OPTIMIZATION OF FUZZY RULE BASES USING CONTINUOUS ANT COLONY SYSTEM

Hadi Nobahari

Sharif University of Technology
Department of Aerospace Engineering
P.O.Box 11365-8639, Tehran, Iran.
nobahari@mehr.sharif.edu

Seid H. Pourtakdoust

Sharif University of Technology
Department of Aerospace Engineering
P.O.Box 11365-8639, Tehran, Iran
pourtak@sharif.edu

ABSTRACT

The well-known Ant Colony Optimization method is applied to the problem of tuning the parameters of a Takagi-Sugeno fuzzy rule base. The latest developed Continuous Ant Colony System algorithm is used to find the global solution of the continuous optimization problem. The method is systematically tested for the approximation of one analytical test function and the performance of the proposed scheme is compared with the Standard Genetic Algorithm.

1. INTRODUCTION

The concept of Fuzzy Logic (FL) was proposed by Professor Lotfi Zadeh in 1965, at first as a way of processing data by allowing partial set membership rather than crisp membership. Soon after, it was proven to be an excellent choice for many control system applications since it mimics human control logic.

Today, fuzzy rule bases are widely used to express the knowledge of operators or experts in various domains such as decision, control, modeling, identification and so on [1,2]. However there are some limitations to their development, the most important of which is: the knowledge does not always completely exist and the manual tuning of all the base parameters takes time. The lack of portability of the rule bases when the dimensions of the control system change, makes the later difficulty still more serious.

To coop with these problems, the learning methods have been introduced. The first attempt was made by Procyk and Mamdani in 1979, with a "self tuning controller" [3]. The gradient descent method was used by Takagi and Sugeno in 1985 as a learning tool for fuzzy modeling and identification [4]. It was used by Nomura, et al. in 1991 as a self tuning method for fuzzy control [5].

The gradient descent method is appropriate for simple problems and real time learning, since it is fast. But it may be trapped into local minima. Also the calculation of the gradients depends on the shape of membership functions employed, the operators used for fuzzy inferences as well as the selected cost function. So, there is a need for more general and flexible methods capable to search within wide solution spaces and intelligent enough to avoid local minima. These methods must not be linked to a given type of application and they must be applicable to several types of fuzzy rule bases and cost functions.

In 1998, Siarry and Guely applied the well-known Genetic Algorithm to the problem of tuning the parameters of a Takagi-Sugeno fuzzy rule base [6].

In this article, we have integrated this parametric learning problem with our latest developed Continuous Ant Colony System (CACS) [7], which is based on the well-known Ant Colony Optimization (ACO) [8,9]. ACO has been inspired from the ants' *pheromone trail following* behavior. It can find the global minimum of the problem to optimize, and it can handle any cost function.

2. THE TAKAGI-SUGENO FUZZY RULE BASE

Takagi-Sugeno (TS) rules differ from Mamdani rules in that their outputs are not defined by membership functions but by non-fuzzy analytical functions (usually constants). This feature should permit to express complicated knowledge with small number of rules. Unfortunately, TS rules are less intuitive and it is more difficult to translate human expertise into such rules as compared with Mamdani rules. Therefore TS rules have been mostly used till now for modeling purposes and building adaptive controllers [6]. Their wider application is becoming more common, as effective learning techniques are devised.

In this paper, a Takagi-Sugeno fuzzy rule base was used to approximate an m inputs analytical function $f(x)$. The rule base has m inputs, one output, and is composed of n rules. The output of the fuzzy rule base is denoted by $y(x)$. Each rule R_i can be written as

$$R_i : \; if \; (x_1 \; is \; g_{i1}) \; and \; ... (x_j \; is \; g_{ij}) \; and \; ... (x_m \; is \; g_{im})$$
$$then \; y_i = f_i(x_1, ..., x_m)$$

where x_j is the j-th input, g_{ij} the membership function of the i-th rule defined on x_j, the output of rule R_i is y_i, and f_i is the analytical function of the output defined for the i-th rule.

In this paper, rules R_i with constant outputs w_i are considered, so that $y_i = w_i$. But the method is applicable to other types of output functions as well.

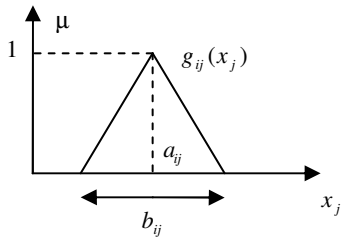

Figure 1. Triangular symmetric membership function

The membership functions g_{ij} are defined as follows (normed triangular symmetric, shown in Fig. 1):

$$g_{ij} = \begin{cases} 1 - 2\left| x_j - a_{ij} \right| / b_{ij} & \text{if } \left| x_j - a_{ij} \right| \le \frac{1}{2} b_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The and operator is the multiplication, so that

$$\mu_i = \prod_{j=1,m} g_{ij}(x_j) \quad (2)$$

where μ_i is the activation degree of the rule R_i.

The output of the rule base is calculated using the center of gravity operator:

$$y(x) = \begin{cases} \dfrac{\sum_{i=1,n} \mu_i w_i}{\sum_{i=1,n} \mu_i} & \text{if } \sum_{i=1,n} \mu_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Let us define the quadratic error *E*:

$$E = \frac{1}{2} \sum_{p=1}^{p_{samp}} \left(y(x_p) - f(x_p) \right)^2 \quad (4)$$

where x_p is the *p*-th training sample and p_{samp} is the number of samples.

The purpose of learning is to minimize the quadratic error *E*. minimization of *E* is performed by tuning the a_{ij}, b_{ij} and w_i parameters. Optimizing the membership function parameters is a complex problem for the following reasons:

i. The cost function is not derivable everywhere.

ii. the cost function is not continuous everywhere (the case when membership functions do not overlap).

iii. numerous parameters have to be optimized (for the case considered in this paper, there is 60 parameter to optimize)

3. ANT COLONY OPTIMIZATION

Ant algorithms were inspired by the observation of the real ant colonies. An important and interesting behavior of ant colonies is their foraging behavior, and in particular, how ants can find the shortest path without using visual cues. While walking from the food sources to the nest and vice versa, ants deposit on the ground a chemical substance called *pheromone* which makes a pheromone trail. Ants use pheromone trails as a medium to communicate with each other. They can smell pheromone and when they choose their way, they tend to choose paths with more pheromone. The pheromone trail allows the ants to find their way back to the food source or to the nest. Also, the other ants can use it to find the location of the food sources, which are previously found by their nest mates.

This pheromone trail following behavior can converge to the shortest path, once employed by a colony of ants. It means that, when there are more paths from the nest to a food source, a colony of ants may be able to use the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back.

Consider two different paths from the nest to the food source with different lengths. Initially there is no pheromone on the two branches, so ants select them with the same probability. Since the ants move at approximately constant speed, at each instant of time the number of ants who have passed the shorter path is greater than the number of ants who have not. Therefore when the ants start their return trip, more pheromone is present on the shorter path, increasing the probability of choosing it. Returning the ants through the shorter path refreshes it faster than the other one and compensates the pheromone evaporation. Thus in this way pheromone is accumulated on the shorter path and for the new ants who want to go to the food source, the probability of choosing it, will increase. Very soon all the ants will be using the shorter path.

3.1. Ant Colony System Basic Features

Ant Colony System (ACS) was one of the first algorithms proposed based on ACO. It was a discrete algorithm, and at first it was applied to the well-known Traveling Salesman Problem (TSP) [8], which is a discrete optimization problem. In this part we will shortly review the basic idea of ACS. Then in the subsequent part, the continuous version of ACS will be presented.

Consider a set of cities. TSP is defined as the problem of finding a minimal cost closed tour that visits all cities and each city only once. In a graph representation, the cities are the nodes and the connection lines between them are the edges. Each edge is associated with a cost measure, which determines the distance or cost of travel between two cities.

Ant colony system uses a graph representation, which is the same as that defined for TSP. In addition to the cost measure, each edge has also a desirability measure, called pheromone intensity, updated at run time by the ants.

Ant colony system works as follows: Each ant generates a complete tour by choosing the cities according to a probabilistic state transition rule. Ants prefer to move to cities, which are connected by short edges with a high amount of pheromone, while in some instances, their selection may be random.

Every time an ant in one city has to choose another city to move to, it samples a random number, q in [0,1]. If q becomes less than a given q_0, then the destination city is chosen by exploitation. It means that the one connected by the edge with the most ratio of pheromone intensity to distance, is chosen. Otherwise a city is chosen by exploration. In this case the one connected by the edge with the most ratio of pheromone intensity to distance, has the most chance to be chosen, but all other cities have also their chances to be chosen proportional to their ratio of pheromone intensity to distance.

While constructing a tour, ants also modify the amount of pheromone on the visited edges by applying a local updating rule. It concurrently simulates the evaporation of the previous pheromone and the accumulation of the new pheromone deposited by the ants when they are building their solutions.

Once all the ants have completed their tours, the amount of pheromone is modified again, by applying a global updating rule. Again a part of pheromone evaporates and all edges that belong to the globally best tour, receive additional pheromone conversely proportional to their length.

3.2. Continuous Ant Colony System Algorithm

A continuous optimization problem is defined as the problem of finding the absolute minimum of a positive non-zero continuous cost function $f(x)$, within a given interval $[a,b]$, which the minimum occurs at a point x_s. In general f can be a multi-variable function, defined on a subset of R^n delimited by n intervals $[a_i, b_i]$, $i = 1,\ldots,n$.

The latest developed Continuous Ant Colony System (CACS) has all the major characteristics of ACS, but certainly in a continuous frame. These are a pheromone distribution over the search space which models the desirability of different regions for the ants, a state transition rule with both exploration and exploitation strategies, and a pheromone updating rule which concurrently simulates pheromone accumulation and pheromone evaporation.

3.2.1. Continuous Pheromone Model

Although pheromone distribution has been first modeled over discrete sets, like the edges of the traveling salesman problem, in the case of real ants, pheromone deposition occurs over a continuous space. Consider a food source, which is surrounded by several ants. The ants' aggregation around the food source causes the most pheromone intensity to occur at the food source position. Then increasing the distance of a sample point from the food source will decrease its pheromone intensity. CACS models this variation of pheromone intensity, in the form of a normal distribution function:

$$\tau(x) = e^{-\frac{(x-x_{\min})^2}{2\sigma^2}} \tag{5}$$

Where $x_{\min}$ is the best point in the interval $[a,b]$ which has been found from the beginning of the trial and σ can be interpreted as an index of the ants aggregation around the current minimum. Note that τ has been modeled as a Probability Distribution Function (PDF) which determines the probability of choosing each point x within the interval [a,b].

3.2.2. State Transition Rule

In CACS, pheromone intensity is modeled using a normal PDF, the center of which is the last best global solution and its variance depends on the aggregation of the promising areas around the best one. So it contains exploitation behavior. In the other hand, a normal PDF permits all points of the search space to be chosen, either close to or far from the current solution. So it also contains exploration behavior. It means that ants can use a random generator with a normal PDF as the state transition rule to choose the next point to move to.

3.2.3 Pheromone Update

Ants choose their destinations through the probabilistic strategy of equation (5). At the first iteration, there isn't any knowledge about the minimum point and the ants choose their destinations only by exploration. It means that they must use a high value of σ (associated with an arbitrary $x_{\min}$) to approximately model a uniform distribution positions.

During each iteration, pheromone distribution over the search space will be updated using the acquired knowledge of the evaluated points by the ants. This process gradually increases the exploitation behavior of the algorithm, while its exploration behavior will decrease. Pheromone updating can be stated as follows: The value of objective function is evaluated for the new selected points by the ants. Then, the best point found from the beginning of the trial is assigned to $x_{\min}$. Also the value of σ is updated based on the evaluated points during the last iteration and the aggregation of those points around $x_{\min}$. To satisfy simultaneously the fitness and aggregation criteria, a concept of weighted variance is defined as follows:

$$\sigma^2 = \frac{\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}} (x_j - x_{\min})^2}{\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}}} \quad , \quad \text{for all } j \text{ in which } f_j \neq f_{\min} \tag{6}$$

where k is the number of ants. This strategy means that the center of region discovered during the subsequent iteration is the last best point and the narrowness of its width is dependent on the aggregation of the other competitors around the best one. The closer the better solutions get (during the last iteration) to the best one, the smaller σ is assigned to the next iteration.

During each iteration, the height of pheromone distribution function increases with respect to the previous iteration and its narrowness decreases. So this strategy concurrently simulates pheromone accumulation over the promising regions and

pheromone evaporation from the others, which are the two major characteristics of ACS pheromone updating rule.

```
procedure CACS()
    initialize()
    while(termination_criterion_not_satisfied)
        move_ants_to_new_locations()
        pheromone_update()
    end while
end procedure

procedure initialize()
    initially guess the best point using a uniform
    random generator
    set the initial values of weighted variances (large
    enough)
end procedure

procedure move_ants_to_new_locations()
    for i=1,k
        for j=1,n
            choose the new x(j) for the ith ant using
            a normal random generator
        end for
    end for
end procedure

procedure pheromone_update
    update the globally best point
    for j=1,n
        update the value of sigma_j
    end for
end procedure
```

Figure 2. The CACS scheme in pseudo-code, k is the number of ants, n is the number of variables to optimize

3.3. Algorithmic Model of CACS

The pseudo-code of CACS scheme is shown in Fig. 2. A high level description of CACS designed to minimize a general multi-variable continuous cost function $f(x_1,...,x_n)$, can be summarized as follows:

Step 1: choose randomly the initially guessed minimum point $(x_1,...,x_n)_{\min}$ over the search space and calculate the value of the function f at this point, namely $f_{\min}$. For each x_i use a uniform PDF over the interval $[a_i,b_i]$.

Step 2: Set the initial value of weighted variance for each pheromone intensity distribution function: $\sigma_i = 3(b_i - a_i)$, $(i = 1,...,n)$. It will be large enough to approximately generate uniformly distributed initial values of x_i within the interval $[a_i,b_i]$.

Step 3: Send ants to points $(x_1,...,x_n)_j$, $j = 1,...,k$. To generate these random locations, a random generator with normal PDF is utilized for each x_i, where its mean and variance are $(x_i)_{\min}$ and σ_i respectively. Note that if a generated x_i is outside the given interval $[a_i,b_i]$, it is discarded and a new x_i is regenerated again.

Step 4: Evaluate *f*, at each discovered point, namely $f_1,...,f_k$. Compare these values with the current minimum value $f_{\min}$ and determine the updated $f_{\min}$ and its associated $(x_1,...,x_n)_{\min}$.

Step 5: if a stopping criterion is satisfied (for the case which is examined in this paper, the algorithm stops after 200000 evaluations) then **stop**, else update the weighted variance parameter σ_i for each variable x_i using equation (7) and go back to **step 3**.

$$\sigma_i^2 = \frac{\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}} [(x_i)_j - (x_i)_{\min}]^2}{\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}}} \qquad (7)$$

4. EXPERIMENTS

The same problem as [6] is considered to evaluate the learning method and to compare the results with those of GA. The problem is approximating the following analytical function with a fuzzy rule base.

$$f(x) = \begin{cases} -9x^2 + 3x + 1/3 & x \in [0,1/3] \\ 2x - 1/3 & x \in [1/3,1/2] \\ 0.03/(0.03 + (4.5x - 3.85)^2) & x \in [0,1/3] \end{cases} \qquad (8)$$

It presents the following properties (Fig. 3):

i. A discontinuity of the derivative at x=1/3
ii. A discontinuity at x=1/2
iii. A peak with a high curvature zone around x=0.85

As [6], a Takagi-Sugeno fuzzy rule base is considered with 20 rules. Here the learning process means to find the optimal values of the a_i, b_i and w_i $(i = 1,...,20)$. Therefore the number of unknowns is 60.

The optimization problem is to minimize the quadratic error *E*, which is calculated with a set of testing samples. The test samples are numerous enough and regularly distributed in the interval on which the analytical function is defined.

To make the results comparable with those of [6], 100 sample points were taken and a limitation of 200000 evaluations of the quadratic error *E*, per run, was imposed.

A typical result is shown in Fig. 3, obtained after 200000 evaluations of *E,* using 10 ants. The best value of *E* was $E_{optimum} = 0.0098$. It is clear from this figure that $f(x)$ is well approximated, but that the obtained membership functions seem rather difficult to interpret in a linguistic way. The corresponding values of a_i, b_i and w_i are listed in table 1.

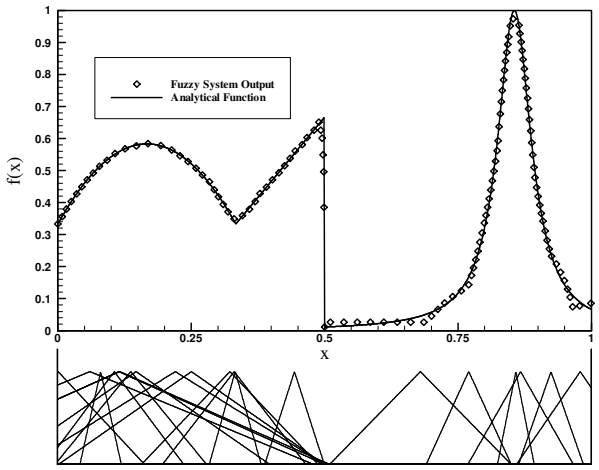

Figure 3. *Typical approximation obtained and the associated membership functions.*

Table 1. *Typical Solution for* a_i, b_i *and* w_i

i	a_i	b_i	w_i
1	8.6764e-001	1.9741e-001	9.4789e-001
2	2.2114e-001	5.5777e-001	9.7425e-001
3	1.1545e-001	7.6909e-001	9.9999e-001
4	4.4422e-001	1.1154e-001	6.5026e-001
5	8.5869e-001	6.8881e-002	1.0000e+000
6	9.7924e-001	2.3847e-001	8.5670e-002
7	7.7044e-001	1.5910e-001	1.9014e-001
8	1.0698e-001	2.5976e-001	8.2945e-001
9	1.1711e-001	7.8577e-001	7.7188e-003
10	3.3332e-001	9.7053e-002	1.4092e-002
11	6.7999e-001	3.3998e-001	2.6516e-002
12	3.3141e-001	2.6843e-001	1.6909e-001
13	1.3771e-001	2.8344e-001	9.9999e-001
14	6.1318e-002	8.3735e-001	1.8681e-001
15	9.2448e-001	1.2144e-001	1.0708e-002
16	1.8369e-003	3.2331e-001	1.0901e-001
17	1.4764e-001	5.0060e-001	1.4603e-002
18	2.5121e-001	4.9751e-001	9.1718e-001
19	3.2334e-001	3.3079e-001	7.6551e-002
20	8.1307e-002	7.7487e-002	5.4206e-001

4.1. Parameter Study

CACS has only one parameter to set which is the number of ants, *k*. According to [7], CACS needs at least 10 ants to propose relatively good results. The variation of $E_{\min}$ versus the number of evaluations is shown in Fig. 4 and Fig. 5. The later one is concentrated over the first 5000 evaluations to observe more precisely the initial behavior. Also table 2 gives the results obtained with different number of ants $(k = 5,10,20,50)$. All presented results are averaged over 10 different runs. The columns report the average values, while the numbers in the parenthesis are the standard deviations. The notations used in the subsequent tables, are defined as follows:

E_{best} : Minimum value of E obtained over all evaluated points by the ants (It allows to evaluate the *convergence quality*)

$Eval(E_{best})$: Number of evaluations at which E_{best} was obtained.

$Eval(E_{ave}/10)$: Number of evaluations at which the average error is below tenth of its initial value at the first iteration (It allows to evaluate the convergence speed at the beginning)

Table 2. *Averaged results vs. the number of ants*

k	E_{best}	$Eval(E_{best})$	$Eval(E_{ave}/10)$
5	0.6032	36980	3184
	(0.0678)	(5938)	(2023)
10	0.0098	126740	1042
	(0.0053)	(16151)	(141)
20	0.0244	173851	4738
	(0.0082)	(9048)	(1318)
50	0.1573	155130	41708
	(0.0271)	(12431)	(18758)

The presented results of Fig. 4, Fig. 5 and table 2, show that, the size of the population of ants has a clear effect on the observed results. As mentioned in [7], CACS is not responsive enough for $k < 10$. Small number of ants (e.g. $k = 5$) yields a fast convergence speed at the first evaluations, but at the same time it may cause an unripe convergence to a local minimum (Fig. 4 and $Eval(E_{best})$ in table 2). In the case of small number of ants the probability of falling into a local minimum increases because it is very probable that during one iteration all ants choose the points near to the current minimum. This in turn decreases the values of the weighted variances, which magnify the aggregation of the later iteration points around the local minimum. When the value of the weighted variances become several times smaller than the distance to the nearest local minimum, it's no longer possible to jump to a better local minimum and the minimization process stops.

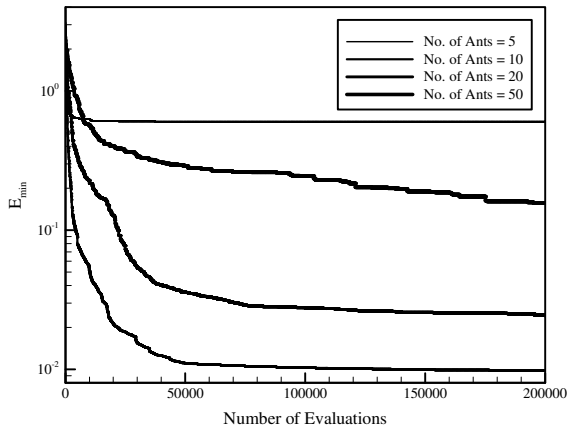

Figure 4. *Minima found vs. number of evaluations (the effect of the number of ants).*

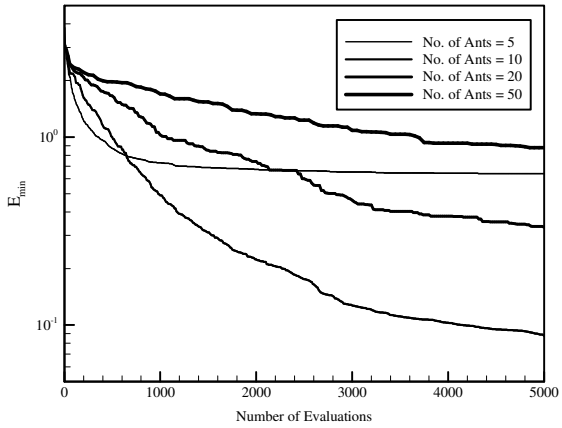

Figure 5. *Minima found during the first 5000 evaluations (the effect of the number of ants).*

The best results are observed for $k = 10$ (see Fig. 4 and the obtained E_{best} and $Eval(E_{ave}/10)$ in table 2). The convergence rate decreases for greater number of ants (with 20 ants, $E_{opt} \approx 0.02$, while $E_{opt} < 0.01$ with 10 ants).

4.2. Comparison with Standard Genetic Algorithm

In order to compare the overall performance of CACS with other algorithms, the results obtained for $k = 10$ are listed in table 3, and have been compared with the well-known Genetic Algorithm of [6], which utilizes the Standard Genetic Algorithm (SGA) with 500 chromosomes, a mutation rate of 0.005, a crossover rate 0.6, 8 bit for coding fuzzy parameters and 400 generations.

Table 3 shows that the best minima found are at the same order, but a little better for SGA than CACS. At the same time comparing $Eval(E_{best})$ and $Eval(E_{ave}/10)$ between two methods, reveals a better convergence speed for CACS than SGA.

A key point in CACS is its simplicity. One can easily find that implementation and use of CACS is very simpler than SGA, which is due to its simpler structure and its fewer control parameters.

Table 3. *Comparison of CACS and GA* results

Method	E_{best}	$Eval(E_{best})$	$Eval(E_{ave}/10)$
CACS	0.0098	126740	1042
	(0.0053)	(5938)	(2023)
SGA	0.0067	183000	21000
	(0.0013)	(11683)	(1871)

5. CONCLUSIONS

A recently developed Continuous Ant Colony System algorithm, based on Ant Colony Optimization, is applied to the problem of finding optimum parameters of a fuzzy rule base. The proposed scheme is used to approximate an analytical test function. The performance of CACS is compared with the Standard Genetic Algorithm. The overall results are compatible. Although SGA can find the minimum slightly better, but it has poorer convergence rate compared with CACS.

One of the best advantages of the CACS scheme is its simplicity, which is mainly due to its simpler structure. Also it has fewer control parameters, which makes the parameter settings process easier than many other methods.

6. REFERENCES

[1] Wang, L. X., *A Course in Fuzzy Systems and Control,* Pearson Education POD, 1997.

[2] Driankov, D. and Palm, R., *Advances in Fuzzy Control,* Heidelberg: Physica-Verlag, 1998.

[3] Procyk, T. J. and Mamdani, E. H., "A linguistic self-organizing process controller," *Automatica, IFAC, Vol. 15, 1979, 15-30.*

[4] Takagi, T. and Sugeno, M., "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Systems, Man and Cybernetics, Vol. 15, 1985, 116-132.*

[5] Nomura, H., Hayashi, I., Wakami, N., "A Self Tuning Method of Fuzzy Control by Descent Method," *Proceedings of the International Fuzzy Systems Assiciation, IFSA91, Engineering Vol., Bruxelles, 1991, 155-158.*

[6] Siarry, P. and Guely, F., "A Genetic Algorithm for Optimizing Takagi-Sugeno Fuzzy Rule Bases," *Fuzzy Sets and Systems, Vol. 99, 1998, 37-47.*

[7] Pourtakdoust, S. H. and Nobahari, H., "An Extension of Ant Colony System to continuous optimization problems," *Lecture Notes in Computer Science*, *Vol. 3172, 2004, 294-301.*

[8] Colorni, A., Dorigo, M. and Maniezzo, V., "Distributed optimization by ant colonies," *Proceedings of the First European Conference on Artifcial Life, Elsevier Science Publisher, 1992, 134–142.*

[9] Dorigo M. and Gambardella, L. M., "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, 1997, 53–66*.