Tuning fuzzy systems parameters with chaotic particle swarm optimization

I A Hodashinsky¹, M B Bardamova¹

¹ Tomsk State University of Control Systems and Radioelectronics, 40, Lenina Ave., Tomsk, 634050, Russia

E-mail: hodashn@rambler.ru

Abstract. In this paper, we investigate the application of chaotic particle swarm optimization (PSO) to fuzzy system parameter estimation. Unlike traditional PSO, chaotic PSO improves search capabilities of particles using chaotic transformations of random coordinates. Different mapping functions were investigated to generate chaotic transformation sequences. The efficiency of the algorithm was tested on approximation problems originating from data sets of the KEEL repository. The experimental study compared several existing techniques for fuzzy systems identification. The results indicate that chaotic PSO is a competitive approach to fuzzy system identification able to create fuzzy systems of high precision at an acceptable set of rules.

1. Introduction

Fuzzy Systems (FSs) have been successfully applied to problems in classification, clustering, modeling, control, decision making and other applications [1].

The main challenge in the development of FS is the generation of a fuzzy rule-base. The design of FS can be considered as a non-linear optimization problem in a multi-dimensional space featuring a complex (fitness landscape) structure. Due to the complexity of the problem, many authors proposed the use metaheuristics, such as evolutionary algorithms (EAs) and swarm methods, to tackle the problem. Whilst EAs, particularly genetic algorithms, have been applied successfully to the optimization of FS [2, 3], the application of swarm algorithms, such as ant colony algorithms [4, 5] and particle swarm optimization (PSO) [6, 7], is not so common. The focus of this paper is on the application of a variant of a PSO algorithm to address the problem of FS parameter tuning. PSO algorithms share similar features to EAs, e.g. both algorithm types do not require gradient information of the function being optimized, they can deal robustly with search landscapes of different topology, and are fairly easy to implement in any standard programming language. However, PSO tends to converge faster to high quality solutions than EAs, making it more suitable to solve optimization problems in real-time [8]. Similar to EAs, PSO can be hybridized easily with other approaches and has found application in various domains; the interested reader can be referred to [9] for a list of 26 application domains of PSO.

However, due to the fast convergence, the risk of getting stuck at local optima is higher than with many other EAs. Different approaches have been proposed to deal with the convergence speed vs solution quality trade-off including modifying the parameters of a PSO algorithm during the optimization process and the application of diversity-enhancing schemes. Recently, the chaos concept
has been introduced to PSO to help control the rapid convergence and improve search capabilities of particles [10].

Chaotic PSO has found application in different types of problems including those with discrete and continuous decision variables. For example, in [11], the algorithm is used to plan the trajectory of a mobile robot, whilst in [12], it is used to solve the traveling salesman problem. In [13, 14], a binary chaotic PSO algorithm is used in conjunction with a nearest neighbor algorithm for selecting attributes in a classification problem. In this paper, we extend the list of applications of chaotic PSO by the optimization of FS parameters, particularly membership functions parameters.

The paper is organized as follows. Section 2 outlines the problem of FS identification and singleton-type FSs, which are used here as an example to demonstrate the performance of chaotic PSO. The basic idea of traditional PSO is described in Section 3. In Section 4, several mapping functions for the generation of chaotic transformation sequences are presented, and the chaotic PSO algorithm is described. In Section 5, the performance of the algorithm is validated using several approximation problems. Finally, Section 6 concludes the paper.

2. Singleton-type fuzzy systems

The main feature of FS is that they are based on the concept of fuzzy representation of information and operate with fuzzy sets instead of numbers.

We consider singleton type FSs, where the i-th rule has the following form:

\[\text{IF } x_1 = A_{1i} \text{ AND } x_2 = A_{2i} \text{ AND } \ldots \text{ AND } x_n = A_{ni} \text{ THEN } y = r_i,\]

where \(A_{ji}\) is a linguistic term estimating variable \(x_j\), and \(r_i\) is a real number estimating output \(y\).

The FS performs mapping \(f : \mathbb{R}^n \rightarrow \mathbb{R}\) defined by the following formula:

\[f(x) = \frac{\sum_{i=1}^{R} \prod_{j=1}^{n} \mu_{A_{ji}}(x_j)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \mu_{A_{ji}}(x_j)},\]

where \(x = (x_1, x_2, \ldots, x_n)\) is an input vector, \(n\) the number of input variables, \(R\) the number of rules, and \(\mu_{A_{ji}}(x_j) \in [0,1]\) is a membership function defining the membership degree of the \(j\)-th input variable to term \(A_{ji}\).

The FS can be represented as follows:

\[y = f(x, \theta),\]

where \(\theta = (\theta_1, \ldots, \theta_N)\) is a vector of parameters, and \(N\) is a total number of parameters identified by the system.

Given a set of training data (a table of observations) \((x_p, t_p), \ p = 1, \ldots, m\)\), where \(x_p\) is an input vector, \(t_p\) is the output value, and \(m\) is the number of observations, the mean square error (MSE) of a FS is calculated according to

\[E(\theta) = \frac{1}{m} \sqrt{\sum_{p=1}^{m} (t_p - f(x_p, \theta))^2}.\] (1)

FS identification includes two main steps: structure identification and parameter estimation. Structure identification is concerned with determining the structure characteristics of a FS, such as the number of fuzzy rules and linguistic terms for each variable. Parameter estimation involves the estimation of the parameters of the membership functions and consequents of fuzzy rules. The paper focuses on parameter estimation, in particular, on estimating membership functions parameters.
The problem of parameter estimation can be solved as a problem of finding a minimum of given function $E(\theta)$ in the multidimensional space defined by a set of coordinates corresponding to the FS parameters.

In order to solve the given minimization problem, it is suggested to use a chaotic particle swarm algorithm.

3. Traditional Particle Swarm Optimization

PSO is a population based stochastic optimization method developed by Kennedy and Eberhart [15]. It is based on observations of the social behavior of animals, such as birds or fish, searching for food.

The particles (conceptual mathematical objects) are located in the space defined by the parameters of the problem being solved. Each position of the particle is characterized by its fitness. In traditional PSO, there are three factors that define the movement of a particle in the search space: inertia, memory and cooperation. Inertia means that a particle cannot change its direction of motion immediately. Each position of the particle is characterized by its fitness. In traditional parameters.

Chaotic PSO extends traditional PSO with the principles of chaotic dynamics, which come into play by computing $x(k+1)$ based on chaotic sequences. The idea is that chaotic dynamics improves the diversity of a swarm and thus reduces the risk of getting stuck at local optima [10, 16]. Given $cx_i(k)$ is the $i$-th chaotic variable at iteration $k$, then the set of commonly used mapping functions to generate chaotic sequences includes:

$$x_{i}(k + 1) = x_{i}(k) + v_{i}(k+1),$$

where $i = 1, 2, \ldots, M$; $x_{i}(k)$ is the position of the $i$-th particle at iteration $k$; $v_{i}(k)$ is the velocity vector of the $i$-th particle at iteration $k$; $c_1$ and $c_2$ are positive speedup factors; $p_{i}(k)$ is the best position of the $i$-th particle so far (i.e. personal best position); $p_{g}(k)$ is the position of the best particle in the swarm (i.e. global best position); $w$ is the inertia factor; $rand$ and $Rand$ are random numbers drawn from interval [0, 1].

Coefficient $c_1$ is a cognitive parameter, which indicates the trust of a particle to its own past experience; this coefficient is responsible for the discovery of new areas in the search space. Coefficient $c_2$ is a social parameter, showing how the particle trusts the swarm; this coefficient is responsible for the exploration of promising search space areas.
a) Logistic mapping [10]
\[ cx_{i}(k+1) = 4 \cdot cx_{i}(k) \cdot (1 - cx_{i}(k)), \text{ where } cx_{i}(0) = 0.2027; \]

b) Sinusoidal iterator [17]
\[ cx_{i}(k + 1) = 2.3 \cdot cx_{i}^{2}(k) \cdot \sin(\pi \cdot cx_{i}(k)), \text{ where } cx_{i}(0) = 0.7; \]

c) Tent mapping [17]
\[
\begin{align*}
\text{if } cx_{i}(k) < 0.7 & : cx_{i}(k+1) = cx_{i}(k) \cdot \frac{0.7}{1-cx_{i}(k)}, \\
\text{if } cx_{i}(k) \geq 0.7 & : cx_{i}(k+1) = cx_{i}(k) \cdot \frac{0.3}{1-cx_{i}(k)}.
\end{align*}
\]

d) Gauss mapping [17]
\[ cx_{i}(k + 1) = \begin{cases} 
0, & \text{if } cx_{i}(k) = 0 \\
1/cx_{i}(k) & , & \text{if } cx_{i}(k) \in (0,1) 
\end{cases}. \]

A chaotic transformation of variable \( x \) is performed using the following procedure.

Step 1. Calculate the value of chaotic variable \( cx_{i} \) by normalizing the value of \( x_{i} \),
\[
\begin{align*}
ct_{i}(k) &= \frac{x_{i}(k) - x_{i,\text{lower}}(k)}{x_{i,\text{upper}}(k) - x_{i,\text{lower}}(k)}, \\
\end{align*}
\]
where \( x_{i,\text{lower}}, x_{i,\text{upper}} \) are minimum and maximum values of \( x_{i} \).

Step 2. Apply the chaotic transformation to variable \( cx_{i}(k) \), based on the selected chaotic sequence.

Step 3. Re-normalize the value of resulting variable \( cx_{i}(k+1) \)
\[ x_{i}(k + 1) = x_{i,\text{lower}}(k) + cx_{i}(k + 1) \cdot (x_{i,\text{upper}}(k) - x_{i,\text{lower}}(k)). \]

Factor \( \text{Score}_{i} \) is introduced for each \( i \)-th particle in the chaotic PSO
\[ \text{Score}_{i}(k) = \frac{E(x_{i,\text{worst}}(k)) - E(x_{i}(k))}{E(x_{i,\text{worst}}(k)) - E(p_{g}(k))}, \]
where \( x_{i,\text{worst}}(k) \) is the worst position of the \( i \)-th particle within the first \( k \) iterations.

Factor \( \text{Score}_{i} \) affects the inertia coefficient, which is calculated for each particle individually as follows (let us note that the inertia is fixed in the traditional PSO):
\[ \text{w}_{i}(k) = \text{w}_{i,\text{low}}(k) + (\text{w}_{i,\text{high}}(k) - \text{w}_{i,\text{low}}(k)) \cdot (1 - \text{Score}_{i}(k)), \]
where \( \text{w}_{i,\text{low}}(k) \) and \( \text{w}_{i,\text{high}}(k) \) are the lowest and highest inertia values of the \( i \)-th particle within the first \( k \) iterations.

The chaotic PSO algorithm is presented below.

Step 1. Set the parameters of the algorithm: a maximum number of iterations \( K \), swarm size \( M \), \( c_{1}, c_{2}, w(0), v(0) = 0, x(0), p_{g}(0), k = 1 \).

Step 2. Evaluate the fitness value of each particle in the swarm by calculating the MSE values (see Equation (1)). If \( k = K \), then go to Step 6.

Step 3. Perform a chaotic transformation (shift) of randomly chosen coordinates of randomly selected particles using a given chaotic sequence.

Step 4. Define \( \text{Score}_{i}(k+1) \), inertia \( \text{w}_{i}(k+1) \), velocity vector \( \text{v}_{i}(k+1) \) and new coordinates \( x_{i}(k+1) \) for each \( i \)-th particle.
Step 5. Calculate the best new positions of each particle $p_i(k+1)$. Redefine the best position in swarm $p_g(k+1)$. $k = k+1$. Go to Step 2.

Step 6. Return $p_g(k+1)$.

5. Experiments

5.1. Experimental setup.

The performance of the chaotic PSO algorithm was validated on four data sets of the KEEL (Knowledge Extraction Evolutionary Learning) repository [18]: diabetes, ele-1, ele-2, and dee. The objective of the experiments was to minimize the MSE (see Equation (1)).

The experiments were conducted according to the following procedure proposed in the repository: First, each data set from the repository was partitioned using 5-fold cross validation. For each split of training (tra) and testing (tst) data set, and each chaotic sequence type presented above, 5 algorithmic runs of the chaotic PSO algorithm were performed (i.e. in total, 25 runs were performed for each data set and chaotic sequence type).

FSs were learnt (by the chaotic PSO algorithm) using training data sets, and then validated on the testing data sets.

Triangular membership functions are considered for each input variable $x_j$. The following default parameters were used for the chaotic PSO algorithm: $c_1 = c_2 = 1.4, w(0) = 0.4, K = 1000, M = 20$. Average MSE values and their standard deviation were calculated for both training and testing data sets. The data sets considered, and FS setups employed for each data set, are detailed further.

**diabetes:** This data set has two real-type input variables and consists of 43 observations. The data was obtained from a study investigating the factors affecting the model of insulin-dependent diabetes mellitus in children. The study analyzes the dependency of the serum C-peptide level on various other factors to understand patterns of residual insulin secretion. The output is the concentration of C-peptide, and the inputs are alkalipenia and a patient’s age [18]. The following FS parameters were used for this problem: 4 fuzzy rules, and 2 terms for each input variable.

**ele-1:** This data set has two input variables and consists of 495 observations. The data investigate the problem of forecasting the total length of low voltage lines (output) as a function of the number of inhabitants and the distance from the city center to the most distant customers (inputs) [18]. The following FS parameters were used for this problem: 16 fuzzy rules, and 4 terms for each input variable.

**ele-2:** This data set has 4 input variables and consists of 1056 observations. The data aim at estimating electrical maintenance costs of a city (output) as a function of the total length of all streets combined, the area of the city, built up area of the city, and the energy capacity required [18]. The following FS parameters were used for this problem: 81 fuzzy rules, and 3 terms for each input variable.

**dee:** This data set has 6 input variables and consists of 365 observations. The data aim at predicting the average daily consumption of electricity in Spain (output) as a function of the daily energy consumption of hydroelectric, nuclear electric, carbon, fuel, natural gas, and other sources of energy in Spain in 2003 [18]. The following FS parameters were used for this problem: 64 fuzzy rules, and 2 terms for each input variable.

5.2. Experimental results

Table 1 shows the average MSE and its standard deviation (for training and testing data sets) obtained by the two best-performing chaotic sequences on each data set.

The performance of the chaotic PSO algorithm has been compared against the methods described by [19]. Tables 2, 3, 4 and 5 compare the results of these algorithms.

For each data set, the chaotic PSO algorithm used the best-performing chaotic sequence as shown in Table 1.
From the tables, it can be concluded that chaotic PSO is able to create FSs of high accuracy using an acceptable number of rules. This indicates a good degree of interpretability of the obtained systems. The good MSE results on testing data sets $MSE_{(tst)}$ indicate that chaotic PSO has a high predictive capacity. In fact, the algorithm performs best on all testing data sets, except for data set diabetes. Here, the algorithm featured the second-best average MSE, but this performance was achieved using fewer rules than the other algorithms. However, it is worthwhile mentioning that on the diabetes data set, chaotic PSO performed also poorly on the training data set. Conversely, on the data sets, ele-1 and ele-2, chaotic PSO achieves the best learning capacity, which translates also to the best predictive ability. Finally, chaotic PSO is also able to perform well on the dee data set: the outcome achieved was the third-best result on the training data sets, and the best on the testing data set, using the second fewest rules of all algorithms compared.

**Table 1.** Average MSE and its standard deviation (for training and testing data sets) obtained by the two best-performing chaotic sequences on each data set.

| Chaotic sequence    | $MSE_{(tra)}$ (average, $\sigma$) | $MSE_{(tst)}$ (average, $\sigma$) |
|---------------------|-----------------------------------|-----------------------------------|
| diabetes (4 rules)  |                                   |                                   |
| Tent Map            | 0.18525 (0.0161)                  | 0.37926 (0.0876)                  |
| Sinusoidal Iterator | 0.19000 (0.0154)                  | 0.36000 (0.0811)                  |
| ele-1 (16 rules)    |                                   |                                   |
| Logistic Mapping    | 317058 (15905)                    | 389927 (28075)                    |
| Tent Mapping        | 309449 (12306)                    | 388291 (23602)                    |
| ele-2 (81 rules)    |                                   |                                   |
| Logistic Map        | 16616 (1355)                      | 17832 (1307)                      |
| Tent Map            | 16806 (1633)                      | 18409 (2109)                      |
| dee (64 rules)      |                                   |                                   |
| Logistic Map        | 0.14023 (0.0051)                  | 0.17003 (0.0153)                  |
| Tent Map            | 0.13901 (0.0060)                  | 0.16855 (0.0133)                  |

**Table 2.** Comparison of the number of rules and average MSE, and their standard deviation (for training and testing data sets) obtained by chaotic PSO and other existing algorithms on data set diabetes.

| Algorithm          | number of rules | $MSE_{(tra)}$ (average, $\sigma$) | $MSE_{(tst)}$ (average, $\sigma$) |
|--------------------|-----------------|-----------------------------------|-----------------------------------|
| Chaotic            | 4               | 0.18525 (0.0161)                  | 0.37925 (0.0876)                  |
| Wang–Mendel        | 18.6 1.4        | 0.22836 (0.0425)                  | 1.40241 (0.6890)                  |
| COR-BWAS           | 18.6 1.4        | 0.17496 (0.0250)                  | 1.45869 (0.7091)                  |
| Thrift             | 46.2 0.7        | 0.07448 (0.0098)                  | 0.87825 (0.3575)                  |
| Pittsburgh         | 15.0 2.9        | 0.10398 (0.0182)                  | 0.95088 (0.7881)                  |
| Fuzzy-GAP          | 10.0 0.0        | 0.14292 (0.0376)                  | 0.50141 (0.3014)                  |
| Pitts-DNF med      | 5.4 0.5         | 0.12958 (0.0136)                  | 0.32134 (0.1922)                  |
| Pitts-DNF max      | 9.6 1.2         | 0.10656 (0.0150)                  | 0.63396 (0.5276)                  |
**Table 3.** Comparison of the number of rules and average MSE, and their standard deviation (for training and testing data sets) obtained by chaotic PSO and other existing algorithms on data set ele-1.

| Algorithm      | number of rules of MSE(tra)  |   | MSE(tst) |   |
|----------------|------------------------------|---|----------|---|
|                | average σ average σ | average σ | average σ | average σ |
| Chaotic        | 16 0 309449 12306 | 388291 | 23602 |
| Wang–Mendel    | 22.0 1.4 423466 8069 | 455262 | 19943 |
| COR-BWAS       | 22.0 1.4 354304 7065 | 417142 | 9823 |
| Thrift         | 46.4 1.0 335086 5285 | 435373 | 57252 |
| Pittsburgh     | 17.2 4.3 342464 19209 | 738691 | 543165 |
| Fuzzy-GAP      | 11 0.0 481603 58989 | 548122 | 70968 |
| Pitts-DNF med  | 8.2 0.7 344636 8999 | 415266 | 71200 |
| Pitts-DNF max  | 14.0 1.1 330496 4815 | 440692 | 40370 |

**Table 4.** Comparison of the number of rules and average MSE, and their standard deviation (for training and testing data sets) obtained by chaotic PSO and other existing algorithms on data set ele-2

| Algorithm      | number of rules of MSE(tra)  |   | MSE(tst) |   |
|----------------|------------------------------|---|----------|---|
|                | average σ average σ | average σ | average σ | average σ |
| Chaotic        | 81 0 16616 1355 | 17832 | 1307 |
| Wang–Mendel    | 65.0 0.0 112270 1498 | 112718 | 4685 |
| COR-BWAS       | 65.0 0.0 102664 1080 | 102740 | 4321 |
| Thrift         | 524.6 6.4 146305 12991 | 168472 | 20135 |
| Pittsburgh     | 240.0 21.1 210717 32027 | 265130 | 30161 |
| Fuzzy-GAP      | 33.0 0.0 279166 90017 | 290062 | 89155 |
| Pitts-DNF med  | 18.6 1.4 86930 3955 | 99310 | 12996 |
| Pitts-DNF max  | 32.4 6.6 70207 1658 | 88017 | 8968 |

**Table 5.** Comparison of the number of rules and average MSE, and their standard deviation (for training and testing data sets) obtained by chaotic PSO and other existing algorithms on data set dee.

| Algorithm      | number of rules of MSE(tra)  |   | MSE(tst) |   |
|----------------|------------------------------|---|----------|---|
|                | average σ average σ | average σ | average σ | average σ |
| Chaotic        | 64 0 0.13900 0.0059 | 0.16855 | 0.0132 |
| Wang–Mendel    | 178 2 0.14117 0.0074 | 0.22064 | 0.0264 |
| COR-BWAS       | 178 2 0.12463 0.0052 | 0.20513 | 0.0231 |
| Thrift         | 13020 33 0.38778 0.0357 | 0.45830 | 0.0804 |
| Pittsburgh     | 982 56 0.42111 0.0784 | 0.72109 | 0.3263 |
| Fuzzy-GAP      | 89 0 0.17751 0.0130 | 0.20633 | 0.0172 |
| Pitts-DNF med  | 57 3 0.13821 0.0060 | 0.27465 | 0.1366 |
| Pitts-DNF max  | 98 5 0.11267 0.0035 | 0.21692 | 0.0359 |
6. Conclusion
In this paper, we validated an efficient algorithm applied to the optimization of FS parameters. Experiments conducted on several benchmark data sets, as well as a comparative analysis with various existing techniques showed that the proposed algorithm is able to create FSs that are highly accurate whilst using few rules.
The algorithm has a good learning ability, meaning MSE values on test data sets are not far off from the MSE values on training data sets. This fact indicates the absence of overtraining.

Acknowledgments
The project was carried out with financial support from the Russian Foundation for Basic Research (project №16-07-00034).

References
[1] A T Azar (ed) 2010 Fuzzy Systems (InTech)
[2] Fazzolari M, Alcala R, Nojima Y, Ishibuchi H and Herrera F 2013 IEEE Trans. Fuzzy Syst. 21 45-65
[3] Herrera F 2005 International Journal of Computational Intelligence Research 1 59-67
[4] Juang C-F and Chang P-H 2010 IEEE T Fuzzy Syst 18 138-149
[5] Khodashinskii I A and Dudin P A 2012 Optoelectronics, Instrumentation and Data Processing 48 54-61
[6] Maldonado Y and Castillo O 2013 Appl. Soft Comput. 13 496-508
[7] Juang C-F 2010 Combination of Particle Swarm and Ant Colony Optimization Algorithms for Fuzzy Systems Design Fuzzy Systems, ed A T Azar (InTech) p 195
[8] Angeline P J 1998 Proc. 7th Int. Conf. on Evolutionary Programming VII (London: Springer) p 601
[9] Poli R, Kennedy J and Blackwell T 2007 Swarm Intell. 1 33-57
[10] Liu B, Wang L, Jin Y H, Tang F and Huang D X 2005 Chaos, Solitons Fractals 25 1261–1271
[11] Zhao Q and Yan S 2005 Advances in Natural Computation, ed L Wang, K Chen and Y S Ong (Berlin: Springer) p 632
[12] Yue C, Yan-duo Z, Jing L. and Hui T 2009 Progress in Robotics, ed Kim J-H et al. (Berlin: Springer) p. 372
[13] Chuang L-Y, Yang C-H and Li J-C 2011 Appl. Soft Comput. 11 239-248
[14] Xu H, Lu S and Zhou S 2013 Proc. 2012 Int. Conf. on Information Technology and Software Engineering (Berlin: Springer) p 619
[15] Kennedy J and Ebenhart R 1995 Proc. 1995 IEEE Inter. Conf. on Neural Networks 4 1942
[16] Cui Z, Cai X and Zeng J 2009 International Journal of Innovative Computing, Information and Control 5 951-960
[17] Alatas B, Akin E and Bedri Ozer A 2009 Chaos, Solitons Fractals 40 1715–1734
[18] Alcalá J, Fernández A, Luengo J, Derrac J, García S, Sánchez L and Herrera F 2011 Journal of Multiple-Valued Logic and Soft Computing 17 255-287
[19] Casillas J, Martinez P and Benitez A D 2009 Soft Computing 13 451-465