# 19

# An Evolutionary Fuzzy Hybrid System for Educational Purposes

Ahmed Ali Abadalla Esmin[1], Marcos Alberto de Carvalho[2],
Carlos Henrique Valério de Moraes[3] and Germano Lambert-Torres[3]
[1]*Lavras Federal University*
[2]*José do Rosário Vellano University*
[3]*Itajuba Federal University*
*Brazil*

## 1. Introduction

When the Fuzzy Set Theory was proposed by Lotfi Zadeh in a seminal paper published in 1965, he noted that the technological resources available until then were not able to automate the activities related to industrial, biological or chemical problems. These activities use typically analog data which are inappropriate to be handled in a digital computer that works with well-defined numerical data, i.e. , discrete values.

Using this idea, Fuzzy Logic can be defined as a way to use data from typical analog processes that move through a continuous track in a digital computer that works with discrete values. The use of Fuzzy Logic for solving control problems has tremendously increased over the last few years. Recently the Fuzzy Logic has been used in industrial process control electronic equipment, entertainment devices, diagnose systems and even to control appliances. Thus, the teaching of fuzzy control in engineering courses is becoming a necessity. In a previous work, it has been presented a computational package for students' self-training on fuzzy control theory. The package contains all required instructions for the users to gain the understanding of fuzzy control principles. The training instructions are presented via a practical example.

Although this approach has proven to be convenient in giving to students an opportunity to appreciate real life like situations, it suffers a serious disadvantage: the type of learning. In fact, students often go through a "trial-and-error" method to select an appropriate control action, such as rule definitions or membership fitting. The problem of this type of learning is a tendency from students to get the erroneous concept that corrective actions are much a matter of guess. The purpose of this chapter is to present a strategy for an automatic membership function fitting using three different evolutionary algorithms, namely: modified genetic algorithms (MGA), particle swarm optimization (PSO) and hybrid particle swarm optimization (HPSO).

The proposed strategies are applied in a computational package for fuzzy logic learning. This computer program was developed for self-training in engineering students in the

theory of fuzzy control. The program contains all necessary instructions for users to understand the principles of diffuse control. In this package the main goal is to park a vehicle in a garage, starting from any starting position. The user must first develop a set of fuzzy control rules and functions of relevance that will shape the trajectory of the vehicle. The processes of fuzzification and defuzzification variables are performed by the program without user interference.

## 2. Description of the main features of the training package

The computational package has as main objective to park a vehicle in a garage, starting from any starting point within a pre-defined area. To this goal, the user should design a set of fuzzy control rules and also the functions of relevance that will control the trajectory of the vehicle. To set these rules, the program offers various menus with Windows and numerical routines. The processes of fuzzification and defuzzification variables are made by the program without user interference (Park et al., 1994).

Figure 1 shows the main screen to represent the problem of parking of a vehicle. This window shows the position of the garage, the existing limits (the walls) and the values of coordinated limits. Also this window presents the input variables *(x, y)* measured from the center point of the rear of the vehicle and finally, the car angle ($\phi$).

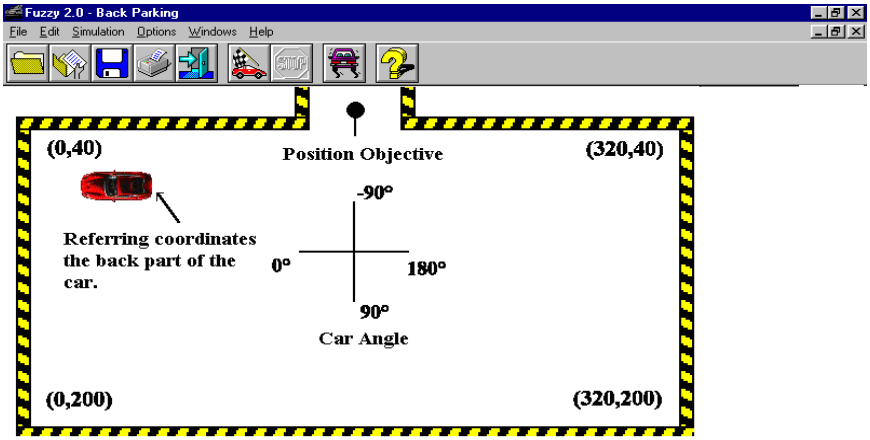



Fig. 1. Main screen of the program

For parking the vehicle, some conditions are established, and belong to two types: computational package related and linked to logical. The conditions attaching to the package represent the physical limitations, they are:

a. limits of input variables:
- position $(x, y)$: $0 < x < 32$ and $0 < y < 20$ (in meters - parking dimensions);
- car angle of $-90° \leq \phi \leq 270°$;
- sense of the vehicle: forward or backward.

b. limit of the output variable:
- vehicle wheel angle $-30° \leq \theta \leq 30°$ (limitation of the real model).

With respect to logical limitations, they may vary according to the types of strategies employed. Some examples of these strategies can be, among others:

- minimization of the number of changes of vehicle direction (forward or backward);
- minimizing the space traversed by vehicle to the garage;
- the restriction of parts of garage for parking.

For the movement of the vehicle shall be laid down the following conditions: Acceleration equal to 1 (m/s$^2$) and maximum speed 1 (m/s). These two values are used as reference for all movements. To reverse the direction of motion of the vehicle there are three possibilities, which are:

a. shock against the wall: when the system verifies that the vehicle will collide against the wall in the next step;
b. rule that forces the inversion: when the reverse order is used as a result of a rule; or,
c. lack of outputs: when no rule is used by the control, i.e., if the output is zero.

The user of this computational package can define a new system by creating the roles of relevance and control rules. Initially, the user sets the number of functions of relevance for each variable. When the functions are created, they are equally spaced on the surface of the control variable. The user can modify these functions of belongingness by Fuzzy Sets Edition window. Figure 2 presents an example of editing for the $x$ input variable.

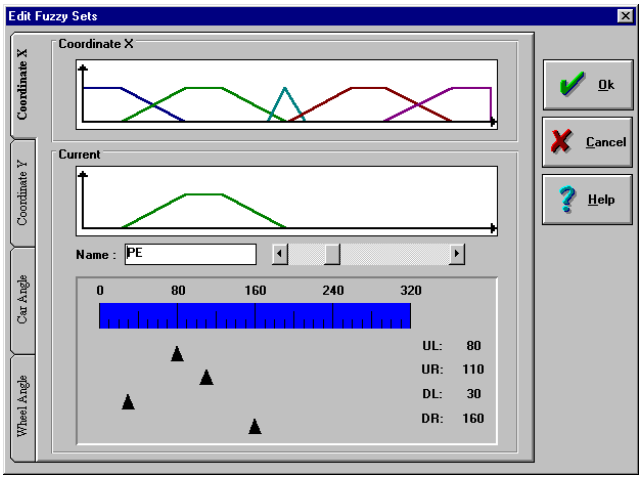



Fig. 2. Fuzzy Sets Edition window

To set the rules for the control, i.e., how the functions of relevance will be grouped, there is the Fuzzy Rules Edition window. In Figure 3, one can find two regions of interest. The first where there is the possibility of selecting the direction (forward or reverse) and coordinated corresponding to the angle of the car. The second region of interest contains the padding of the conclusion of the rule. This can be done by selecting one of the output values (or none for a rule does not set or reverse). For instance for $x$ = LE (left), $y$ = YT (small values), car angle = RB (right big angle) and direction = ahead (forward) values was selected as NB (negative big angle) to wheel angle, which corresponds to rule:

"IF *x* is **and** LE *y* **and** is YT **and** *car angle* is RB **and** direction of movement is *forward (ahead)* THEN wheel angle is NB."

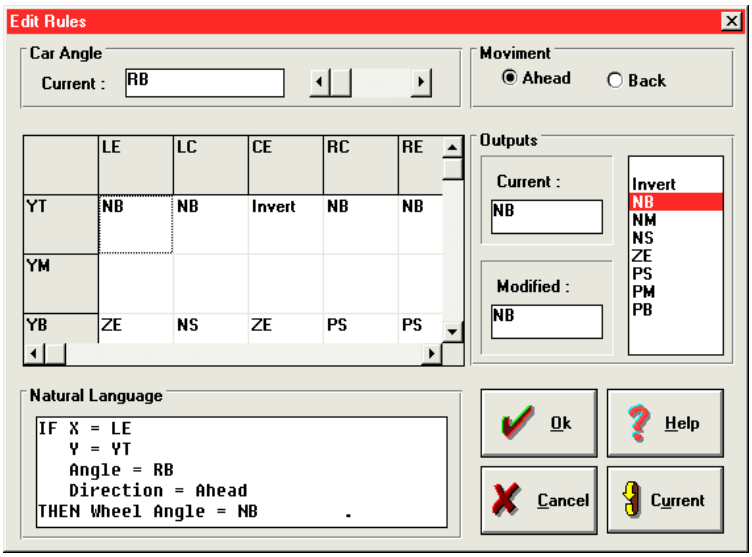



Fig. 3. Fuzzy Rules Edition window

Through the initial position of the Edition menu option the user can set a start position (x-*y* coordinates and car angle $\phi$) to the vehicle. The simulation is started through the Start menu simulation. Figure 4 shows three possible initial positions.

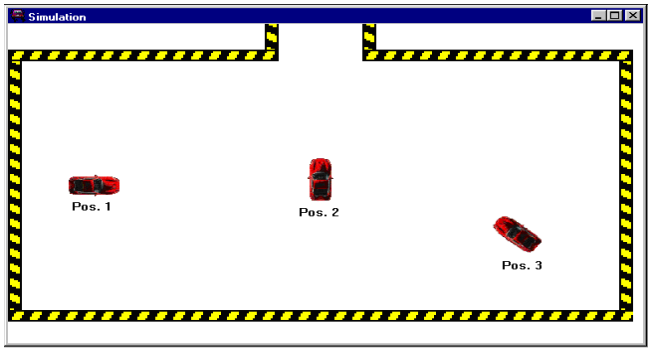
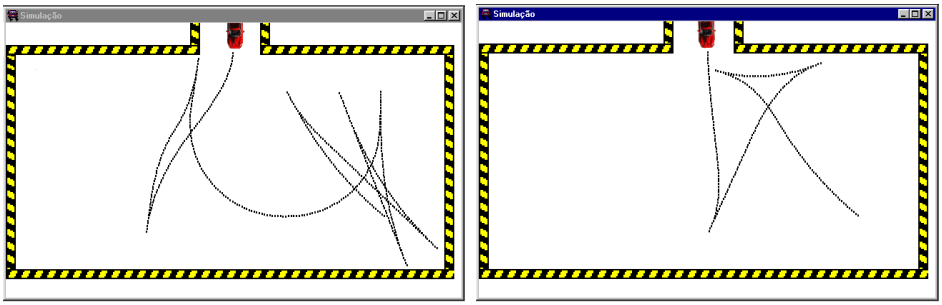



Fig. 4. Three possible start positions

In the examples of simulation in Figure 5, you can check the trail left by the vehicle during its trajectory. Each point means iteration (i.e. a full pass on the set of rules) and the count is recorded in the variables window. The first example, shown in Figure 5 (a), has produced 628 iterations; while, in the second, in Figure 5(b), for another set of rules was queried by 224 times, for the same start position. It is easy to see that the second set of rules has a batter performance than the first one for this start position.




(a) (b)

Fig. 5. Examples of computational simulation package

The computational package has features that allow you to vary the size of the car between: small, medium or large. This variation creates the opportunity to verify the behavior of the control system for an outfit that has changed some of its grandeur. The computational package also has three methods of defuzzification, which are: the centroid (center of gravity), average of the areas and average of maximum values (Kandel & Langholz, 1993).

In the first version of the computational package, the learning process used is by trial and error. The user creates the functions of relevance, provides a set of rules and then performs several tests to verify the quality of control. It is known that this learning process (by trial and error) may not bring the expected results because several errors of interpretation can occur (da Silva et al., 2010).

## 3. Description of the evolutionary methods used in the hybrid system

The whole task of search and the optimization has several components, including: the search space, where they are considered all possibilities of solution of a given problem and the evaluation function (or function), a way to evaluate members of the search space. There are many methods of search and evaluation functions.

Search optimization techniques and traditional begin with a single candidate, iteratively, is manipulated using some heuristics (static) directly associated with the problem to be solved. Generally, these processes are not heuristic algorithmic and its simulation in computers can be very complex. Despite these methods were not sufficiently robust, this does not imply they are useless. In practice, they are widely used, successfully, in innumerable applications (Ross, 2010).

On the other hand, the evolutionary computation techniques operate on a population of candidates in parallel. Thus, they can search in different areas of the solution space, allocating an appropriate number of members to search in multiple regions.

Meta-heuristic methods differ from traditional methods of search and optimization, mainly in four aspects (Esmin et al., 2005; Medsker, 2005):

1. Meta-heuristic methods work with an encoding of the set of parameters and not with their own parameters.
2. Meta-heuristic methods work with a population and not with a single point.

3. Meta-heuristic methods use cost information or reward and not derived or other auxiliary knowledge.
4. Meta-heuristic methods use probabilistic transition rules and not deterministic.

In addition to being a strategy to generate-and-test very elegant, because they are based on social organization or biological evolution, are able to identify and explore environmental factors and converge to optimal solutions, or approximately optimal in overall levels. The better a person adapt to their environment, the greater your chance of surviving and generate descendants: this is the basic concept of social organization or biological genetic evolution. The biological area more closely linked to genetic algorithms is the genetics, and the social area is particle swarm optimization.

### 3.1 Genetic algorithms

In the years 50 and 60, many biologists began to develop computational simulations of genetic systems. However, it was John Holland who began, in earnest, developing the first researches in the theme. Holland was gradually refining their ideas and in 1975 published his book "Adaptation in Natural and Artificial Systems" (Holland, 1975), now considered the bible of genetic algorithms. Since then, these algorithms are being applied with success in the most diverse problems of optimization and machine learning.

Genetic algorithms are global optimization algorithms, based on the mechanisms of natural selection and genetics. They employ a parallel search strategy and structured, but random, which is geared toward enhancing search of "high fitness" points, i.e. points where the function to be minimized (or maximized) has values relatively low (or high).

Although they are not random, random walks, directed not because explore historical information to find new points of search where are expected best performances. This is done through iterative processes, where each iteration is called generation.

During each iteration, the principles of selection and reproduction are applied to a population of candidates that can vary, depending on the complexity of the problem and the computational resources available. Through the selection, if determines which individuals will be able to reproduce, generating a particular number of descendants for the next generation, with a probability given by its index of fitness. In other words, individuals with greater relative adaptation have greater chances of reproducing.

The starting point for the use of genetic algorithms, as a tool for troubleshooting is the representation of these problems in a way that the genetic algorithms to work properly on them. Most representations are genotype, use vectors of finite size in a finite alphabet.

Traditionally, individuals are represented by binary vectors, where each element of a vector (1) denotes the presence or absence (0) of a particular characteristic. However, there are applications where it is more convenient to use representations for integers as shown later in this work.

The basic principle of operation of AGs is that a selection criterion will do with that, after many generations, the initial set of individuals generates another set of individuals more able. Most methods are designed to check individuals preferentially choose majors with fitness, although not exclusively, in order to maintain the diversity of the population. A

selection method used is the method of roulette, where individuals of one generation are chosen to be part of the next generation, through a raffle of roulette. Figure 6 shows the representation of the roulette to a population of 4 individuals.

| *I* | Individual $x_i$ | $f_i(x)$ Fitness Function ($x^2$) | Relative Values of Fitness Function |
|---|---|---|---|
| 1 | 00101 | 25 | 0.04 |
| 2 | 00010 | 4 | 0.01 |
| 3 | 10110 | 484 | 0.81 |
| 4 | 01001 | 81 | 0.14 |
| **Total** | | 594 | 1.00 |

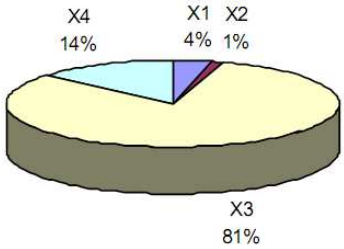



Fig. 6. Individuals of a population and its corresponding check roulette

In this method, each individual of the population is represented in roulette in proportion to its index of fitness. Thus, individuals with high fitness are given a greater portion of the wheel, while the lowest fitness is given a relatively smaller portion of roulette. Finally, the roulette wheel is rotated a certain number of times, depending on the size of the population, and are chosen, as individuals who will participate in the next generation, those drawn in roulette.

A set of operations is necessary so that, given a population, to generate successive populations that (hopefully) improve your fitness with time. These operators are: crossover (crossover) and mutation. They are used to ensure that the new generation is entirely new, but has in some way, characteristics of their parents, i.e. the population diversifies and maintains adaptation characteristics acquired by previous generations. To prevent the best individuals does not disappear from the population by manipulating the genetic operators; they can be automatically placed on the next generation via playing elitist.

This cycle is repeated a specified number of times. The following is an example of genetic algorithm. During this process, the best individuals, as well as some statistical data, can be collected and stored for evaluation.

```
Procedure AG
{g = 0;
inicial_population (P, g)
evaluation (P, g);
Repeat until (g = t)
{g = g +1;
      Father_selection (P, g);
recombination (P, g);
mutation (P, g);
evaluation (P, g);
   }
}
```

Where *g* is the current generation; *t* is the number of generations to terminate the algorithm; and *P* is the population.

These algorithms are computationally very simple, are quite powerful. In addition, they are not limited by assumptions about the search space, for continuity, existence of derivatives, and so on.

### 3.1.1 Genetic operators

The basic principle of genetic operators is to transform the population through successive generations, extending the search until you reach a satisfactory outcome. Genetic operators are needed to enable the population to diversify and keep adaptation characteristics acquired by previous generations.

The operator mutation is necessary for the introduction and maintenance of genetic diversity of the population, arbitrarily changing one or more components of a structure chosen, as is illustrated in Figure 7, thus providing the means for introduction of new elements in the population. Thus, the mutation ensures that the probability of reaching any point in the search space will never be zero, in addition to circumvent the problem of local minima, because with this mechanism, slightly changes the search direction. The mutation operator is applied to individuals with a probability given by the mutation rate *Pm*; usually uses a small mutation rate, because it is a genetic operator secondary.

| | |
|---|---|
| Before the mutation | 0 1 0 0 0 |
| After the mutation | 0 1 1 0 0 |

Fig. 7. Example of mutation

The crossing is the operator responsible for the recombination of traits of parents during play, allowing future generations to inherit these traits. It is considered the predominant genetic operator, so it is applied with probability given by the crossover rate *Pc*, which must be greater than the rate of mutation.

This operator can also be used in several ways; the most commonly used are:

a. One-point: a crossover point is chosen and from this point the parental genetic information will be exchanged. The information prior to this point in one of the parents is related to information subsequent to this point in the other parent, as shown in the example in Figure 8.

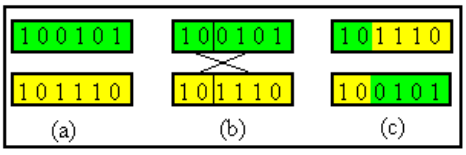



Fig. 8. Example of crossover from one-point: (a) two individuals are chosen; (b) a crossover point (2) is chosen; (c) the characteristics are recombined, generating two new individuals

b. Multi-points: is a generalization of this idea of an exchange of genetic material through points, where many crossing points can be used.
c. Uniform: don't use crossing points, but determines, through a global parameter, which the probability of each variable be exchanged between parents.

### 3.1.2 Genetic parameters

It is also important to analyze how some parameters influence the behavior of genetic algorithms in order to establish them as the needs of the problem and available resources.

**Population size.** The size of the population affects the overall performance and efficiency of AGs. With a small population performance may fall, because this way the population provides a small coverage of the search space of the problem. A large population typically provides a representative coverage of the problem domain, and preventing premature convergence solutions to local rather than global. However, for working with large populations, larger computational resources are required, or that the algorithm works by a much longer time period.

**Passing Rate.** The higher this ratio, the faster new structures will be introduced in the population. But if this is too high, the majority of the population will be replaced and can be lost high fitness structures. With a low value, the algorithm can become very slow.

**Mutation rate.** A low mutation rate prevents a given position stay stagnant in a value, and allow to reach anywhere in the search space. With a very high search becomes essentially random.

## 3.2 Particle Swarm Optimization

The optimization method called Particle Swarm Optimization (PSO) as other meta-heuristics recently developed, simulates the behavior of systems making the analogy with social behaviors. PSO was originally inspired by biological partner behavior associated with group of birds (Goldberg, 1989). This topic will be discussed in more detail after the basic algorithm is described.

The PSO was first proposed by John Kennedy and Russell Eberhart (1995a, 1995b). Some of the interesting features of PSO include ease of implementation and the fact that no gradient information is required. It can be used to solve a range of different optimization problems, including most of the problems can be solved through genetic algorithms; one can cite as an example some of the applications, such as neural network training (Lee & El-Sharkawi, 2008) and to minimize various types of functions (Eberhart et al., 1996).

Many popular optimizations algorithms are deterministic, as the gradient-based algorithms. The PSO, like its similar, belonging to the family of Evolutionary Algorithm is an algorithm of stochastic type that needs no gradient information derived from error function. This allows the use of PSO in functions where the gradient is unavailable or the production of which is associated with a high computational cost.

### 3.2.1 The PSO algorithm

The algorithm maintains a population of particles, where each particle represents a potential solution to an optimization problem. $S$ assumed as being the size of the swarm. $I$ each particle can be represented as an object with various features. These characteristics are as follows:

$x_i$: the current position of the particle;
$v_i$: the current speed of the particle;
$y_i$: the best personal position achieved by the particle.

The best personal position *i* particle represents the best position that the particle has visited and where he obtained the best evaluation. In the case of a task of minimizing, for example, a position that earned the lowest function value is considered to be the best position or with highest fitness assessment. *F* symbol is used to denote the objective function being minimized. The update equation for the best staff position is given by equation (1) using *t* time explicitly.

$$y_i(t+1) = \begin{cases} y_i(t) & if \quad f(y_i(t) \le f(x_i(t+1))) \\ x_i(t+1) & if \quad f(y_i(t) > f(x_i(t+1))) \end{cases} \tag{1}$$

There are two versions of PSO, calls *gbest* templates and *lbest (the global best and the best place)* (Goldberg, 1989). The difference between the two algorithms is based directly in the way that a particular particle interacts with its set of particles. To represent this interaction will be used the symbol $\hat{y}$ . The details of the two models will be discussed in full later. The definition of $\hat{y}$ as used in *gbest* model, is shown by equation (2).

$$\begin{aligned} \hat{y}(t) \in \{y_0(t), y_1(t), ....., y_s(t)\} \mid \; & f(\hat{y}(t)) \\ = \min\{f(y_0(t)), f(y_1(t)), ....., f(y_s(t))\} & \end{aligned} \tag{2}$$

Note that this definition shows that $\hat{y}$ is the best position until then found by all particles in the swarm *S* size.

The PSO algorithm makes use of two independent random sequences $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$. These strings are used to give nature to stochastic algorithm, as shown below in the equation (3). The values of $r_1$ and $r_2$ are scaled through constant $c_1 > 0$ and $c_2 \le 2$. These constants are called *acceleration coefficients*, and they exert influence on the maximum size of a particle can give in a single iteration. The speed that updates the step is specified separately for each dimension $j \in 1 \ldots n$, so that $v_{i,j}$ denotes the dimension j vector associated with the particle speed *i*. The update speed is given by the following equation:

$$\begin{aligned} v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + \\ c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \end{aligned} \tag{3}$$

In the definition of the equation, the constant speed update $c_2$ regulates clearly the maximum size of the step in the direction of better global particle, and the constant $c_1$ adjusts the size of the step in the direction of better personal position of the particle. The value of $v_{i,j}$ is maintained within the range of $[-v_{max}, v_{max}]$ by reducing the probability that a particle can exit the search space. If the search space is defined by the interval $[-x_{max}, x_{max}]$, then the value of $v_{max}$ is calculated as follows:

$$v_{max} = k \quad X \quad x_{max} \quad \text{where} \quad 0.1 \le k \ge 1.0$$

The position of each particle is updated using your new velocity vector:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4}$$

The algorithm consists of repeated application of the equations above update. Below the basic PSO algorithm code is shown.

*Create and initialize:*
*i* – current particle;
*s* – PSO of n-dimensions:

***repeat:***
  ***for each*** *particle i = [1 .. s]*
    ***If*** $f(S_i.x.x) < f(S_i.y)$
      *then* $S_i.y = x_i S.$
    ***If*** $f(S_i.y) < f(s.\ \hat{y}\ )$
      ***then*** $S.\ \hat{y}\ = Y_i S.$
  ***end loop***
*Update S using the equations (3) and (4)*
***until*** *the stopping condition is* ***True***
***end***

The startup mentioned in the first step of the algorithm consists of the following:

1. initialize each coordinated $x_{i,j}$ with a random value in the range $[-x_{max}, x_{max}]$, for the entire $i \in 1 \ldots s$ and $j \in 1 \ldots n$. This distributes the initial positions of the particles along the search space. Select a good random distribution algorithm to obtain a uniform distribution in the search space.
2. initialize each $v_{i,j}$ with a value taken from the range $[-v_{max}, v_{max}]$ for the entire $i \in 1 \ldots s$ and $j \in 1 \ldots n$. Alternatively, the velocities of particles may be initialized with 0 (zero), provided that the initial positions are initialized in a random fashion.

The stopping criterion mentioned in the algorithm depends on the type of problem to be solved. Typically the algorithm is run for a predetermined and fixed number of iterations (a fixed number of function evaluation) or until it reaches a specific value of error. It is important to realize that the term speed models the rate of change in the position of the particle. The changes induced by speed update equation (3) represent acceleration, which explains why the constants $c_1$ and $c_2$ are called acceleration coefficients.

A brief description of how the algorithm works is given as follows: Initially, a particle any is identified as being the best particle in the group, based on his ability using the objective function. Then, all particles will be accelerated in the direction of this particle, and at the same time in the direction of own best positions previously found. Occasionally particles explore the search space around the current best particle. This way, all particles will have the opportunity to change their direction and seek a new 'best' particle. Whereas most functions have some form of continuity, chances are good to find the best solutions in the space that surrounds the best particle. Approximation of the particles coming from different directions in the search space towards the best solution increases the chances of finding the best solutions that are in the area nearby the best particle.

### 3.2.2 The behavior of the PSO

Many interpretations have been suggested regarding the operation and behavior of the PSO. Kennedy, in his research strengthened biological vision partner-PSO, performing experiments to investigate the roles of the different components of the velocity update

equation (Kennedy & Eberhart, 1995). The task of training a neural network was used to compare performance of different models. Kennedy made use of *lbest* model (see *lbest* section for a complete description of this template), instead *gbest* model.

For this update equations developed two speed, the first by using just the experience of the particle, called the *component of cognition*, and the second, using only the interaction between the particles and called *social component.*

Consider the equation speed update (3) presented earlier. The term $c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)]$ is associated only with the cognition, where it takes into account only the experiences of the particle itself. If an OSP is built using only the cognitive component, the upgrade speed equation becomes:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)]$$

Kennedy found that the performance of this model of "only with cognition" was less than the original PSO's performance. One of the reasons of bad performance is attributed to total absence of interaction between the different particles.

The third term in the equation, speed update $c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)]$, represents the social interaction between the particles. A version of PSO with just the social component can be constructed using the following equation: speed update

$$v_{i,j}(t+1) = v_{i,j}(t) + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)]$$

It was observed that in the specific problems that Kennedy, investigated the performance of this model was superior to the original PSO.

In summary, the term speed of PSO update consists of two components, the component of cognition and the social component. Currently, little is known about the relative importance of them, although initial results indicate that the social component is more important in most of the problems studied. This social interaction between the particles develops cooperation between them to resolve the problem.

### 3.2.3 Model of the best global (*gbest*)

The model allows *gbest* a faster rate of convergence at the expense of robustness. This model keeps only a single "best solution", called the *best global particle*, between all particles in the swarm. This particle acts as an attractor, pulling all particles to it. Eventually, all particles will converge to this position. If it is not updated regularly, the swarm can converge prematurely. The equations for update $\hat{y}$ and $x_i$ are the same as shown above:

$$\begin{aligned} \hat{y}(t) \in \{y_0(t), y_1(t), ....., y_s(t)\} \mid\ & f(\hat{y}(t)) \\ = \min\{f(y_0(t)), f(y_1(t)), ....., f(y_s(t))\} & \end{aligned} \tag{5}$$

$$\begin{aligned} v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + \\ c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \end{aligned} \tag{6}$$

Note that $\hat{y}$ is called *the best overall position*, and belongs to the particle called *the best global particle*.

### 3.2.4 The model of the best location (*lbest*)

The *lbest* model tries to prevent premature convergence keeping multiple attractors. A subset of particles is defined for each particle of which is selected *the best local particle*, $\hat{y}_i$. The symbol $\hat{y}_i$ is called *the best local position* or *better in the vicinity (the local best position or the neighborhood best)*. Assuming that the indexes of the particles are around space *s*, the equations of *lbest* update for a neighborhood size *l* are as follows:

$$N_i = \{y_{i-l}(t), y_{i-l+1}(t), ....., y_{i-1}(t), y_i(t), y_{i+1}(t), ...., y_{i+l}(t)\} \tag{7}$$

$$\hat{y}_i(t+1) \in N_i \mid f(\hat{y}_i(t+1)) = \min\{f(a)\}, \forall a \in N_i \tag{8}$$

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \tag{9}$$

Note that the particles are selected in the subset $N_i$ and they have no relation with the other particles within the domain of the search space; the selection is based solely on the index of the particle. This is done for two main reasons: the computational cost is lower, by not requiring grouping, and this also helps to promote the expansion of information on good solutions for all particles, although these are local search.

Finally, you can observe that the *gbest* model is in fact a special case of *lbest* model, when the $l = s$, i.e. when the selected set encompasses the entire swarm.

### 3.2.5 Considerations about the similarity between PSO and EAs

There is a clear relationship of PSO with the evolutionary algorithms (EAs). To some authors, the PSO maintains a population of individuals who represent potential solutions, one of the features found in all EAs. If the best personal positions ($y_i$) are treated as part of the population, then clearly there is a weak check (Lee & El-Sharkawi, 2008). In some algorithms of ES, the descendants (*offspring*), parents compete, replacing them if they are more suited. The equation (1) resembles this mechanism, with the difference that the best staff position (the father) can only be replaced by your own current position (descending), provided that the current position is more adapted to the best old staff position. Therefore, it seems to be some weak form check this on the PSO.

The speed update equation resembles arithmetic crossover operator (*crossover*) found in AGs. Typically, the intersection arithmetic produces two descendants that are results of mixing both parents involved in the crossing. The equation of speed update, PSO without term $v_{i,j}$ (see equation 3), can be interpreted as a form of arithmetic crossover involving two parents, returning only one descendant. Alternatively, the update equation of speed, without the term $v_{i,j}$. It can be seen as changing operator.

The best way to analyze the term $v_{i,j}$ is not to think of each iteration as a population replacement process by a new engine (birth and death), but as a process of continuous adaptation (Eberhart and J. Kennedy, 2001). This way the values of $x_i$ are not replaced, but continually adapted using vectors speed $v_i$. This makes the difference between the OSP and the other EAs clearer: the PSO maintains information on the position and velocity (changes in position); In contrast, traditional EAs only keep information on the position.

In spite of the opinion that there is some degree of similarity between the PSO and the majority of other EAs, the PSO has a few features that currently are not present in any other EAs, especially the fact that the PSO models the speed of the particles as well as their positions.

### 3.3 Hybrid Particle Swarm optimization

The Hybrid Particle Swarm algorithm with Mutation (HPSOM) incorporates the mutation process often used in genetic algorithm in PSO (Esmin et al., 2005). This process will allow the particles can escape a local optimum point and perform searches in different area in the search space. This process starts by random choice in Particle Swarm and move to a new different position within the search space. The process of mutation used is given by the following equation:

$$mut(p[k]) = p([k]^* - 1) + \omega \tag{10}$$

Where the $p[k]$ is the randomly chosen particle swarm of and $\omega$ is also obtained from a random order within the following scale: $[0, 0.1(x_{max} - x_{min})]$ representing 0.1 times the length of the search space. The HPSOM algorithm has the following pseudocode.

**begin**
Create and initialise*:*
***While (*** *stop condition is* ***false)***
***begin***
*evalaute*
*update velocity and position*
*mutation*
**end**
**end**

## 4. Integration of meta-heuristic methods with fuzzy control

### 4.1 Advantages of hybrid systems

The integration of fuzzy systems with meta-heuristics methods has some characteristics in common and others that complement each other, as shown in Table 1. The junction of these two techniques forms a proper way to deal with non-linear systems and data. Systems that use these techniques have improved their performance in terms of efficiency and speed of execution.

Fuzzy systems have the advantage of storing knowledge. This is a feature of expert systems so that rules, for example, are easy to modify. Fuzzy systems are an effective and convenient alternative to represent the troubleshooting when the states are well defined. However, for

large and complicated systems, fuzzy systems become difficult to adjust, depending on manual methods that involve trial and error. The fuzzy relation matrix representing the relationships between concepts and actions can be unwieldy, and the best values for the parameters needed to describe the functions of relevance may be difficult to determine. The performance of a diffuse system can be very sensitive to specific values of the parameters.

| | Knowledge Saving | Learning | Optimizing | Speed | Non-Linear Systems |
|---|---|---|---|---|---|
| Fuzzy Systems | ✓ | | | ✓ | ✓ |
| Meta-heuristic Methods | | ✓ | ✓ | ✓ | ✓ |

Table 1. Comparison of characteristics of fuzzy logic with meta-heuristic techniques

In general, meta-heuristic methods offer distinct advantages of optimization of functions of relevance and even learning fuzzy rules. The meta-heuristic methods result in a more comprehensive search, reducing the chance of finishing in a local minimum, through sampling of several solutions sets simultaneously. Fuzzy logic contributes with the evaluation function, stage of genetic algorithm where the adjustment is determined.

There are several possible ways to use meta-heuristic methods with fuzzy systems. A type of hybrid system involves the use of separate modules as part of a global system. The modules based on meta-heuristic methods and fuzzy logic can be grouped singly or with other subsystems of computational intelligent or conventional programs that form an application system.

Another use is the design of systems that are primarily of applications with fuzzy logic. The use of genetic algorithms aims to improve the design process and the performance of the operating system based on fuzzy system. The meta-heuristic methods can be used to discover the best values for functions of relevance when the manual selection of values is difficult or takes a long time.

There are different types of meta-heuristic methods. Among them, genetic algorithms (GA) and particle swarm optimization (PSO), which are used in this chapter. These two methods, more another variation of the PSO, called hybrid PSO (HPSO), are chosen due to their features for integration with other systems. The general procedure for using the meta-heuristic methods with fuzzy systems is shown in Figure 6. For example, a possible solution (represented by a chromosome or a bird) can be defined as a concatenation of the values of all functions of relevance. When the triangular functions are used to represent the functions of relevance, the parameters are the centers for each set widths and fuzzy. An initial range of possible parameter values, the fuzzy system is rotated to determine how much it works well. This information is used to determine the fit of each solution and to establish a new population. The cycle is repeated until you found the best set of values for the parameters of the functions of relevance.

This process can be expanded to use the population that includes information about the conditions and actions corresponding to fuzzy rules. Include them in meta-heuristic treatment allows the system to learn or refine the fuzzy rules.

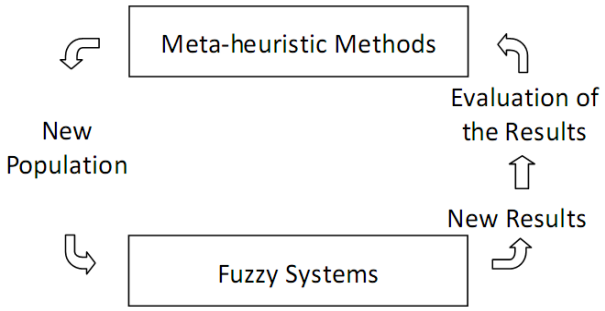



Fig. 9. Overall process for using a meta-heuristic method to improve the performance of a fuzzy system

### 4.2 Description of the training module

The integration of meta-heuristic methods with the fuzzy control has been implemented as follows:

a. the chromosome (or particle) was defined as the concatenation of the adjustment values of the functions of relevance
b. parameters are the centers and the widths of each fuzzy sets. The genes of chromosome (or the particles) are composed by these parameters.
c. a range of possible parameter values, the fuzzy system is rotated to determine how much it works well
d. this information is used to determine the fit of each chromosome or particle (adaptability) and establish a new population, and
e. the cycle is repeated until the number of user-defined generations (or iterations). Each generation (or iteration) is found the best set of values for the parameters of the functions of relevance.

For the meta-heuristic training, many initial positions that the vehicle will start from are defined by the user. Each initial position assesses a sub-population of the chromosomes (or particles) that represents the set of values for the parameters of the functions of relevance, seeking thus an optimization of the control not only over a single trajectory, but all possible starting positions of if from the vehicle to the parking.

After the settings makes by the user, such as number of population, number of generations (or iterations), GA values (rates of crossover, mutation, and son), and PSO values (values of $r_1$, $r_2$, $c_1$, $c_2$, and so on), the adjustment of the fuzzy membership functions starts.

The main idea behind the training is to establish the value of adjustment to the fuzzy membership functions of relevance that is how the function shifted to the left or right and how much it will shrink or expand. It is made by 2 parameters for each membership function, denoted by $k_i$ and $w_i$, for the fuzzy membership function $i$. The value $k$ makes a shift in the membership function, if with negative value to left or if with positive value to right; while the value of $w$ shrink the function for negative values and expand the function for positive values. These values are included in the functions in the following way.

To describe each function of relevance of fuzzy controller are defined four parameters, they are: *LL* (lower left), *LR* (lower right), *UL* (upper left) and *UR* (upper right). In this case, all functions are trapezoids. Figure 10(a) shows the position of each these values. For setting the functions the following equations are used:

$$\begin{aligned} LL_{New} &= (LL_{Old} + k_i) - w_i \\ LR_{New} &= (LR_{Old} + k_i) + w_i \\ UL_{New} &= (UL_{Old} + k_i) \\ UR_{New} &= (UR_{Old} + k_i) \end{aligned} \tag{11}$$

Figure 10(b) shows an example of shifting membership to the following values: $k$ = -8 and $w$ = 2.

$$\begin{aligned} LL_{New} &= (LL_{Old} - 8) - 2 \\ LR_{New} &= (LR_{Old} - 8) + 2 \\ UL_{New} &= (UL_{Old} - 8) \\ UR_{New} &= (UR_{Old} - 8) \end{aligned} \tag{12}$$

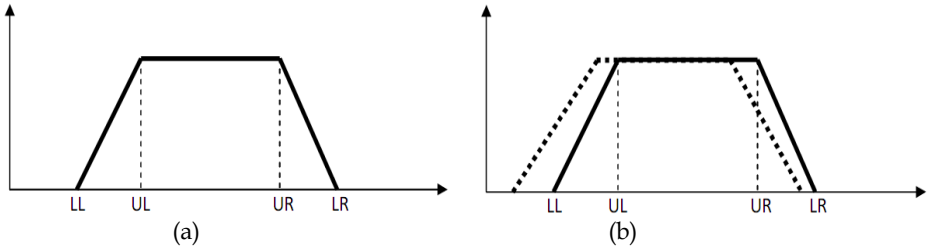



Fig. 10. Typical fuzzy membership function: (a) parameters of relevance, (b) training parameters

Meta-heuristic methods are used to find the optimal values, according to the strategy and starting points used, for $k_i$ and $w_i$ to the functions of relevance.

Usually a viable solution to a problem is associated with an individual (chromosome or particle) $p$ in the form of a $m$ vector with positions $p = \{x_1, x_2, x_3, ..., x_m\}$ where each component $x_i$ represents a gene. Among the types of representation of individuals, the best known are: the binary representation and representation for integers. The binary representation is the classic, as proposed by John Holland (1992). However, for this development the integer code is used to represent each part of the individuals, i.e., each individual is composed by the adjustment coefficients $k_i$ and $w_i$ which are integer values.

With respect to the size of the chromosome, the size of each individual depends on the number of user-defined relevance functions. For a fuzzy control with a group of 18 functions of relevance for example, an individual with 36 variables ($k_i$ and $w_i$ where $i$ = 1, …, 18) is composed.

The population is initialized by setting each part of all individuals to zero (functions given by the user, the coefficients are equal to zero) and the other individuals are initialized with a string of positive or negative integers in a random procedure taken into a range [-10, 10].

The evaluation function has the role to assess the level of fitness (adaptation) of each chromosome generated by algorithms. The problem goal is to minimize the trajectory of the vehicle to be parked. In case the evaluation function is given by:

$$f = \frac{1}{1+I} \tag{13}$$

where $I$ is the total number of iterations until the final position into the park lot. According to the fitness function, the fitness of each chromosome is inversely proportional to the number of iterations.

The integration of meta-heuristic training algorithms with fuzzy model has made as follow:

1. The individual is defined as a link of the membership functions adjustment values.
2. The parameters are the centers and widths of each fuzzy set. These parameters compose the individual.
3. To check the performance of the fuzzy system it is rolled up from an initial set of possible parameters.
4. This information is used for set up each individual adjustment (adaptability) and the making of the evolution of the particle.
5. The cycle repetition is made up to complete the defined meta-heuristic method iteration number made by the user. To each meta-heuristic method iteration the best values set for the membership functions parameters is found.

## 5. Illustrative training examples of fuzzy control

### 5.1 Fuzzy control

This section presents the tests with fuzzy controls that have had their relevance adjusted using meta-heuristic methods. These tests demonstrate the efficiency of such mechanisms, allowing an objective assessment of results found. The original relevance functions are shown in Figure 11. This control has 148 rules, 15 functions relevant for the $x$ and $y$ input variables and car angle, and 7 output functions of angle of the wheel.

Table 2 shows the training results for the fuzzy functions shown in Fig. 11. Three initial positions have been used in this test. This table has the number of iterations that are generated by the vehicle to park using the original relevance functions.

| Position | X | Y | Angle of the car | Iterations without training |
|---|---|---|---|---|
| 1 | 2.5 | 12.0 | 180 | 330 |
| 2 | 16.0 | 13.0 | -90 | 888 |
| 3 | 27.5 | 16.0 | -40 | 655 |

Table 2. Initial positions for training and number of iterations

Figure 12 shows the vehicle in each of the initial positions. These positions were chosen according to the points where the vehicle doesn't develop a good trajectory until park and therefore generating an excessive number of iterations. The main idea is setting several

initial positions will not only minimize the trajectories for these points, but as well as for other points, thus achieving a global minimization of space covered. Figures 13 show the trajectories for each initial position.

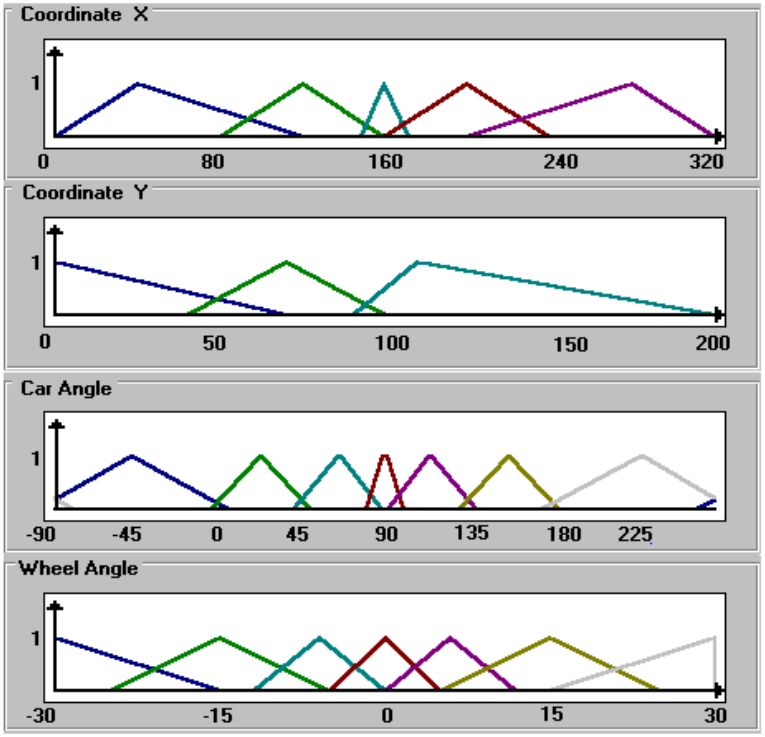



Fig. 11. Original relevance functions

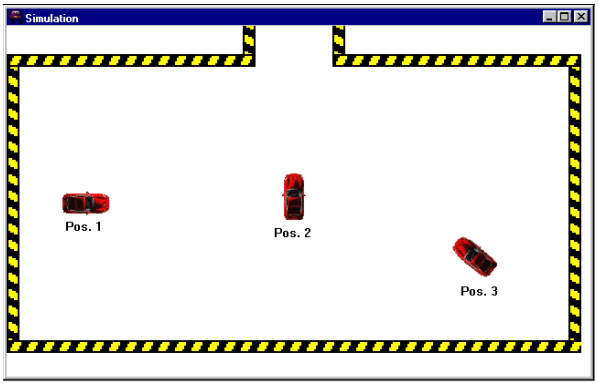
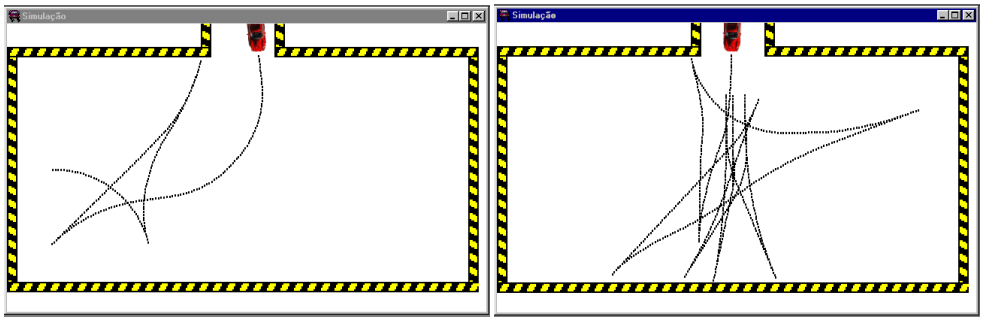
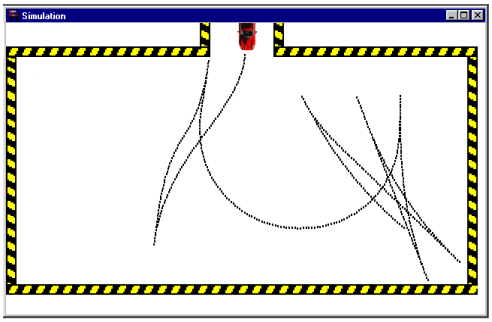



Fig. 12. Initial positions training




(a) (b)




(c)

Fig. 13. Simulation results with fuzzy control without training for the following initial position: (a) position 1, (b) position 2, (c) position3

### 5.2 Meta-heuristic methods training fuzzy control memberships

The definition of several initial positions will not only minimize the routes referred to these points but also for other points, resulting a global minimization of traveled space. The defined GA and PSO parameters for the training are shown in Tables 3 and 4.

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

Table 3. GA parameters

| | |
|---|---|
| Size of Population | 14 |
| Number of Iterations | 30 |
| Vmax | 10 |

Table 4. PSO parameters

After the training if the algorithm described in Section 4.2 with the fuzzy membership functions presented in Figure 11 and for three initial positions presented in Table 2, three

sets of fuzzy membership are computed one for each meta-heuristic method of training (GA, PSO and HPSO). For example, the resultant GA fuzzy membership functions after adjustment are shown in Figure 14.

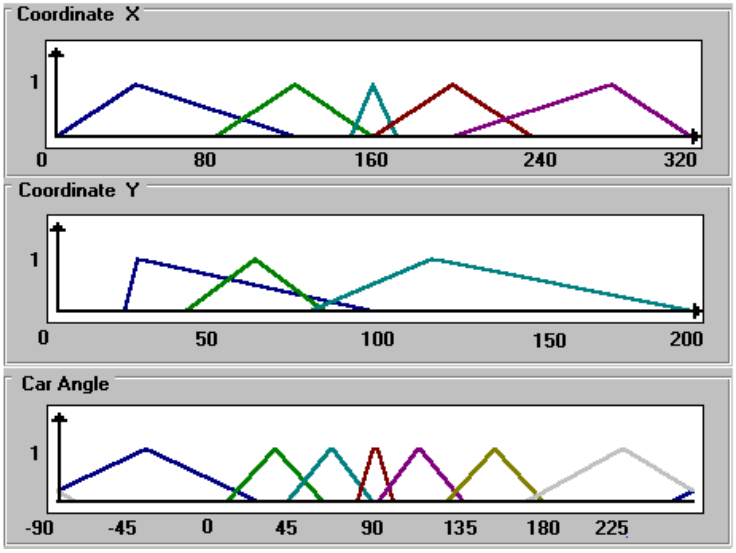



Fig. 14. Membership functions after the genetic algorithm adjustment

The generated results by meta-heuristic methods are shown in Table 5. The reduction of 941 iterations (50,2%) for GA training, 1116 iterations (59,6%) for PSO training, and 930 iterations (49,6%) were made for parking the vehicle starting from the three initial positions. These results are not optimal. Other control setups could be chosen in oder to get better results from these three initila start positions. The idea of theses simulations is presented possible adjustements of the fuzzy memberships. Also, ohter silmulations with other initial positions could create other fuzzy membership functions.

Other kind of possible simulation is to verify the quality of the resultant fuzzy membership for other initial position different from the initial position used in the training. Table 6 presents results of simulations results made starting from initial positions not used in the training for 4 types of adjustments of the fuzzy functions: human setting, and GA, PSO and HPSO trainning methods (from the functions setting by the human). The average of results of meta-heuristic methods are able to improve the better chose of the human being.

| Position | Iterations without training | Iterations with GA training | Iterations with PSO training | Iterations with HPSO training |
|---|---|---|---|---|
| 1 | 330 | 280 | 285 | 278 |
| 2 | 888 | 384 | 592 | 402 |
| 3 | 655 | 277 | 239 | 250 |
| Total | 1873 | 941 | 1116 | 930 |
| Average | 624,33 | 331.67 | 372 | 310 |

Table 5. Iterations after the meta-heuristic training

| Case | X | Y | Car Angle | Iterations generated by Fuzzy Controls | | | |
|---|---|---|---|---|---|---|---|
| | | | | Human Setting | GA Trained | PSO Trained | HPSO Trained |
| 1 | 1 | 126 | 182 | 450 | 329 | 529 | 445 |
| 2 | 6 | 46 | 132 | 167 | 154 | 284 | 303 |
| 3 | 8 | 41 | 190 | 1000 | 1000 | 1000 | 1000 |
| 4 | 10 | 187 | 228 | 453 | 328 | 303 | 291 |
| 5 | 15 | 70 | -90 | 318 | 162 | 282 | 313 |
| 6 | 51 | 112 | 48 | 278 | 130 | 128 | 120 |
| 7 | 51 | 112 | 54 | 280 | 132 | 126 | 120 |
| 8 | 70 | 95 | -40 | 275 | 261 | 465 | 257 |
| 9 | 74 | 69 | 190 | 164 | 164 | 162 | 162 |
| 10 | 76 | 193 | 232 | 605 | 363 | 663 | 363 |
| 11 | 88 | 46 | 44 | 283 | 305 | 475 | 289 |
| 12 | 115 | 120 | 0 | 182 | 280 | 180 | 182 |
| 13 | 120 | 90 | 45 | 182 | 156 | 150 | 146 |
| 14 | 131 | 140 | -72 | 457 | 292 | 592 | 512 |
| 15 | 141 | 69 | -28 | 342 | 314 | 314 | 225 |
| 16 | 154 | 166 | -80 | 863 | 436 | 980 | 420 |
| 17 | 160 | 135 | 268 | 1101 | 545 | 545 | 445 |
| 18 | 161 | 191 | 178 | 315 | 286 | 270 | 266 |
| 19 | 173 | 140 | -72 | 762 | 590 | 580 | 544 |
| 20 | 208 | 143 | 244 | 363 | 310 | 321 | 312 |
| 21 | 217 | 66 | -50 | 684 | 325 | 725 | 506 |
| 22 | 228 | 194 | -48 | 830 | 655 | 855 | 476 |
| 23 | 246 | 169 | 154 | 312 | 307 | 507 | 320 |
| 24 | 250 | 180 | -40 | 739 | 800 | 800 | 489 |
| 25 | 265 | 170 | -40 | 672 | 329 | 629 | 483 |
| 26 | 290 | 95 | -40 | 280 | 190 | 189 | 192 |
| 27 | 300 | 124 | 258 | 317 | 306 | 326 | 319 |
| 28 | 305 | 156 | -90 | 350 | 346 | 340 | 320 |
| 29 | 314 | 73 | -46 | 235 | 355 | 223 | 210 |
| 30 | 314 | 194 | -44 | 513 | 744 | 402 | 388 |
| Average | | | | 459.07 | 363.13 | 444,83 | 347,27 |

Table 6. Results of simulations for different initial position from the used to training

## 6. Conclusion

The fuzzy systems are a convenient and efficient alternative for solution of problems where the fuzzy statements are well defined. Nevertheless, the project of a fuzzy system may became difficult for large and complex systems, when the control quality depends of "try-and-error" methods for defining the best membership functions to solve the problem.

The meta-heuristic method training modulus provides an automatic way for the adjustment of the membership functions parameters. These techniques show that the performance of a fuzzy control may be improved through the genetic algorithms, the particle swarm

optimization or the hybrid particle swarm optimization, substituting for the "try-and-error" method, as used before by students for this purpose, with no good results.

The meta-heuristic methods 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 meta-heuristic methods where the adjustment is settled.

## 7. Acknowledgment

The authors would like to express their thanks to the financial support of this work given by the Brazilian research agencies: CNPq, CAPES, and FAPEMIG.

## 8. References

da Silva Filho, J.I. ; Lambert-Torres, G. & Abe, J.M. (2010). *Uncertainty Treatment using Paraconsistent Logic*, IOS Press, ISBN 978-1-60750-557-0, Amsterdam, The Netherlands.

Eberhart, R. C. & Kennedy, J. (2001). *Swarm Intelligence*, ISBN 1558605959, Morgan Kaufmann, San Francisco, USA.

Eberhart, R. C. ; Simpson, P. & Dobbins, R. (1996). *Computational Intelligence PC Tools*, ISBN 0122286308, Academic Press Professional, Amsterdam, The Netherlands.

Eberhart, R.C. & Kennedy, J. (1995a). A New Optimizer using Particle Swarm Theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, ISBN 0-7803-2676-8, Nagoya, Japan, Oct. 4-6, 1995.

Esmin, A.A.; Lambert-Torres, G. & Souza, A.C.Z. (2005). A Hybrid Particle Swarm Optimization Applied to Loss Power Minimization. *IEEE Transactions on Power Systems*, Vol.20, No.2, (May 2005), pp. 859-866, ISSN 0885-8950.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, ISBN 0201157675, Addison Wesley, Boston, USA.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigam Press, ISBN 0-262-58111-6, Ann Arbor, USA.

Kandel A. & Langholz, G. (1993). *Fuzzy Control Systems*, CRC Press, ISBN 9780849344961, Boca Raton, USA.

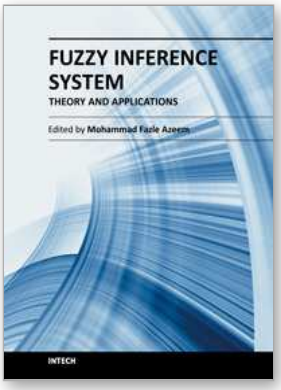
Kennedy, J. & Eberhart, R.C. (1995b). Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, Vol.IV, pp. 1942-1948, ISBN 0852966415, Perth, Australia, June 26-28, 1995.

Lee, K.Y. & El-Sharkawi, M.A. (2008). *Modern Heuristic Optimization Techniques: Applications to Power Systems*, Wiley – IEEE Press, ISBN 0-471-45711-6, Hoboken, USA.

Medsker, L.R. (1995). *Hybrid Intelligent Systems*, Kulwer Academic Pub., ISBN 0792395883, Boston, USA.

Park, D. ; Kandel, A. & Langholz, G. (1994). Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control. *IEEE Transactions on System, Man and Cybernetics*, Vol.24, No.1, (January 1994), pp. 39-47, ISSN 0018-9472.

Ross, T.J. (2010). *Fuzzy Logic with Engineering Applications*, John Wiley and Sons, ISBN 9780470743768, West Sussex, United Kingdom.




**Fuzzy Inference System - Theory and Applications**

Edited by Dr. Mohammad Fazle Azeem

ISBN 978-953-51-0525-1
Hard cover, 504 pages
**Publisher** InTech
**Published online** 09, May, 2012
**Published in print edition** May, 2012

This book is an attempt to accumulate the researches on diverse inter disciplinary field of engineering and management using Fuzzy Inference System (FIS). The book is organized in seven sections with twenty two chapters, covering a wide range of applications. Section I, caters theoretical aspects of FIS in chapter one. Section II, dealing with FIS applications to management related problems and consisting three chapters. Section III, accumulates six chapters to commemorate FIS application to mechanical and industrial engineering problems. Section IV, elaborates FIS application to image processing and cognition problems encompassing four chapters. Section V, describes FIS application to various power system engineering problem in three chapters. Section VI highlights the FIS application to system modeling and control problems and constitutes three chapters. Section VII accommodates two chapters and presents FIS application to civil engineering problem.



**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821