A Fuzzy Time Series-Based Model Using Particle Swarm Optimization and Weighted Rules

Ortiz Arroyo, Daniel

Published in:
arXiv.org (e-prints)

Publication date:
2023

A Fuzzy Time Series Based Model Using Particle Swarm Optimization and Weighted Rules

Daniel Ortiz-Arroyo
Department of Energy *
doa@et.aau.dk

October 28, 2023

Abstract

During the last decades, a myriad of fuzzy time series models have been proposed in scientific literature. Among the most accurate models found in fuzzy time series, the high-order ones are the most accurate. The research described in this paper tackles three potential limitations associated with the application of high-order fuzzy time series models. To begin with, the adequacy of forecast rules lacks consistency. Secondly, as the model's order increases, data utilization diminishes. Thirdly, the uniformity of forecast rules proves to be highly contingent on the chosen interval partitions. To address these likely drawbacks, we introduce a novel model based on fuzzy time series that amalgamates the principles of particle swarm optimization (PSO) and weighted summation. Our results show that our approach models accurately the time series in comparison with previous methods.

1 Introduction

Fuzzy time series modeling is a technique that models time series using fuzzy logic [20, 37]. During the last two decades a large number of fuzzy time series models have been proposed to model a variety of forecast problems, such as university enrollments [34, 29, 31, 1, 2, 33, 27, 28, 24, 4, 3, 22, 35, 15, 23, 7, 11, 8, 21, 32], stock market indexes [6, 10, 16, 12, 14, 13, 36, 8], temperature prediction [25, 5, 11], car road accidents [17], IT-project costs [7] and annual rice production [27, 28].

Originally, the concept of fuzzy time series was proposed by Song and Chissom [29, 31, 30] in a paper series to serve as a framework for forecast modeling. They proposed the time-variant and the time-invariant model to deal with the problem of forecasting student enrollments at the University of Alabama. Subsequently, Chen [1] introduced a simplified procedure that reduced the high computational overhead of its predecessors. In a subsequent investigation, Chen [2] expanded upon his prior research outlined in [1], wherein he introduced a high-order fuzzy time series model. This extended model exhibited notably superior performance compared to its first-order counterpart. More recently, Chen and Chung [3] further built upon Chen's work detailed in [1] and [2]. In their latest work, they present a forecasting method that leverages high-order fuzzy time series and employs a genetic algorithm paradigm to collectively adjust interval lengths, aiming to enhance forecast accuracy. A somewhat analogous approach was proposed by Kuo et al [21], where particle swarm optimization (PSO) is used in a similar manner. Up to this point, these two models have yielded the most promising results when applied to the enrollment dataset at the University of Alabama. Nevertheless, there remain three potential shortcomings associated with high-order models that require attention. Firstly, the forecast accuracy remains unsatisfactory. Secondly, as the order increases, data utilization diminishes. Thirdly, forecast accuracy proves to be highly sensitive to the chosen interval partitions. To address these issues, we introduce a novel fuzzy time series-based model that combines particle swarm optimization (PSO) with weighted fuzzy rules. The fuzzification method was originally proposed in [26], in this paper, we provide a more detailed description of the model with examples. It must be noted that in this paper we do not consider the forecasting problem for time series but the creation of an accurate fuzzy time series model capable of approximating the time series with precision.

*Author acknowledges the contribution of Jens Runi Poulsen in this research

The remaining part of the paper is organized as follows. Section 2 reviews the concept of PSO. Section 3 provides an overall description of our proposed algorithm. Section 4 demonstrates how the proposed algorithm is used to model a short time series. Section 5 presents experimental results. Finally, in section 6, we provide our conclusion.

2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) [18, 19] is an optimization method that draws inspiration from the collective behaviors observed in bird flocks and fish schools. In PSO, bird flocks are simulated as particle swarms navigating through a virtual search space in search of the optimal solution. Each particle is assigned a fitness value, which is assessed against a fitness function that needs to be optimized. The motion of each particle is influenced by a velocity parameter. In each iteration, the particles move randomly within a confined area, but their individual movements are guided toward the particle that is closest to the optimal solution. Each particle retains knowledge of its own best position (the best solution it has discovered) as well as the global best position (the best solution found by any particle within the group). The parameters are continually updated each time a new best position is identified, leading to an evolving solution as each particle adjusts its position.

PSO starts with the initialization of a collection of randomly generated particles, each representing a candidate solution. Subsequently, an iterative process is used to enhance the existing set of solutions. In the course of each iteration, every particle generates new solutions, which are individually assessed against (1) the particle's own best solution achieved in prior iterations and (2) the best solution currently identified by any particle within the entire swarm. Each candidate solution is represented as a position. When a particle discovers a position superior to its current best-known position, its personal best position is then updated. Moreover, if the new personal best position surpasses the current global best position, the global best position is also adjusted. Once the evaluation phase concludes, each particle updates both its velocity and position using the following equations:

$$v_i = \omega v_i + c_1 r_1(\hat{x}_i - x_j) + c_1 r_2(\hat{g} - x_j) \tag{1}$$

and

$$x_j = x_j + v_i \tag{2}$$

where

- v_i is the velocity of particle p_i and is limited to $[-V_{max}, V_{max}]$ where V_{max} is a user-defined constant.
- ω is an inertial weight coefficient.
- $\hat{x}_i$ is the current personal best position.
- x_j is the current position.
- $\hat{g}$ is the global best position.
- c_1 and c_2 are user-defined constants saying how much the particle is directed towards good positions. They affect how much the particle's local best and global best influence its movement. Generally c_1 and c_2 are set to 2.
- r_1 and r_2 are randomly generated numbers between 0 and 1.

Note that the velocity controls the motion of each particle. The speed of convergence can be adjusted by the inertial weight coefficient and the constants c_1, c_2. Whenever computed velocity exceeds its user-defined boundaries, the computed results will be replaced by either $-V_{min}$ or V_{max}.

3 Algorithm Overview

In this study, we have developed a training algorithm, which will be discussed in detail in the subsequent sections. The training phase will utilize the short enrollment dataset from the University of Alabama [29] as input. The output of this phase consists of forecast enrollments, which will be employed for evaluating the performance in comparison to related studies, as outlined in Section 5. The comprehensive structure of the algorithm is depicted in Figure 1.

The algorithm can divided into two main components, fuzzification and defuzzification. These are highlighted by the dashed areas. Both of these components are decoupled which implies that they can function independently of each other and thus can be used in combination with other alternatives. The fuzzification component is further decomposed

Algorithm 1 The running procedure of the PSO Algorithm

```
for all particles do
    initialize velocities and positions
end for
while stopping criteria is unsatisfied do
    for each particle do
        compute velocities by equation 1
        increment positions by equation 2
        if current fitness value is better than current local best value then
            update local best positions
        end if
        if current fitness value is better than current global best value then
            update global best positions
        end if
    end for
end while
```

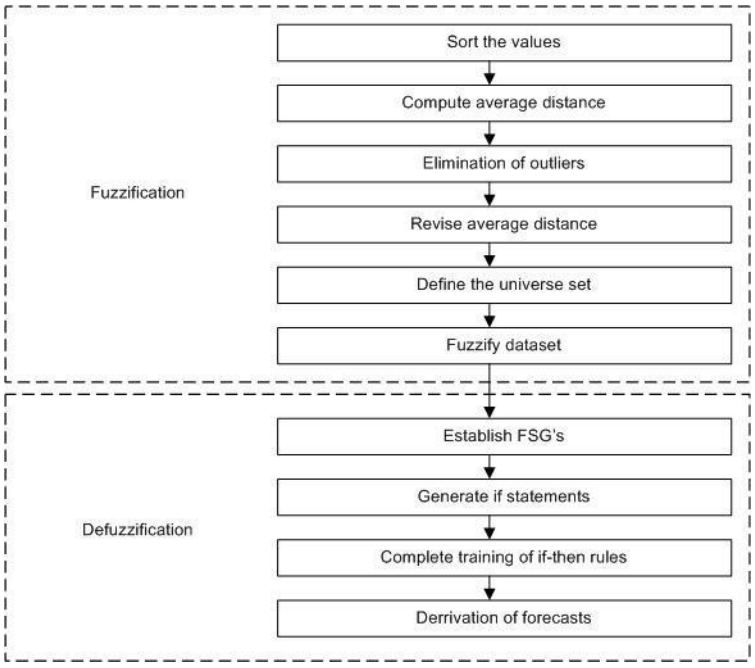

Figure 1: Overall algorithm structure

into a six-step process where the first four steps are data preprocessing functions. The fuzzification task itself comprises the last two steps. Ultimately the goal of this phase is to generate a series of fuzzy sets or interval partitions and to establish associations between the fuzzy sets and the dataset values. After the fuzzification process is completed, data is further processed by the defuzzification component. During this phase, the fuzzy sets generated in the previous phase, are grouped into patterns and transformed into forecast rules. Finally, forecasts are computed by matching the if-then rules with equivalent patterns in the enrollment dataset.

4 Forecasting enrollments with the proposed method

4.1 Fuzzifying the Data

The fuzzification algorithm described here is a further modification of the trapezoid fuzzification approach proposed by Cheng et al in [7]. It differs in the way that it doesn't require the number of sets to be submitted in advance. Instead, the algorithm determines the number of sets based on the variations in data. The advantage of this approach is that the fuzzification process can be carried out automatically. The proposed procedure can be described as a six-step process:

1. Sort the values in ascending order.
2. Compute the average distance between any two consecutive values and the standard deviation.
3. Eliminate outliers.
4. Compute the revised average distance between any two remaining consecutive values.
5. Define the universe of discourse.
6. Fuzzify the dataset.

First, the values in the historical dataset are sorted in ascending order. Then the average distance between any two consecutive values in the sorted dataset is computed and the corresponding standard deviation. The average distance is given by the equation:

$$AD(x_i, \ldots, x_n) = \frac{1}{n-1} \sum_{i=1}^{n-1} |x_{p(i)} - x_{p(i+1)}|, \tag{3}$$

where p is a permutation that orders the values ascendantly: $x_{p(i)} \leq x_{p(i+1)}$. The standard deviation is computed as

$$\sigma_{AD} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - AD)^2}. \tag{4}$$

Both the average distance and standard deviation are used in step 3 to define outliers in the sorted dataset. Outliers are values that are either abnormally high or abnormally low. These are eliminated from the sorted dataset in order to reduce the impact of distorting elements on the average distance value. An outlier, in this context, is defined as a value less than or larger than one standard deviation from the average. After the elimination process is completed, a revised average distance value is computed for the remaining values in the sorted dataset, as in step 2. The revised average distance, obtained in step 4, is used in steps 5 and 6 to partition the universe of discourse into a series of trapezoidal fuzzy sets. In step 5, the universe of discourse is determined. Its lower and upper bound is determined by locating the largest and lowest values in the dataset and augmenting these by (1) subtracting the revised average distance from the lowest value and (2) adding the revised average distance to the highest value. More formally, if D_{max} and D_{min} are the highest and lowest values in the dataset, respectively, and AD_R is the revised average distance, the universe of discourse, U, can be defined as $U = [D_{min} - AD_R, D_{max} + AD_R]$.

When U has been determined, fuzzy subsets can be defined on U. Since subsets are represented by trapezoidal functions, the membership degree, for a given function, μ_A, and a given value, x, is obtained by the equation:

$$\mu_A = \begin{cases} \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \leq a_2 \\ 1, & a_2 \leq x \leq a_3 \\ \dfrac{a_4 - x}{a_4 - a_3}, & a_3 \leq x \leq a_4 \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Prior to the fuzzification of data, we need to know the number of subsets to be defined on U. The number of sets, n, is computed by

$$n = \frac{R - S}{2S}, \tag{6}$$

where R denotes the range of the universe set and S denotes the segment length. Equation 6 originates from the fact that we know the following about S:

$$S = \frac{R}{2n + 1}. \tag{7}$$

The range, R, is computed by

$$R = UB - LB, \tag{8}$$

where UB and LB respectively denote the upper bound and lower bound of U. The segment length, S, equals the average revised distance, AD_R, which in turn constitutes the length of left spread (ls), core (c), and right spread (rs) of the membership function (see fig. 2). That is $ls = AD_R, c = AD_R$ and $rs = AD_R$.

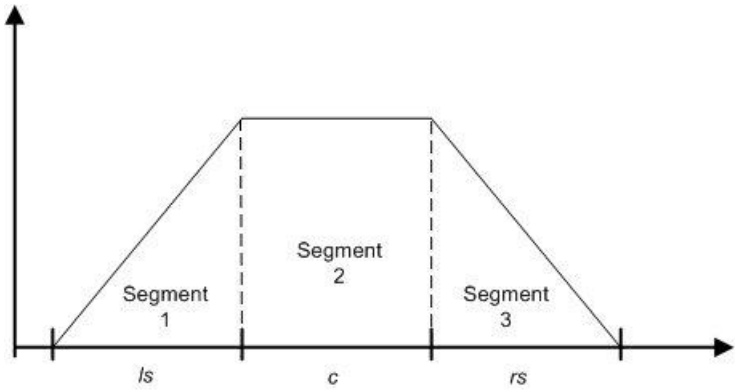

Figure 2: The segments of a trapezoidal fuzzy number

In short, the task here is to decide how many fuzzy sets to generate when the length of each segment, S, and the range, R, are known. When the number of sets has been computed, the sets can be defined on U and data can be fuzzified which completes the final step of the algorithm.

4.1.1 An Example

In the following example, we will fuzzify the first four years (1971 - 1974) of student enrollment at the University of Alabama. The values to be fuzzified are 13055, 13563, 13867, and 14696. Since the sequence of values already appears in ascending order, the sorting part is omitted. The average distance and the standard deviation are respectively computed as

$$AD = \frac{|13055 - 13563| + |13563 - 13867| + |13867 - 14696|}{3} = 547$$

and

$$\sigma_{AD} = \sqrt{\frac{(508 - 547)^2 + (304 - 547)^2 + (829 - 547)^2}{3}} = 216.1 \approx 216.$$

Next, possible outliers are eliminated. Recall that outliers include the values less than or larger than one standard deviation from AD. This means only the values satisfying the condition:

$$547 - 216 \leq x \leq 547 + 216,$$

are taken into consideration when computing the revised average distance. In this case, only one of the three values satisfies the above condition, namely 508. Thus the revised average distance, AD_R, and the segment length, S, equals 508. At this point, steps 1 - 4 are completed. Prior to defining the universe set, U, we need to determine the lower bound and the upper bound (UB) of U. Following equation 8, LB and UP are computed as

$$LB = 13055 - 508 = 12547$$
$$UB = 14696 + 508 = 15204.$$

Hence $U = [12547, 15204]$. The range, R, is computed as the difference between UB and LB. Hence we get $15204 - 12547 = 2657$. Finally, the number of sets, n, is computed as

$$n = \frac{2657 - 508}{2 \cdot 508} = 2.12 \approx 2.$$

Fuzzy set	Trapezoidal fuzzy number (a,b,c,d)	Crisp interval
A_1	(12547,13055,13602,14149)	$\mu_1 = [13055, 13602]$
A_2	(13602,14149,14696,15402)	$\mu_1 = [14149, 14696]$

Table 1: Fuzzifying the first four years of enrollment.

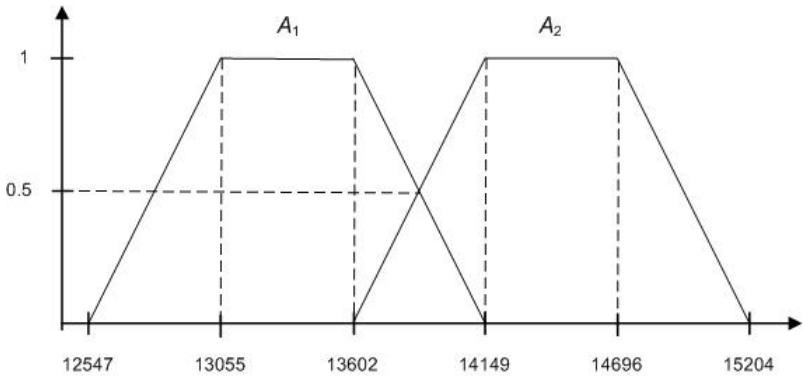

Figure 3: Generated membership functions.

Knowing the universe of discourse and the parameters of n, R and S, the fuzzy sets are generated as shown in figure 3 and table 1.

Note the difference between the points a, b, c and d, in the fuzzy number A_1 and A_2, in table 1 and figure 3, is not exactly 508. This is because the implemented algorithm adapts the segment length, such that the lowest value in the dataset always appears as the left bound of the crisp interval, and the highest value in the dataset always appears as the right bound of the crisp interval. From table 1 and figure 3 it can be seen that the lowest of the four values (i.e. 13055), appears as the lower bound of the first crisp interval, μ_1, and the highest value (i.e. 14696), appears as the upper bound in the second crisp interval, u_2. Normally these values cannot be matched precisely without adjusting the segment length, due to rounding errors occurring as a result of equation 3 and 6.

We are now able to fuzzify the first four historical enrollments according to membership functions A_1 and A_2, defined by:

$$A_1 = \begin{cases} 0, & x < 12547 \\ \dfrac{x - 12547}{13055 - 12547}, & 12547 \leq x \leq 13055 \\ 1, & 13055 \leq x \leq 13602 \\ \dfrac{14149 - x}{14149 - 13602}, & 13602 \leq x \leq 14149 \\ 0, & x > 14149. \end{cases}$$

and

$$A_2 = \begin{cases} 0, & x < 13602 \\ \dfrac{x - 13602}{14149 - 13602}, & 13602 \leq x \leq 14149 \\ 1, & 14149 \leq x \leq 14696 \\ \dfrac{15204 - x}{15204 - 14696}, & 14696 \leq x \leq 15204 \\ 0, & x > 15204. \end{cases}$$

As figure 4 indicates, the boundaries of A_1 and A_2 overlap such that more than one interval may be met. For example, the enrollment for the year 1973 is 13867. This value meets both membership functions. The membership degree in A_1 is $0.5155 \approx 0.52$, and in A_2, it is $0.4845 \approx 0.48$. Hence the enrollment for 1973 is fuzzified as A_1. A special case occurs if the membership degree is 0.5, as this implies that a value has the same membership status in two different sets. In such cases, the respective value is associated with both A_1 and A_2. The results of fuzzifying the first four years of enrollment are shown in table 2.

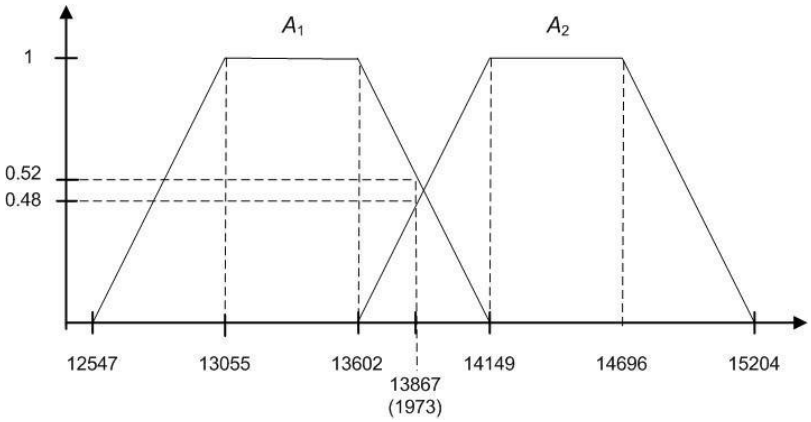

Figure 4: Generated membership functions.

Year	Enrollment	Fuzzy set	Membership degree
1971	13055	A_1	1
1972	13563	A_1	1
1973	13867	A_1	0.52
1974	14696	A_2	1

Table 2: Fuzzified Enrollments 1971 - 1974.

4.1.2 Fuzzifying the Enrollment Dataset

Following the same procedure as in the example from the previous section, the resultant trapezoidal sets, derived by processing the entire enrollment data from 1971 to 1992, are as shown in table 3. The fuzzified annual enrollments are listed in table 4.

Generally it is assumed that the fuzzy sets, $A_1, A_2, \ldots, A_n$, individually represent some linguistic variable. However, with 17 intervals, linguistic values may not make much sense. In the procedure proposed here, this issue is ignored, since the utility of linguistic values has yet to be demonstrated in a fuzzy time series context. However it does not mean that they are not useful in other application contexts.

Fuzzy Set	**Fuzzy Number**
A_1	(12861,13055,13245,13436)
A_2	(13245,13436,13626,13816)
A_3	(13626,13816,14007,14197)
A_4	(14007,14197,14388,14578)
A_5	(14388,14578,14768,14959)
A_6	(14768,14959,15149,15339)
A_7	(15149,15339,15530,15720)
A_8	(15530,15720,15910,16101)
A_9	(15910,16101,16291,16482)
A_{10}	(16291,16482,16672,16862)
A_{11}	(16672,16862,17053,17243)
A_{12}	(17053,17243,17433,17624)
A_{13}	(17433, 17624,17814,18004)
A_{14}	(17814, 18004,18195,18385)
A_{15}	(18195,18385,18576,18766)
A_{16}	(18576,18766,18956,19147)
A_{17}	(18956,19147,19337,19531)

Table 3: Generated fuzzy sets.

Year	Enrollment	Fuzzy Set
1971	13055	A_1
1972	13563	A_2
1973	13867	A_3
1974	14696	A_5
1975	15460	A_7
1976	15311	A_7
1977	15603	A_7
1978	15861	A_8
1979	16807	A_{11}
1980	16919	A_{11}
1981	16388	A_{10}
1982	15433	A_7
1983	15497	A_7
1984	15145	A_6
1985	15163	A_6
1986	15984	A_8
1987	16859	A_{11}
1988	18150	A_{14}
1989	18970	A_{16}
1990	19328	A_{17}
1991	19337	A_{17}
1992	18876	A_{16}

Table 4: Fuzzified annual enrollments.

4.2 Defuzzifying Output

In the following section, we will present a novel approach to defuzzify forecast output. The training phase comprises the following steps:

1. Establish fuzzy set groups (FSG's).
2. Convert the FSG's into corresponding if rules.
3. Complete training of the if rules.
4. Derive forecasts.

Before we go into detail with the individual steps, it is important to understand how defuzzified output is computed. First, recall that the definition of an n-order fuzzy relationship [2] is denoted as

$$F(t-n), \ldots, F(t-2), F(t-1) \rightarrow F(t),$$

where F represents a fuzzified forecast value at time t. Normally it is assumed that the left-hand side of the fuzzy relation is fuzzified in the same manner as the right-hand side. For example, if F, on the left-hand side represents a trapezoidal set, then F, on the right-hand, side represents a trapezoidal set as well. In the modified version introduced here, this notion has been revised such that $F(t)$ is given by the following defuzzification operator, $Y(t)$, defined by

$$Y(t) = \sum_{i=1}^{n} a_{t-i} \cdot w_i, \tag{9}$$

where $w_i \in [0, 1]$ and a_{t-i} denotes the actual value at time $t-i$. Otherwise stated, the defuzzified output is the weighted sum of the actual values from time $t-n$ to $t-1$, where n depends on the time series span. For example, if n = 2, we have

$$Y(t) = (a_{t-1} \cdot w_1) + (a_{t-2} \cdot w_2). \tag{10}$$

One question that needs to be addressed is how the defuzzification operator deployed here should be interpreted from a fuzzy logical point of view. The thought here is simply to consider the weights as a fuzzy relationship between past

values (inputs) and the future value (output). Each w_i represents the strength of the causal relationship between a given input and some unknown output. The closer w_i is to 1, the stronger the relationship and vice versa.

It has to be stressed that the defuzzification operator introduced here is not an aggregation operator [9] from a traditional point of view, since it does not satisfy the boundary condition which is one of the basic conditions of fuzzy aggregation operators. The proposed operator has been specifically adapted to solve the problem at hand because no other operators have been found useful in this context. Averaging operators [9], for example, never produce outputs less than the minimum value of arguments or larger than the maximum value of arguments. In the current situation, this requirement is undesirable due to the fact that future demand patterns often fluctuate beyond the boundaries of previous min and max values. To illustrate this, we need to take a closer look at the enrollment data in table 4. For $t = 1973$ and $n = 2$, we get, $a_{1972} = 13563$ and $a_{1971} = 13055$. Assuming $Y(t)$ is an averaging operator, the output is restricted to the interval [13055,13563]. However actual output for $t = 1973$ is 13867 which is out of reach by any averaging operator. Consider another case for $t = 1981$ and $n = 2$. We then get $a_{1980} = 16919$ and $a_{1979} = 16807$. If $Y(t)$ is the min operator, we get min(a_{1980}, a_{1979}) = 16807, and, if $Y(t)$ is the max operator, we get max(a_{1980}, a_{1979}) = 16919. But the actual output for $t = 1981$ is 16388 which also is unreachable by any averaging operator. As a consequence, a basic requirement for the defuzzification operator proposed here is that it covers a broader interval than min and max. A reasonable assumption with regards to the bounds of arguments, a_{t-i}, is that they are within the limits of the universe set.

4.3 Establishment of Fuzzy Set Groups

In conventional fuzzy time series, fuzzy relationships are identified immediately after data have been fuzzified. However, in the model presented here, the right-hand side of the fuzzy relation is not known until the weights have been determined. Instead of identifying relationships and establishing fuzzy logical relationship groups, we establish fuzzy set groups (FSG's). The purpose of the FSG establishment is to partition historical data into unique sets of sub-patterns which subsequently are converted into IF rules. In the first pass of the algorithm, consecutive sets are grouped pairwise. Table 5 shows the fuzzified data in table 4 grouped in this manner. Every FSG appears in chronological order.

Label	FSG	Label	FSG
1	$\{A_1, A_2\}$	12	$\{A_7, A_7\}$
2	$\{A_2, A_3\}$	13	$\{A_7, A_6\}$
3	$\{A_3, A_5\}$	14	$\{A_6, A_6\}$
4	$\{A_5, A_7\}$	15	$\{A_6, A_8\}$
5	$\{A_7, A_7\}$	16	$\{A_8, A_{11}\}$
6	$\{A_7, A_7\}$	17	$\{A_{11}, A_{14}\}$
7	$\{A_7, A_8\}$	18	$\{A_{14}, A_{16}\}$
8	$\{A_8, A_{11}\}$	19	$\{A_{16}, A_{17}\}$
9	$\{A_{11}, A_{11}\}$	20	$\{A_{17}, A_{17}\}$
10	$\{A_{11}, A_{10}\}$	21	$\{A_{17}, A_{16}\}$
11	$\{A_{10}, A_7\}$	Ø	Ø

Table 5: Establishment of FSG's.

To exemplify the principles of grouping, consider the years 1971, 1972, and 1973 which are fuzzified as A_1, A_2 and A_3, respectively. The pairwise grouping of sets is carried out in the following order:

$$\{F(t-2), F(t-1)\} = \{A_{i,t-2}, A_{i,t-1}\}.$$

Following this principle, the following two FSG's are derived:

$$\{F(1971), F(1972)\} = \{A_1, A_2\}$$

and

$$\{F(1972), F(1973)\} = \{A_2, A_3\}.$$

In table 5, these groups are labeled as 1 and 2, respectively. Ultimately the goal of grouping sets in this manner is to obtain a series of FSG's free of ambiguities. An ambiguity occurs if two or more FSG's contain the same combination of fuzzy sets. From table 5, it can be seen that not all FSG's are unique. Note that the FSG's labeled as 5, 6 and 12 are identical, as is the case with 8 and 16. In order to obtain a series of disambiguated FSG's, we extend the ambiguous FSG's by

including the previous set in the respective time series. Formally this means that the combination ,$\{F(t-2), F(t-1)\}$, is extended to include $F(t-3)$, so the respective FSG now equals $\{F(t-3), F(t-2), F(t-1)\}$. Table 6 shows the extensions of the ambiguous FSG's identified in table 5.

Label	***F*(*t*-2), *F*(*t*-1)**	***F*(*t*-3)**	***F*(*t*-3), *F*(*t*-2), *F*(*t*-3)**
5	$\{A_7, A_7\}$	A_5	$\{A_5, A_7, A_7\}$
6	$\{A_7, A_7\}$	A_7	$\{A_7, A_7, A_7\}$
8	$\{A_8, A_{11}\}$	A_7	$\{A_7, A_8, A_{11}\}$
12	$\{A_7, A_7\}$	A_{10}	$\{A_{10}, A_7, A_7\}$
16	$\{A_8, A_{11}\}$	A_6	$\{A_6, A_8, A_{11}\}$

Table 6: Extending ambiguous FSG's.

The extension process is continued until a unique combination of sets is obtained for each FSG. From table 6, we see that only a single extension is required to obtain a unique combination of sets in this particular case. An updated overview of the FSG's is shown in table 7.

Label	**FSG**	**Label**	**FSG**
1	$\{A_1, A_2\}$	12	$\{A_{10}, A_7, A_7\}$
2	$\{A_2, A_3\}$	13	$\{A_7, A_6\}$
3	$\{A_3, A_5\}$	14	$\{A_6, A_6\}$
4	$\{A_5, A_7\}$	15	$\{A_6, A_8\}$
5	$\{A_5, A_7, A_7\}$	16	$\{A_6, A_8, A_{11}\}$
6	$\{A_7, A_7, A_7\}$	17	$\{A_{11}, A_{14}\}$
7	$\{A_7, A_8\}$	18	$\{A_{14}, A_{16}\}$
8	$\{A_7, A_8, A_{11}\}$	19	$\{A_{16}, A_{17}\}$
9	$\{A_{11}, A_{11}\}$	20	$\{A_{17}, A_{17}\}$
10	$\{A_{11}, A_{10}\}$	21	$\{A_{17}, A_{16}\}$
11	$\{A_{10}, A_7\}$	Ø	Ø

Table 7: Disambiguated FSG's.

4.4 Converting FSG's into Forecast Rules

Defuzzified output, $Y(t)$, is obtained by matching historical patterns with a corresponding if-then rule which is generated on the basis of the content of the FSG's. The task is fairly simple as the sequence of elements in each FSG is the same as they appear in time. This means that for any FSG of size n, the elements appear in the same sequence as in the corresponding time series: $F(t-n), F(t-n+1), \ldots, F(t-1)$. Each FSG can be therefore easily transformed into if rules of the form:

if$(F(t-1) = A_{i,t-1} \wedge F(t-2) = A_{i,t-2} \wedge \ldots \wedge F(t-n) = A_{i,t-n})$;
then $w_{1,t-1} =?$, $w_{2,t-2} =?$,...,$w_{n,t-n} =?$

For convenience, the sequence of conditions in the if rules appear in reversed order compared to their equivalent FSG's. For example, an FSG of the form:

$$\{A_{i,t-2}, A_{i,t-1}\},$$

is converted into an equivalent if rule of the form:

$$\text{if}(F(t-1) = A_{i,t-1} \wedge F(t-2) = A_{i,t-2}).$$

Whenever a rule is matched, the resultant weights are returned and the forecast value, $Y(t)$, is computed according to equation 9. To illustrate this, suppose we need to find a matching if-then rule when forecasting the enrollment for the year 1973. From table 4, we get $F(1971) = A_1$ and $F(1972) = A_2$ for $t = 1973$. Now, assume the following if-then rule exists in the current rule base:

$$\text{if}(F(t-1) = A_{i,t-1} \wedge F(t-2) = A_{i,t-2}) \rightarrow w_{1,t-1} = 0.6488, w_{2,t-2} = 0.3882.$$

The respective if rule is then matched as

$$\text{if}(F(1973-1) = A_1 \wedge F(1973-2) = A_1) \rightarrow w_{1,1972} = 0.6488, w_{2,1971} = 0.3882.$$

Using equation 9, forecast enrollment for the year 1973 is computed as

$$Y(1973) = (13563 \cdot 0.6488) + (13055 \cdot 0.3882) = 13867.62 \approx 13868.$$

By processing all of the FSG's in table 7, a series of incomplete if statements are generated as shown in table 8.

Rule	**Matching Part**
1	if$(F(t-1) = A_2) \wedge F(t-2) = A_1)$
2	if$(F(t-1) = A_3 \wedge F(t-2) = A_2)$
3	if$(F(t-1) = A_5 \wedge F(t-2) = A_3)$
4	if$(F(t-1) = A_7 \wedge F(t-2) = A_5)$
5	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_5)$
6	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_7)$
7	if$(F(t-1) = A_8 \wedge F(t-2) = A_7)$
8	if$(F(t-1) = A_{11} \wedge F(t-2) = A_8 \wedge F(t-3) = A_7)$
9	if$(F(t-1) = A_{11} \wedge F(t-2) = A_{11})$
10	if$(F(t-1) = A_{10} \wedge F(t-2) = A_{11})$
11	if$(F(t-1) = A_7 \wedge F(t-2) = A_{10})$
12	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_{10})$
13	if$(F(t-1) = A_6 \wedge F(t-2) = A_7)$
14	if$(F(t-1) = A_6 \wedge F(t-2) = A_6)$
15	if$(F(t-1) = A_8 \wedge F(t-2) = A_6)$
16	if$(F(t-1) = A_{11} \wedge F(t-2) = A_8 \wedge F(t-3) = A_6)$
17	if$(F(t-1) = A_{14} \wedge F(t-2) = A_{11})$
18	if$(F(t-1) = A_{16} \wedge F(t-2) = A_{14})$
19	if$(F(t-1) = A_{17} \wedge F(t-2) = A_{16})$
20	if$(F(t-1) = A_{17} \wedge F(t-2) = A_{17})$
21	if$(F(t-1) = A_{16} \wedge F(t-2) = A_{17})$

Table 8: Generated if rules in chronological order.

4.5 Training the If-Then Rules With PSO

In the following, we are going to provide an example of how PSO is utilized to tune the weights in the defuzzification operator in equation 9. The user-defined parameters are set as follows:

- The inertial coefficient, ω, equals 1.4.
- The self confidence and social confidence coefficient, c_1 and c_2 both equals 2, respectively.
- The minimum and maximum velocity is limited to [-0.01,0.01].
- The minimum and maximum position is limited to [0,1].
- The number of particles equals 5.

The fitness function employed here is the squared error (SE), defined by

$$SE = (forecast\_value_t - actual\_value_t)^2 \tag{11}$$

Basically, the idea is to evaluate the aggregated result, $Y(t)$, against the actual outcome at time t, and adjust the weights in the defuzzification operator such that SE is minimized. By minimizing SE for each t, MSE is minimized as well. In the following example, the stopping criteria is defined by setting the minimum SE to 3 and the maximum number of iterations to 500.

During the first step of the algorithm, the weights (positions) are initialized. We assume the existence of a stronger relationship between actual output and the more recent observations in the time series data. So, if $F(t-1)$ is fuzzified as A_i and $F(t-2)$ as A_j, a stronger relationship is assumed to exist between A_i and $Y(t)$ than between A_j and $Y(t)$. Therefore, relatively higher weights are assigned to the most recent observations when positions are initialized. Applying this approach, w_{t-i} will usually remain larger than w_{t-i+1} at the point of termination. Table 9 and 10 respectively show the initial positions and velocities of all particles for a given rule.

Particle	**Position 1** (w_1)	**Position 2** (w_2)	**SE**
1	0.75	0.5	8,024,473
2	0.75	0.5	8,024,473
3	0.75	0.5	8,024,473
4	0.75	0.5	8,024,473
5	0.75	0.5	8,024,473

Table 9: Initial positions of all particles.

In the current example, rule 1 in table 8 is trained. As can be seen from table 9, the personal best positions are the same for all particles at the initialization phase. Hence the personal best position equals the global best position for all particles.

Particle	v_1	v_2
1	0.0049	0.0011
2	0.0032	0.0065
3	0.0034	0.0081
4	0.0023	0.0009
5	0.0007	0.0048

Table 10: Randomized initial velocities of all particles.

When all particles and velocities have been initialized, the velocities are updated before positions are incremented. Velocities are updated according to equation 1. The computations yield:

$$
\begin{aligned}
v_{1,1} &= (1.4 \cdot 0.0049) + 2 \cdot r_1(0.75 - 0.75) + 2 \cdot r_2(0.75 - 0.75) && = 0.0069 \\
v_{1,2} &= (1.4 \cdot 0.0011) + 2 \cdot r_1(0.5 - 0.5) + 2 \cdot r_2(0.5 - 0.5) && = 0.0015 \\
v_{2,1} &= (1.4 \cdot 0.0032) + 2 \cdot r_1(0.75 - 0.75) + 2 \cdot r_1(0.75 - 0.75) && = 0.0045 \\
v_{2,2} &= (1.4 \cdot 0.0065) + 2 \cdot r_1(0.5 - 0.5) + 2 \cdot r_2(0.5 - 0.5) && = 0.0091 \\
v_{3,1} &= (1.4 \cdot 0.0034) + 2 \cdot r_1(0.75 - 0.75) + 2 \cdot r_2(0.75 - 0.75) && = 0.0048 \\
v_{3,2} &= (1.4 \cdot 0.0081) + 2 \cdot r_1(0.5 - 0.5) + 2 \cdot r_2(0.5 - 0.5) && = 0.0113 \\
v_{4,1} &= (1.4 \cdot 0.0023) + 2 \cdot r_1(0.75 - 0.75) + 2 \cdot r_2(0.75 - 0.75) && = 0.0032 \\
v_{4,2} &= (1.4 \cdot 0.0009) + 2 \cdot r_1(0.5 - 0.5) + 2 \cdot r_2(0.5 - 0.5) && = 0.0013 \\
v_{5,1} &= (1.4 \cdot 0.0007) + 2 \cdot r_1(0.75 - 0.75) + 2 \cdot r_2(0.75 - 0.75) && = 0.0001 \\
v_{5,2} &= (1.4 \cdot 0.0048) + 2 \cdot r_1(0.5 - 0.5) + 2 \cdot r_2(0.5 - 0.5) && = 0.0067.
\end{aligned}
$$

Positions are incremented according to equation 2. Incremented positions after the first iteration are shown in table 11.

Particle	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	**SE**
1	0.7549	0.5011	8,488,885
2	0.7532	0.5065	8,767,574
3	0.7534	0.5081	8,907,895
4	0.7523	0.5009	8,269,618
5	0.7507	0.5048	8,438,491

Table 11: The positions of all particles after the first iteration.

After the first iteration, none of the computed SE values in table 11 are less than 8,024,473. Thus no personal best positions nor global best positions are reached at this point. At some point, the stopping criteria are met and the algorithm terminates. The personal best positions of all particles after termination are listed in table 12.

Particle	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	**SE**
1	0.6738	0.3699	10,159
2	0.6854	0.3502	1
3	0.6686	0.3662	325
4	0.6724	0.3482	40,597
5	0.6879	0.3383	14,522

Table 12: Particle positions after termination.

According to table 12, particle 2 has the global best position. Hence the weights associated with the rule in question equals 0.6854 and 0.3502. The results of training the rules in table 8 are shown in table 13.

Algorithm 2 PSO algorithm for training of the if-then rules.

for all rules **do**
  find the matching pattern in the fuzzified dataset
  retrieves actual values within the range of pattern and initializes variables
  while stopping criteria is unsatisfied **do**
    for each particle p_i, from $i = 1$ to n **do**
      $v_{ij} = \omega \cdot v_{ij} + c_1 \cdot r_1(local\_b_{ij} - w_{ij}) + c_2 \cdot r_2(global\_b_j - w_{ij})$, from $j = 1$ to n
      if $V_{min} > v_{ij}$ **then**
        $v_{ij} = V_{min}$
      end if
      if $V_{max} < v_{ij}$ **then**
        $v_{ij} = V_{max}$
      end if
      update position as $w_{ij} = w_{ij} + v_{ij}$, from $i = 1$ to n
      compute defuzzified output as $Y(t)_i = \sum_{i=1}^{n} a_{t-j} \cdot w_{ij}$
      compute squared error $SE_i = (Y(t)_i - actual\_value_t)^2$
      if $SE_i < SE_{local\_best}$ **then**
        $SE_{local\_best} = SE_i$
        $local\_b_{ij} = w_{ij}$, from $i = 1$ to n
      end if
      if $SE_i < SE_{global\_best}$ **then**
        $SE_{global\_best} = SE_i$
        $global\_best_{ij} = w_{ij}$, from $i = 1$ to n
      end if
    end for
  end while
  update weights as $\{w_{t-1} = global\_best_1 \wedge \ldots \wedge w_{t-n} = global\_best_n\}$
end for

Rule	Matching Part	Weights
1	if$(F(t-1) = A_2) \wedge F(t-2) = A_1)$	then w_1=0.6488 and w_2=0.3882
2	if$(F(t-1) = A_3 \wedge F(t-2) = A_2)$	then w_1=0.6586 and w_2=0.4102
3	if$(F(t-1) = A_5 \wedge F(t-2) = A_3)$	then w_1=0.667 and w_2=0.408
4	if$(F(t-1) = A_7 \wedge F(t-2) = A_5)$	then w_1=0.6395 and w_2=0.369
5	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_5)$	then w_1=0.4411, w_2=0.3158 and w_3 = 0.2699
6	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_7)$	then w_1=0.4638, w_2=0.4645 and w_3=0.0978
7	if$(F(t-1) = A_8 \wedge F(t-2) = A_7)$	then w_1=0.6695 and w_2=0.3967
8	if$(F(t-1) = A_{11} \wedge F(t-2) = A_8 \wedge F(t-3) = A_7)$	then w_1=0.4379,w_2=0.3892 and w_3=0.2171
9	if$(F(t-1) = A_{11} \wedge F(t-2) = A_{11})$	then w_1=0.1604 and w_2=0.8137
10	if$(F(t-1) = A_{10} \wedge F(t-2) = A_{11})$	then w_1=0.5497 and w_2=0.3798
11	if$(F(t-1) = A_7 \wedge F(t-2) = A_{10})$	then w_1=0.5997 and w_2=0.3809
12	if$(F(t-1) = A_7 \wedge F(t-2) = A_7 \wedge F(t-3) = A_{10})$	then w_1=0.4151, w_2=0.3966 and w_3=0.1582
13	if$(F(t-1) = A_6 \wedge F(t-2) = A_7)$	then w_1=0.6194 and w_2=0.3731
14	if$(F(t-1) = A_6 \wedge F(t-2) = A_6)$	then w_1=0.7524 and w_2=0.302
15	if$(F(t-1) = A_8 \wedge F(t-2) = A_6)$	then w_1=0.3869 and w_2=0.704
16	if$(F(t-1) = A_{11} \wedge F(t-2) = A_8 \wedge F(t-3) = A_6)$	then w_1=0.4668, w_2=0.3847 and w_3=0.2725
17	if$(F(t-1) = A_{14} \wedge F(t-2) = A_{11})$	then w_1=0.654 and w_2=0.4212
18	if$(F(t-1) = A_{16} \wedge F(t-2) = A_{14})$	then w_1=0.635 and w_2=0.4012
19	if$(F(t-1) = A_{17} \wedge F(t-2) = A_{16})$	then w_1=0.6202 and w_2=0.3874
20	if$(F(t-1) = A_{17} \wedge F(t-2) = A_{17})$	then w_1=0.5932 and w_2=0.3831
21	if$(F(t-1) = A_{16} \wedge F(t-2) = A_{17})$	then w_1=? and w_2=?

Table 13: Generated if-then rules in chronological order.

5 Experimental Results

To make this work comparable with those of others, the procedure for evaluating model performance is carried out in the same manner as in other publications. That is by forecasting historical enrollments and then evaluating performance on the basis of performance indicators. The mean squared error (MSE) and the mean average percentage error (MAPE) are used as the basis for evaluating model performance:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(F_i - A_i)^2 \tag{12}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|F_i - A_i|}{A_i} \times 100, \tag{13}$$

where A_i and F_i denote the actual output and forecasted ouput at time i, respectively.

A comparison of the proposed model versus other related models is presented in table 14. The model's parameters used in this experiment are the same as in the example in section 4.5. We have selected the best result out of 10 runs. The models referenced in table 14 are all among those with the highest accuracy rates published. As can be seen from table 14, the MSE and the MAPE of the proposed model is 1 and 0.006, respectively. This is significantly lower than for any of the referenced models. Judging by these results, it can therefore be concluded that the proposed model outperforms any fuzzy time series model currently published.

In the introductory section, it was argued that one of the potential drawbacks of the high-order models is that data becomes underutilized as the model's order increases. To explain this in further detail, we need to take a closer look at the outcome produced by the models in [3] and [21]. Note that the first 7 and 8 years of enrollment data are not forecast. This is because the number of fuzzy sets to be matched for each forecast period increases with the order. For example in [21], the number of sets to be matched is 8. By increasing the number of set combinations to be matched, fewer forecast rules are obtained for future use, and thus data becomes underutilized. Moreover, as the number of sets to be matched increases, the statistical likelihood of encountering equivalent pattern combinations in future datasets decreases.

6 Conclusions

In this paper, we have introduced a novel fuzzy time series model that integrates the principles of weighted summation and particle swarm optimization (PSO) to individually fine-tune fuzzy rules. This combination of techniques enables more precise calibration of fuzzy rules to align with the underlying data patterns, reducing their sensitivity to chosen interval partitions. Furthermore, the individualized adjustment of fuzzy rules diminishes the necessity for increasing the model's order, in contrast to high-order fuzzy time series models. As a result, we achieve more effective data utilization in two ways: (1) by increasing the number of fuzzy rules and (2) by reducing the number of pattern combinations required to match future time series data. Additionally, we have introduced a fuzzification method capable of determining the appropriate number of interval partitions based on observed variations in the time series data.

Empirical experiments demonstrate that our proposed model outperforms comparable approaches. The practical utility of this method heavily relies on its capacity to derive fitting fuzzy rules. The weighted fuzzy rules exhibit a remarkable ability to closely emulate the original time series. The application of this approach for forecasting future values within the time series is a potential avenue for future research.

References

[1] Shyi-Ming Chen. Forecasting enrollments based on fuzzy time series. *Fuzzy Sets and Systems 81*, 1996.

[2] Shyi-Ming Chen. Forecasting enrollments based on high order fuzzy time series. *Cybernetics and Systems: An Int. Journal 33*, 2002.

[3] Shyi-Ming Chen and Nien-Yi Chung. Forecasting enrollments using high-order fuzzy time series and genetic algorithms. *International Journal of Intelligent Systems 21*, 2006.

[4] Shyi-Ming Chen and Chia-Ching Hsu. A new method to forecast enrollments using fuzzy time series. *Int. Journal of Applied Science and Engineering 2*, 2004.

[5] Shyi-Ming Chen and Hwang Jeng-Ren. Temperature prediction using fuzzy time series. *Systems, Management and Cybernetics 30*, 2000.

Year	Enroll	Chen (order 3) [1]	Li and Cheng [24]	Sing (order 3) [28]	Stevenson and Porter [32]	Chen and Hsu [4]	Chen and Chung (order 9)	Kuo et al (order 9) [21]	Proposed model
1971	13055	-	-	-	-	-	[3]	-	-
1972	13563	-	13500	-	13410	13750	-	-	-
1973	13867	-	13500	-	13932	13875	-	-	13868
1974	14696	14500	14500	14750	14664	14750	-	-	14696
1975	15460	15500	15500	15750	15423	15375	-	-	15460
1976	15311	15500	15500	15500	15847	15313	-	-	15309
1977	15603	15500	15500	15500	15580	15625	-	-	15602
1978	15861	15500	15500	15500	15877	15813	-	-	15861
1979	16807	16500	16500	16500	16773	16834	16846	-	16806
1980	16919	16500	16500	16500	16897	16834	16846	16890	16919
1981	16388	16500	16500	16500	16341	16416	16420	16395	16390
1982	15433	15500	15500	15500	15671	15375	15462	15434	15434
1983	15497	15500	15500	15500	15507	15375	15462	15505	15497
1984	15145	15500	15500	15250	15200	15125	15153	15153	15143
1985	15163	15500	15500	15500	15218	15125	15153	15153	15163
1986	15984	15500	15500	15500	16035	15938	15977	15971	15982
1987	16859	16500	16500	16500	16903	16834	16846	16890	16859
1988	18150	18500	18500	18500	17953	18250	18133	18124	18150
1989	18970	18500	18500	18500	18879	18875	18910	18971	18971
1990	19328	19500	19500	19500	19303	19250	19334	19337	19328
1991	19337	19500	19500	19500	19432	19250	19334	19337	19336
1992	18876	18500	18500	18750	18966	18875	18910	18882	18875
MSE		86694	85040	76509	21575	5611	1101	234	1
MAPE		1.53	1.53	1.41	0.57	0.36	0.15	0.014	0.006

Table 14: Comparing results of different fuzzy time series models.

[6] Tai-Liang Chen, Ching-Hsue Cheng, and Hia-Jong Teoh. High-order fuzzy time series based on multi period adaption model for forecasting stock markets. *Physica A 387*, 2008.

[7] Ching-Hsue Cheng, Jing-Rong Chang, and Che-An Yeh. Entrophy-based and trapezoid fuzzification-based fuzzy time series approaches for forecasting IT project cost. *Technological Forecasting & Social Change 73*, 2006.

[8] Ching-Hsue Cheng, Guang-Wei Cheng, and Jia-Wen Wang. Multi-attribute fuzzy time series method based on fuzzy clustering. *Expert Systems with Applications 34*, 2008.

[9] Marcin Detyniecki. Fundamentals on aggregation operators, 2001. `http://www-poleia.lip6.fr/~marcin/papers/Detynieck_AGOP_01.pdf`(validated April 2 2011).

[10] Kun-Huang Huarng, Tiffany Hui-Kuang Yu, and Yu W. Hsu. A multivariate heuristic model for fuzzy-time series forecasting. *Systems, Management and Cybernetics 37*, 2007.

[11] Kunhuang Huarng. Effective lengths on intervals to improve forecasting in fuzzy time series. *Fuzzy Sets and Systems 123*, 2001.

[12] Kunhuang Huarng and Hui-Kuang Yu. A type 2 fuzzy time series model for stock index forecasting. *Physica A 353*, 2005.

[13] Kunhuang Huarng and Tiffany Hui-Kuang Yu. The application of neural networks to forecast fuzzy time series. *Physica A 363*, 2006.

[14] Kunhuang Huarng and Tiffany Hui-Kuang Yu. Ratio-based lengths of intervals to improve fuzzy time series forecasting. *Systems Management and Cybernetics 36*, 2006.

[15] Jeng-Ren Hwang, Shyi-Ming Chen, and Chia-Hoang Lee. Handling forecast problems using fuzzy time series. *Fuzzy Sets and Systems 100*, 1998.

[16] Tahseen A. Jilani and Syed M. A. Burney. A refined fuzzy time series model for stock market forecasting. *Statistical Mechanics and its Applications 387*, 2008.

[17] Tahseen A. Jiliani and Syed M. A. Burney. Multivariate high order fuzzy time series forecasting for car road accidents. *Int. Journal of Computational Intelligence 4*, 2008.

[18] James Kennedy and Rusell Eberhart. Particle swarm optimization. *Proc. of IEEE Int. Conference on Neural Network*, 1995.

[19] James F. Kennedy, Rusell C. Eberhart, and Yuhui Shi. *Swarm intelligence*. Morgan Kaufman, 2001.

[20] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Prentice Hall, 1995.

[21] I-Hong Kuo, Shi-Jinn Horng, Tzong-Wann Kao, Tsung-Lieh Lin, Lee Cheng-Ling, and Yi Pan. An improved method for forecasting enrollments based on fuzzy time series and particle swarm optimization. *Expert Systems with Applications 36*, 2009.

[22] Hsun-Shih Lee and Ming-Tao Chou. Fuzzy forecasting based on fuzzy time series. *Int. Journal of Computer Mathematics 81*, 2004.

[23] Sheng-Tun Li and Yen-Peng Chen. Natural partioning-based forecasting model for fuzzy time series. *Fuzzy Systems 3*, 2004.

[24] Sheng-Tun Li and Yi-Chung Cheng. Deterministic fuzzy time series model for forecasting enrollments. *Expert Systems with Applications 34*, 2008.

[25] Sheng-Tun Li, Yi-Chung Cheng, and Lin Su-Yu. A fcm-based deterministic forecasting model for fuzzy time series. *Computers and Mathematics with Applications 56*, 2008.

[26] Daniel Ortiz-Arroyo and Jens Runi Poulsen. A weighted fuzzy time series forecasting model. *Indian journal of science and technology*, 11(27):1–11, 2018.

[27] Shiva Raj Singh. A robust method of forecasting based on fuzzy time series. *Applied Mathematics and Computation 188*, 2007.

[28] Shiva Raj Singh. A simple time variant method for fuzzy time series forecasting. *Cybernetics and Systems: An Int. Journal 38*, 2007.

[29] Qiang Song and Brad Chissom. Forecasting enrollments with fuzzy time series - part i. *Fuzzy Sets and Systems 54*, 1993.

[30] Qiang Song and Brad Chissom. Fuzzy time series and its models. *Fuzzy Sets and Systems 54*, 1993.

[31] Qiang Song and Brad Chissom. Forecasting enrollments with fuzzy time series - part ii. *Fuzzy Sets and Systems 62*, 1994.

[32] Meredith Stevenson and John E. Porter. Fuzzy time series forecasting using percentage change as the universe of discourse. *World Academy of Science, Engineering and Technology 55*, 2009.

[33] Joe Sullivan and William H. Woodall. A comparison of fuzzy forecasting and markov modeling. *Fuzzy Sets and Systems 64*, 1994.

[34] Chao-Chih Tsai and Shun-Jyh Wu. Forecasting enrollments with high-order fuzzy time series. *Fuzzy Information Processing Society*, 2000.

[35] Ruey-Chyn Tsaur, Jia-Chi Yang, and Hsiao-Fan Wang. Fuzzy relation analysis in fuzzy time series model. *Computers and Mathematics with Applications 49*, 2005.

[36] Hui-Kuang Yu. Weighted fuzzy time series models for taiex forecasting. *Physica A 349*, 2005.

[37] Lofti A. Zadeh. Fuzzy sets. *Information and Control 8*, 1965.