performed by the program with no action required from the user.

To represent the vehicle parking problem the program has the basic picture shown in Fig. 1 (a). It shows the garage layout, the existing obstacles (the walls in this case) and the coordinates limit values. Also we can see the problem entry variables, i.e. (x, y) measured from the vehicle back central point, and the car angle (ϕ).

The user may define a starting position for the vehicle (Coordinate X, Y and Angle). The simulation begins with a simulation option. Fig. 1 (b) shows the vehicle course using the set of 362 rules for the displacement.

In the simulation example of Fig. 1 (b), it is shown the vehicle track left when routing. Each point means an iteration (i.e. a whole passage through the rules set) being the counting registered in the Variables window. In this example the fuzzy control has made 256 iterations.



(a) (b)

Fig. 1. (a) Program basic picture (b) Simulation Example

If we want to reduce the iterations number, i.e. to minimize the route made by the vehicle, we have to change the rules or to introduce new membership functions, or to adjust the existent ones. To define the best values for membership functions by hand is difficult and takes a lot of time.

A training model using particle swarm optimization was developed in this work and its results are compared with a training model using genetic algorithms. The control is optimized through automatic adjustment of existing membership functions, with no action required from the user. This modulus was introduced in the computing package.

3 Genetic Training Modulus

The integration of genetic algorithms with fuzzy control was made as follow [4]:

- The chromosome was defined as a link of the membership functions adjustment values;
- The parameters are the centers and widths of each fuzzy set. These parameters compose the chromosome genes;
- To check the performance of the fuzzy system it is rolled up from an initial set of possible parameters;
- These information are used for set up each chromosome adjustment (adaptability) and the making of a new population;
- The cycle repetition is made up to completion of the defined generations number made by the user. To each generation it is found the best values set for the membership functions parameters.

For the genetic training it is possible to define the initial locations, as to evaluate each chromosome the vehicle will start from, representing the membership parameters set of values, looking for the control optimization, not for a sole route, but for all possible initial locations from which the vehicle can start for parking.

3.1 Genetic Algorithms Components

For description of each fuzzy controller membership function introduced by the computing package, four parameters are defined. They are: *IE* (bellow left), *ID* (bellow right), *SE* (above left) and *SD* (above right).

For the adjustment of membership functions the following equations (1) were defined:

$$IE = (IE + ki) - w_i; \qquad ID = (ID + k_i) + w_i;$$
$$SE = (SE + k_i); \qquad SD = (SD + k_i). \qquad (1)$$

Being k_i and w_i adjustment factors. k_i makes each membership function move to the right or left with no change in the form. The membership function shrinks or expands through the factor w_i.

These factors take any integer positive or negative value, depending of the defined adjustment value made by the user.

Genetic algorithms are used to find the optimum values according to the strategy and the initial points used, as well as for k_i and w_i for the membership functions.

3.2 Genetic Representation of Solutions

Usually, a possible solution for a problem is associated with a chromosome p by a vector with m positions $p = \{x1, x2, x3, ..., xm\}$, where each component x_i represents a gene.

The most known among the chromosomes representation ways are: the binary representation and

the representation by integers. The binary representation is a classic one.

However, in this paper the representation by integers was used, because the genes of each chromosome are composed by the adjust coefficients k_i and w_i which are integer values.

With regard to chromosome size, i.e., how many genes each chromosome will have, will depend upon the number of membership functions defined by user. For a fuzzy control with a group of 18 membership functions, for example, there will be chromosomes with 36 genes. This happen because for each function we have two adjust coefficients: k_i and w_i. A 36 positions vector then represents the chromosome.

3.3 Evaluation Function

This function deals with the evaluation of the aptitude level (adaptation) of each chromosome generated by algorithms. For the present problem the aim is to minimize the vehicle-parking route. For this case the evaluation function (2) is given by:

$$FA = 1 / (1 + I). \quad (2)$$

Where I is the total iterations up to parking, regarding the adjustment made for each chromosome in the membership function. According to this function, each chromosome aptitude will be inversely proportional to the iterations number.

3.4 Genetic Operators

The genetic operators come from the natural evolution principle and govern the chromosomes renewal. The genetic operators are necessary for the population diversification, and the acquired adaptation characteristics from former generations are preserved.

The crossover process, or recombination, comprises a random cut upon father chromosomes in which genes will be changed, generating two descendents. For example, let us consider two-father chromosomes p_1 - p_2. A cross-point is randomly chosen. The former information up to this point in one of the fathers is linked to the latter information to this point into other father.

The mutation process consists in making the changes according to the adjustment value, defined by the user in the values of one or more chromosome genes. For an adjustment value of 10, for example, the change of a selected gene will fall into some value of the interval [-10, 10].

3.5 Renovation and Selection Criteria

The renovation and selection criteria introduced in this modulus, through the algorithms, were the reproduction one. Reproduction is a process in which the chromosomes copies are used for the next generation, according the evaluation function values. Chromosomes with high aptitude value will contribute with one or more exactly equal descendents for the next generation.

The next generation of effectively reproduced chromosomes number is given by the whole part of the expected descendants from each chromosome. This process was introduced through the roulette technique, where the chromosomes that show a bigger adaptation have greater probability to be selected.

Finally, the roulette turns a determinate number of times, depending on the chromosomes number necessary for the population completion, and the ones drawn by the roulette are chosen as the chromosomes that will take part in the next generation.

3.6 Stop Criteria

The stop criteria used was the one that defines the maximum number of generations to be produced. When genetic algorithms, the new populations generating process is finished, and the best solution to complete the generation number is the one among the individuals better adapted to the evaluation function.

3.7 Algorithm Presentation

Being G the generation number, P the population, Pc the crossover rate (recombination), Pm the mutation rate and VA the allowed adjustment value for membership functions, the algorithm shown bellow generates as starting the vector s with G positions. Each one of the vector elements is the best chromosome of one generation.

STEP 1. Generate initial population P with genes in the interval [-VA, +VA].
STEP 2. Evaluate population P. Store in vector s the best chromosome.
STEP 3. If completed in the generation number G go to STEP 13.
STEP 4. Calculate the aptitude regarding population P.
STEP 5. Calculate the expected descendents from population P.
STEP 6. To draw descendents from population P' starting from population P.
STEP 7. Compose the grouping of population P'.
STEP 8. To draw the cross-point for the father chromosomes of population P'.

STEP 9. Make the crossing for population P' according to Pc.
STEP 10. Make the mutation in each chromosome according to Pm.
STEP 11. Evaluate the population P'. Store in vector s the best chromosome. Make P = P'.
STEP 12. Go back to STEP 3.
STEP 13. End.

4 PSO Training Modulus

Particle Swarm Optimization (PSO) [3] finds the optimal solution by simulating such social behaviors of group as fish schooling or bird flocking. That is, PSO is an optimization method that uses a principle of social behavior of a group. A group can achieve the objective effectively by using the common information of every agent and the information owned by the agent itself.

PSO was basically developed through simulation of bird flocking in two-dimension space. The position of each agent (particle) is represented by XY-axis position and the velocity is expressed by vx and vy (the velocity of X-axis and Y-axis respectively). The agent position is modified by the position and velocity information.

The concept of PSO can be described as follows: Each agent knows its best value so far (*pbest*) and its (XY) position. Moreover, each agent knows the best value in the group (*gbest*) among pbest. Each agent tries to modify its position using the current velocity and its position. The velocity of each agent can be calculated using the equation (3):

$$v_i^{k+1} = w_i \times v_i^k + c_1 \times rand \times (pbest_i - s_i^k) + c_2 \times rand \times (gbest - s_i^k). \quad (3)$$

Where:
v_i^k: Velocity vector of agent i at iteration k;
v_i^{k+1}: modified velocity of agent i at next iteration $k+1$;
s_i^k: Positioning vector of agent i at iteration k;
rand: random number between 0 and 1;
pbest : best position found by agent I;
gbest: best position found by agent group;
c_i: weight coefficients for each term;
w_i: weight function for velocity of agent i.

The current position of an agent is calculated by the equation (4):

$$s_i^{k+1} = s_i^k + v_i^{k+1}. \quad (4)$$

4.1 The Integration of the PSO with Fuzzy Control

The integration of PSO algorithms with fuzzy control was made as follow [5]:

- The subpopulation was defined as a link of the membership functions adjustment values;
- The parameters are the centers and widths of each fuzzy set. These parameters compose the particle (agent);
- To check the performance of the fuzzy system it is rolled up from a initial set of possible parameters;
- These information are used for set up each subpopulation adjustment (adaptability) and making the evolution of these subpopulation;
- The cycle repetition is made up to completion of the defined PSO iteration number made by the user. For each PSO iteration it is found the best values set for the membership functions parameters.

For the PSO training it is possible to define the initial locations, as to evaluate each subpopulation the vehicle will start from, representing the membership parameters set of values, looking for the control optimization not for a sole route but for all possible initial locations from which the vehicle can start for parking.

The evaluation function applied was the same as in GAs. According this function, each subpopulation aptitude will be inversely proportional to the iterations number.

4.2 PSO Representation of Solutions

The fuzzy control usually is composed by many functions. So, for this problem it is necessary to use the subpopulation concept. The solution for a problem is associated with a population of particles composed subpopulation sp by a vector with m positions $sp = \{p_1, p_2, p_3, ..., p_m\}$ where each component p_i represents an agent (particle). Each subpopulation represents one possible solution.

However, each particle position is composed by the adjustment coefficients k_i and w_i which are integer values.

With regard to subpopulation size, i.e., how many agents each subpopulation will have, this depends on the number of membership functions defined by user. For a fuzzy control with a group of 18 membership functions, for example, there will be subpopulation with 36 agents. This is because for each function we have two adjustment coefficients: k_i and w_i. A 36 positions vector then represents the subpopulation. Each particle position is composed by the adjust coefficients k_i and w_i.

4.3 Algorithm Presentation

Being IP the PSO iteration number, P the population, N the subpopulation number, M the particle number, Vmax the velocity max number and VA the allowed adjustment value for membership functions, the algorithm shown bellow generates as a start the vector *gbest* with M positions to store the best subpopulation result.

STEP 1. Generate initial subpopulation with particles in the interval [-VA, +VA] for all subpopulation (N);

STEP 2. Generate initial Velocity Vx and Vy using random values and don't exceed the Vmax for all subpopulation (N);

STEP 3. Evaluate subpopulation (SP_i). If F (SP_i) is better than the $pbest_i$, the F(SP_i) is set to *pbest*. If *pbest* is better than *gbest*, the positioning vector *pbest* is set to *gbest*;

STEP 4. Calculate a new subpopulation velocity values using Eq. (3) for (SP_i);

STEP 5. Calculate a new subpopulation SP_i values using Eq. (4);

STEP 6. IF the iteration number (I) does not complete the Subpopulation number (N) go to STEP 3;

STEP 7. If the PSO iteration number reaches to the pre-determined one (IP), then stop. Otherwise, go to STEP 3.

5 Tests and Results

Table 1 shows the three start positions used for tests and the vehicle's number of iterations until parking, using the originals and trained membership functions.

These positions were chosen according to the points where the vehicle doesn't follow a good route till parking. The definition of several initial positions will result in a global minimization of traveled space. The defined GAs and PSO parameters are shown in Table 2 and 3 respectively.

Table 4 presents the simulation results made starting from initial positions not used in the training.

The results presents an average reduction of iterations number for vehicle to reach the final position for the PSO and GAs trained control. These values represent the global reduction of vehicle route starting from positions not used in AGs and PSO training. The PSO training algorithm is slower than GAs training, because of the amount of communication performed by the particles (agents) after each iteration.

It is possible to notice that in some positions (position 8) the iterations number is bigger than the ones generated by the original control (without training). This increase comes from the modifications made in the membership functions, that makes the vehicle to change for a different route to reach the final position.

Table 4. Simulations results

Position	X	Y	Car Angle	Iterations generated by Fuzzy Controls		
				Original	Trained GAs	Trained PSO
1	1	126	182	450	329	445
2	6	46	132	167	154	306
3	8	41	190	1000	1000	1000
4	15	70	-90	318	162	313
5	70	95	-6	263	275	257
6	74	69	-70	453	456	450
7	76	193	232	605	363	363
8	88	46	44	283	305	289
9	115	120	240	463	411	289
10	131	140	-72	457	292	512
11	141	69	-28	342	314	225
12	154	166	-80	863	436	950
13	160	135	268	1101	545	445
14	217	66	-50	684	325	506
15	228	194	-48	830	655	476
16	246	169	154	312	307	310
17	250	180	-40	739	800	489
18	265	170	-40	672	329	483
19	300	124	258	317	306	308
20	305	156	-40	521	318	449
Total				**10840**	**8082**	**8865**
Average				**542**	**404,1**	**443,25**

Table 1. Start positions before and after training with time spent in seconds.

Position	X	Y	Car angle	Iterations without training	Iterations with PSO training	Iterations with GAs training
1	25	120	180	330	285	280
2	160	130	-90	888	592	384
3	275	160	-40	655	439	277
Total				1873	1316	941
Average				624.33	438.66	331.67
Time (seconds)					1891	1166

Table 2. GA parameters.

Population Size	14
Generations Number	30
Crossover Probability	90%
Mutation Probability	1%

Table 3. PSO parameters.

Population Size	14
Iteration Number	30
Vmax	10

5 Conclusions

The fuzzy systems are a convenient and efficient alternative for solution of problems where the fuzzy state are well defined. Nevertheless, the project of a fuzzy system may became difficult for large and complex systems, when the control quality depends on subjective decisions to define the best membership functions to solve the problem.

This paper presented and compared the Particle Swarm Optimization and the Genetic Algorithms training modulus, applied to fitting fuzzy membership functions. As a case study, a computing package for the fuzzy logic teaching is used. The PSO and GAs training modules developed in this work are added in this package with an automatic technique for the fitting of the membership functions parameters. This technique shows that the performance of a fuzzy control may be improved through the AGs and PSO algorithms.

The genetic algorithms provided distinctive advantages for the optimization of membership functions, resulting in a global survey, reducing the chances of ending into a local minimum, once it uses several sets of simultaneous solutions. The fuzzy logic supplied the evaluation function, a stage of the genetic algorithm where the adjustment is settled.

PSO is able to generate an optimal set of parameters for fuzzy reasoning model based on either their initial subjective selection or on a random selection. It is also shown that by training this algorithm with some specific start positions it is reached a good global optimization result.

The implementation of PSO is easier than GAs, but the PSO training algorithm is slower than GAs training, because the needs of communication between the particles (agents) after each iteration. Both algorithms show better results than fuzzy control, and some studies are under development to improve the PSO algorithm [6].

References:

[1] G. Lambert-Torres, V.H. Quintana & L.E. Borges da Silva - "A Fuzzy Control Lab for Educational Purposes," *Proceedings of the Canadian Conference on Engineering Education*, pp. 117-124, Kingston, Canada, Jun. 16-18, 1996.

[2] D. Park, A. Kandel & G. Langholz - "Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control," *IEEE Transactions on SMC*, Vol. 24, No. 1, pp. 39-47, January 1994.

[3] J. Kennedy and R.C. Eberhart - "Particle swarm optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, 1942-1948. IEEE Press.

[4] M.A. Carvalho, G. Lambert-Torres, L.E. Borges da Silva & J.O.P. Pinto, "Fitting Fuzzy Membership Functions using Genetic Algorithms." In: *Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 8-11, 2000, Nashville, USA.

[5] A.A.A. Esmin, A.R. Aoki & G. Lambert-Torres - "Fitting Fuzzy Membership Functions using Particle Swarm Optimization." In: *Proceedings of the Sixth ICNNSC - International Conference on Neural Networks and Soft Computing*, June 11-15, 2002, Zakopane, Polish.

[6] Y. Shi and R.C. Eberhart, "A modified Particle Swarm Optimiser", *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4-9, 1998.