Almaraashia, M. and John, Robert and Hopgood, A. and Ahmadi, S. (2016) Learning of interval and general type-2 fuzzy logic systems using simulated annealing: theory and practice. Information Sciences, 360 . pp. 21-42. ISSN 1872-6291

Learning of Interval and General Type-2 Fuzzy Logic Systems using Simulated Annealing: Theory and Practice

M. Almaraashi[a,*], R. John[b], A. Hopgood[c], S. Ahmadi[d]

[a]*The University College in Aljamoum, Umm Al-Qura University, Makkah, Saudi Arabia.*
[b]*Automated Scheduling Optimization and Planning Group (ASAP), University of Nottingham, NG8 1BB, UK.*
[c]*HEC Management School, University of Liege, 4000 Liege, Belgium.*
[d]*Center for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, LE1 9BH, UK.*

Abstract

This paper reports the use of simulated annealing to design more efficient fuzzy logic systems to model problems with associated uncertainties. Simulated annealing is used within this work as a method for learning the best configurations of interval and general type-2 fuzzy logic systems to maximize their modeling ability. The combination of simulated annealing with these models is presented in the modeling of four benchmark problems including real-world problems. The type-2 fuzzy logic system models are compared in their ability to model uncertainties associated with these problems. Issues related to this combination between simulated annealing and fuzzy logic systems, including type-2 fuzzy logic systems, are discussed. The results demonstrate that learning the third dimension in type-2 fuzzy sets with a deterministic defuzzifier can add more capability to modeling than interval type-2 fuzzy logic systems. This finding can be seen as an important advance in type-2 fuzzy logic systems research and should increase the level of interest in the modeling applications of general type-2 fuzzy logic systems, despite their greater computational load.

Keywords: simulated annealing, interval type-2 fuzzy logic systems, general type-2 fuzzy logic systems, learning

1. Introduction

Fuzzy logic systems have been applied successfully to a broad range of problems in different application domains. One such type of application is concerned with using fuzzy logic for system modeling and approximation where a fuzzy inference system is used to model human knowledge or to approximate non-linear and dynamic systems. However, the existence of uncertainties and lack of information in many real-world

*Corresponding author.
Email address: msmaraashi@uqu.edu.sa (M. Almaraashi)

problems makes it difficult to model such problems using expert knowledge only. Examples of such problems include identifying systems with no known rule-base and systems with only historic data observation. It becomes clear that when designing a simple fuzzy logic system with few inputs, the experts may be able to provide efficient rules but, as the complexity of the system grows, suitable rule-base and membership functions become difficult to acquire. Therefore, some automated tuning and learning methods are often used to cope with such situations. The objective of these methods is to get parameterized functions that best model these problems according to chosen criteria. The use of automated methods to design fuzzy logic systems has helped to model many real-world problems that are difficult to understand by experts and it is now a well-established methodology for modeling and approximation applications. The **motivation** for this research is two-fold:

- Type-2 fuzzy logic systems have numerous parameters that need to be determined in the design of any system. The determination of these parameters is an open research question and motivates our approach to learning type-2 fuzzy systems.
- The growth in interest in type-2 fuzzy logic has not fully manifested itself in real-world applications using general type-2 fuzzy sets. The emphasis has been on interval type-2 fuzzy sets, thus not taking advantage of the more general representation. By allowing for the learning and optimization of type-2 fuzzy systems we expect the use of general type-2 fuzzy sets to grow.

So the motivation is clear, and we now elaborate on these points.

Learning and optimization. This research is concerned with the learning of type-2 fuzzy logic systems, both general and interval. Type-2 fuzzy logic systems are now well established as both a research topic and an application tool. The motivation for the use of type-2 fuzzy sets is that type-1 fuzzy logic has problems when faced with environments that contain uncertainties that are typical in a large number of real-world applications. These uncertainties in the environment translate into uncertainties about membership functions [38]. Type-1 fuzzy logic cannot fully handle these uncertainties because it is precise in nature and for many applications it is unable to model knowledge adequately, while type-2 fuzzy logic offers a higher level of imprecision modeling [26]. The extra dimension and parameters in type-2 fuzzy sets are supposed to provide more design freedom and flexibility than type-1 fuzzy sets. The use of automated learning methods becomes important as complexity grows when designing type-2 fuzzy logic systems.

Many approaches have been proposed to learn and tune type-1 and type-2 fuzzy logic systems including search algorithms such as genetic algorithms and particle swarm algorithms, as well as local search algorithms and classical learning methods. Compared to genetic algorithms, few researchers have studied use of simulated annealing to learn type-1 fuzzy logic systems such as [16, 12, 49]. So far as we are aware, the only research reported on the use of simulated annealing to design type-2 fuzzy logic systems is the authors' previous work in [3, 4, 5, 6].

Helping develop real-world applications. Another motivation for this research comes from the lack of applications using general type-2 fuzzy logic systems. Type-2

fuzzy logic is a growing research topic with much evidence of successful applications. However, almost all developments of type-2 fuzzy logic systems have been based on interval type-2 fuzzy logic [45][27]. The heavy computational load associated with the generalized form of type-2 sets is the main driver for the lack of applications of general type-2 fuzzy sets compared with the interval model. This prior work has reinforced the common concept that interval type-2 fuzzy logic systems can add more modeling capabilities than type-1 fuzzy logic systems but with extra computational cost. Learning and optimization of general type-2 fuzzy logic systems are open areas for more research, as well as the ongoing research on how to reduce the complexity of general type-2 fuzzy logic systems, especially in the type-reduction phase of the system. The large number of methods used to design type-1 and interval type-2 fuzzy logic systems can be seen as potential candidates for general type-2 fuzzy logic systems and some of them might uncover further possibilities for modeling uncertainty. However, recent advances in general type-2 fuzzy logic systems research, including new representations, optimized operations and faster type-reduction methods, indicate an expected growth in applications. Despite the larger number of computations associated with general type-2 fuzzy sets, there may well be benefits compared to interval type-2 fuzzy sets. This ability can be unveiled using automated designing methods rather than being chosen by the designer manually. Automated methods can fine-tune initial fuzzy logic system designs due to the lack of a rational basis for choosing secondary membership functions for general type-2 fuzzy sets [36, p.302]. This issue enforces the need for using automated methods in such problem. The other factor affecting the usage of general type-2 fuzzy logic systems is the lack of practical parameterization methods to handle the third dimension in general type-2 fuzzy sets. In general, a general type-2 fuzzy logic system has the potential to model more uncertainties despite the large amount of computations associated with it especially when applied to non real-time applications. In consequence, the question of how much general type-2 fuzzy logic systems can add to modeling performance over interval type-2 fuzzy logic systems is another issue that warrants investigation.

The research reported here introduces a new method for learning general type-2 fuzzy systems with a unique combination of learning the footprint of uncertainty (FOU) followed by learning the secondary membership functions (SMF). In addition, we show that when using the vertical slice type reducer we have improvement over other approaches implemented here. Furthermore, interval type-2 fuzzy logic systems were applied to answer the question of to what extent general type-2 fuzzy sets can add more abilities and flexibilities to modeling than interval type-2 fuzzy sets. A detailed analysis is carried out of the learning of general type-2 fuzzy systems on a set of real-world data with and without added noise and, as such, provides significant insight into how the future of learning general type-2 fuzzy systems can be carried out. These methods are applied to four benchmark problems: noise-free Mackey-Glass time series forecasting [34], noisy Mackey-Glass time series forecasting [34], and two real-world problems, namely the estimation of the low-voltage electrical line length in rural towns and the estimation of the medium-voltage electrical line maintenance cost [11].

The rest of this paper starts with a review of the methods and concepts used in this work in section 2 and issues related to the design of general type-2 fuzzy logic systems in sections 3 and 3.2. The methodology and the results are detailed in sections 4 and 5

and some conclusions are drawn in section 6.

2. Background

2.1. Type-2 fuzzy sets and systems

A type-2 fuzzy set [36, p.83][38], denoted $\tilde{A}$, is characterized by a type-2 membership function $\mu_{\tilde{A}}(x,u)$ where $x \in X$ and $u \in J_x \subseteq [0,1]$. For example :

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{1}$$

where $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$. When all the secondary grades $\mu_{\tilde{A}}(x,u)$ equal 1 then $\tilde{A}$ is an interval type-2 fuzzy set. Interval type-2 fuzzy sets are easier to compute than general type-2 fuzzy sets. The footprint of uncertainty (FOU) is a 2D representation of an interval type-2 set and represents the union of all primary memberships and can be described by a lower and upper membership functions. The ease of computation and representation of interval type-2 fuzzy sets is the main reason for their wide usage in real-world applications. The principal membership function [36, p.86] occurs when there is only one secondary grade equal to 1 at each secondary membership function of type-2 set. Therefore, the principal membership function is the union of all such points at which the unity occurs as follows:

$$\mu_{principal}(x) = \int_{x \in X} u/x \text{ where } f_x(u) = 1 \tag{2}$$

Using the Zadeh extension principle, union and intersection of type-2 fuzzy sets are defined (known as join and meet respectively) [29]. Karnik and Mendel [29] has proposed a method to calculate meet and join operations when all secondary membership functions are normal and convex. Coupland and John [15] has presented an extension to this formula to allow the use of non-normal secondary membership functions.

There are some representations for type-2 fuzzy sets that have been proposed in the literature such as vertical-slice representation [36, p.83][38], wavy-slice representation [38], geometric representation [15], alpha-planes [41], alpha cuts [22] and Z-slices [45]. The most well-known representations among them are the vertical-slice and wavy-slice representations. The vertical-slice representation [36, p.83] represent fuzzy sets by using secondary sets in a vertical-slice manner where :

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid \forall x \in X\} \tag{3}$$

$$\mu_{\tilde{A}}(x) = \mu_{\tilde{A}}(u|x) = \int_{\forall u \in J_x \subseteq [0,1]} f_x(u)/u \tag{4}$$

This representation is very useful for computation. In wavy-slice representation [38], a type-2 fuzzy set is represented as a union of embedded type-2 fuzzy sets where each embedded type-2 fuzzy set $\tilde{A}_e$ has the same domain of type-2 fuzzy set $\tilde{A}$. The type-2 embedded set $\tilde{A}_e$ has been defined for discrete universes of discourse X and U, an embedded type-2 set $\tilde{A}_e$ has N elements, where $\tilde{A}_e$ contains exactly one element

from J_{x_1}, J_{x_2}, ..., J_{x_N}, namely $u_1, u_2, ..., u_N$, each with associated secondary grade, namely $f_{x_1}(u_1)$, $f_{x_2}(u_2)$, ..., $f_{x_N}(u_N)$ [36, p.83][38]. For example:

$$\tilde{A}_e = \sum_{i=1}^{N} [f_{x_i}(u_i)/u_i]/x_i, u_i \in J_{xi} \subseteq U = [0, 1]. \tag{5}$$

In this definition, the embedded set contains N elements represented using the primary memberships $u_i \in J_{x_i}$ that is linked to its secondary membership grades $f_{x_i}(u_i)$ in ordered pairs. So type-2 fuzzy set A can be shown as a union of embedded type-2 fuzzy sets as follows:

$$\tilde{A} = \sum_{j=1}^{n} \tilde{A}_e^j \tag{6}$$

where the total number of type-2 embedded sets in type-2 fuzzy set A is calculated using the number of discretised points in the primary domain N and the primary membership functions M (known as the secondary domain) as follows:

$$n = \prod_{i=1}^{N} M_i \tag{7}$$

where $\tilde{A}_e^j$ denotes the j the type-2 embedded fuzzy set in type-2 fuzzy set $\tilde{A}$. The wavy-slice representation known as the Mendel-John Representation Theorem (RT) has been proposed by [38]. It is useful for theoretical derivations but not useful for practical use because of the astronomical number in the union of embedded sets. However, it is very useful when dealing with interval type-2 fuzzy sets due to the ability of using type-1 fuzzy mathematics which is easy to deal with [39].

Type-2 fuzzy logic systems are rule-based systems that are similar to type-1 fuzzy logic systems in terms of the structure and components but type-2 FLS has an extra output process component which is called the type-reducer before defuzzification. The components of a type-2 Mamdani fuzzy system are fuzzifier, rules, inference engine, type-reducer and defuzzifier. The type-reducer reduces output type-2 fuzzy sets to type-1 fuzzy sets then the defuzzifier reduces it to a crisp output. The type-reduction stage is the most computationally expensive stage in a type-2 fuzzy logic system.

2.2. Type-2 fuzzy sets and uncertainty modelling

Type-1 fuzzy logic has been used successfully in a wide range of problems such as control system design, decision making, classification, system modeling and information retrieval. However, the type-1 approach is not able to directly model all uncertainties and minimize their effects [38]. These uncertainties exist in a large number of real-world applications. They can be a result of uncertainty in inputs, uncertainty in outputs, uncertainty that is related to the linguistic differences, uncertainty caused by the change of conditions in the operation, and uncertainty associated with the noisy data when training the fuzzy logic system [36, p.68]. All these uncertainties translate into uncertainties about the membership functions of the fuzzy sets [38]. Therefore, the existence of uncertainties in the majority of real-world applications makes the use

of type-1 fuzzy logic inappropriate in many cases especially with problems related to inefficiency of performance in fuzzy logic control [21]. Problems related to modeling uncertainty using membership functions of type-1 fuzzy sets have been recognized early and [50] introduced higher types of fuzzy sets called type-n fuzzy sets including type-2 fuzzy sets [37]. Type-2 fuzzy logic systems have many advantages compared with type-1 fuzzy logic systems, including the ability to handle different types of uncertainties and the ability to model problems with fewer rules [21]. Two factors should be considered regarding the the widespread perception that a general type-2 fuzzy logic system should outperform the interval form which also should outperform a type-1 fuzzy logic system [46]. These two factors are the dependence of performance on the choice of the model parameters as well as on the variability of uncertainty within the application [46]. Therefore, a good choice of the model's parameters using automated methods is desirable to get clearer conclusions regarding this comparison. Despite these promising indicators of the general type-2 fuzzy logic systems, almost all developments of type-2 fuzzy logic systems have been based on interval type-2 fuzzy logic systems. However, new representations allow us to consider general type-2 fuzzy logic systems. These representations include geometric T2FLS [15], alpha-planes [41], alpha cuts [22] and Z-slices [45, 10][47]. There have been a number of developments in reducing the computations for general type-2 fuzzy logic systems. For type-reduction, the geometric defuzzifier [15], the sampling defuzzifier [19] followed by importance sampling defuzzifier [31] and a centroid defuzzifier based on the alpha representation [32] have been proposed. One attempt to design general type-2 sets based on zSlices representation was proposed in [10] where survey data and device characteristics were used to build zSlices automatically. Other work using an alpha-planes representation has been applied, e.g. as a method for edge-detection [35] and a learning method to forecast Mackey-Glass time-series [41]. The latter showed a better performance of general type-2 fuzzy logic systems using a simpler model known as "triangle quasi-type-2 fuzzy logic system" first presented in [40]. Some other researchers used some neural network concepts or classification algorithms such as: type 2 Adaptive Network Based Fuzzy Inference System (ANFIS) [28], general type-2 fuzzy neural network (GT2FNN) [24] and fuzzy C-means algorithm with a model known as "efficient triangular type-2 fuzzy logic system" [43]. To the best of the authors' knowledge, no attempt to employ a learning method to general type-2 fuzzy logic systems using the vertical-slices representation has been reported. To achieve this objective, apart from using a practical type-reducer, some kinds of parametrization are needed for general type-2 sets to allow learning or optimization techniques to deal with these parameters easily rather than having all the secondary grades or membership functions chosen manually. The parametrization method should preserve most of the freedom associated with GT2FLS.

Our proposed practical design methodology aims to reduce the computations needed to get the best footprint of uncertainty (FOU). The proposed parametrization method was first presented in [6] and [7]. In addition, this paper presents a novel approach for learning all parameters of general type-2 fuzzy logic systems using simulated annealing under the vertical-slices representation.

2.3. Simulated annealing and type-2 fuzzy logic systems

The simulated annealing algorithm is a simple and general optimization algorithm for finding global minima [30]. It has been used widely to search for optimal or nearly optimal solutions in a wide range of optimization problems. In this work, it acts as a learning algorithm to automatically design fuzzy logic systems by searching for the best configurations of these systems. One of the motivations for using simulated annealing with fuzzy systems is that it does not require the existence of mathematical properties such as differentiability in the problem, which allows the possibility of using all fuzzy structure components including non-differentiable t-norms and non-differentiable membership functions. Although the combination might have more complexity and longer search time than local search algorithms, it is more likely to find the global or near global optima of the configuration of fuzzy logic systems than local search approaches. This is due to the ability of simulated annealing to avoid local optima by accepting higher-cost states with some probability in order to explore the problem space. In addition, simulated annealing can suit high dimensionality problems as it scales well with the increase of variable numbers, which makes it a good candidate for the optimization of fuzzy logic systems [16]. Also, it is able to handle cost functions with different degrees of non-linearities, discontinuities, and stochasticity [23]. The problem of optimizing membership functions of the fuzzy logic system in order to minimize the objective function is a complex problem due to the large number of parameters used as well as the the non-differentiable and non-continuous objective functions [20]. The simulated annealing convergence normally requires an exponential time which causes the algorithm to be impractical in some cases [1, p.14]. One of the criticisms of simulated annealing is the difficulty in fine-tuning its parameters, so it can be time-consuming for developers to find an optimal fit [23]. The formalizations and configurations for simulated annealing to design fuzzy logic systems can be chosen from a large number of choices proposed in the literature.

3. Designing and learning of general type-2 fuzzy logic systems

3.1. A practical choice for general type-2 fuzzy set

In order to get an effective and practical form of general type-2 set, the chosen form should:

- have a low computational burden;
- preserve most of the freedom associated with general type-2 sets.

These two objectives are normally in conflict as more freedom (through parameters) requires more computations. Therefore, some trade-offs are needed using some parameterization mechanisms. One way to do this is to have parameterized secondary membership functions that are asymmetric and convex. For example, consider a triangular secondary membership function with an apex in the area between the lowest and the highest FOU points (FOU_{lower} and FOU_{upper}) and primary memberships for each x in the domain. The asymmetry is preferred to allow optimizing the apex location of the SMF when their primary memberships are fixed. The other preferred property is to

have a convex SMF to allow quick meet and join operations when using these sets in GT2FLS.

The secondary membership function of a general type-2 fuzzy set is itself a type-1 fuzzy set. Our approach is to learn the 'apex' of the secondary membership functions using a location indicator of the apex point called the "apex factor" (AF). The apex factor for each SMF takes the value in $[0, 1]$ where if it is zero it takes a value at the SMF and at unity the UMF. This works for any asymmetric and convex shape of SMF including non-normal. In other words, to allow learning the best location for the apex for each SMF, a function to determine the SMF's apex locations in FOU for each x in the primary domain is needed. The values of the apex locations must be bounded by the highest and the lowest FOU points (FOU_{upper} and FOU_{lower}) for each x in the domain. An example of this approach is to have a piecewise linear function or a smooth piecewise-polynomial function and to use some interpolation methods. However, it could be possible to ensure this condition of boundaries when designing the first model of the general type-2 set but this is very difficult to trace and ensure for each x in the continuous domain when learning $SMF_{apex}(x)$ as the interpolation might define some apexes domains outside the FOU boundaries. Therefore, a new parametric formula is proposed here that normalizes the FOU apex locations to be within $FOU(x)$ for each x in the primary domain. This is done by defining the $SMF_{apex}(x)$ as following [6]:

$$SMF_{apex}(x) = 1/(FOU_{low}(x)+g(x)\times(FOU_{up}(x)-FOU_{low}(x))). \quad 0 \leqslant g(x) \leqslant 1 \tag{8}$$

where $g(x)$ is a parameter called "apex factor" that is used as an apex location indicator for each x. This parameter can be used to change the apex location without the need to check for boundary conditions. For example when $g(x) = 0.5$, the location of the apex is in the middle between $FOU_{upper}(x)$ and $FOU_{lower}(x)$ and the resulting SMF is symmetrical. Therefore, this parameter is acting as a variable representing the apex locations when doing some optimization or learning for the general type-2 set. An example of the use of this parameter is to use a piecewise linear function to determine this parameter for all x in the primary domain. For example, suppose that $k_1, k_2, ..., k_n$ are ordered points in the x domain and $g(k1), g(k_2), ..., g(k_n)$ are their apex factors which both define the piecewise linear function, then:

$$g(x) = \begin{cases} 0.5, & x < k_1 \\ g(k_i) + \frac{x-k_i}{k_{i+1}-x} \times (g(k_{i+1}) - g(k_i)), & k_i \leqslant x \leqslant k_{i+1} \\ 0.5, & x > k_n \end{cases} \tag{9}$$

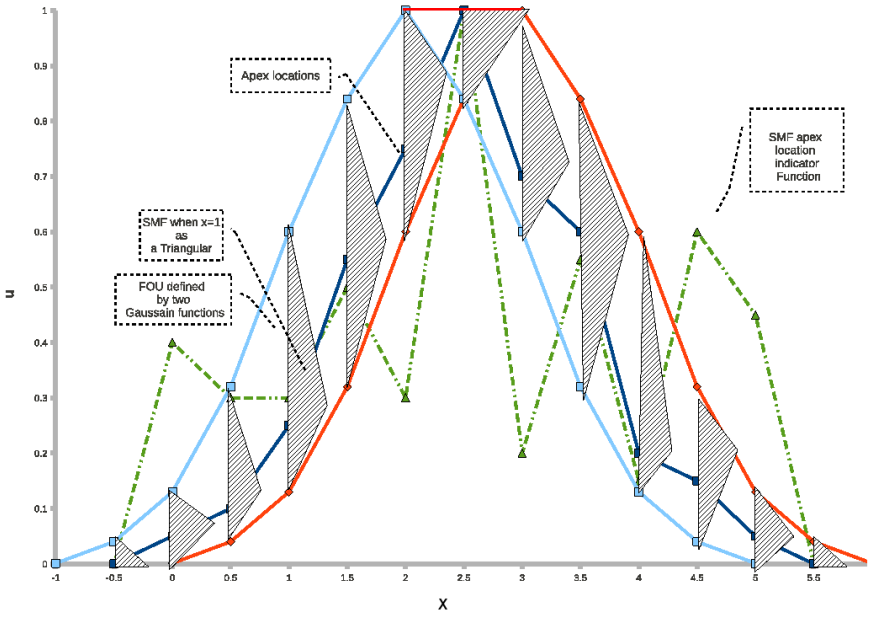
Therefore, each point in the primary domain is linked with one apex factor $g(x)$. Another similar function to determine the height of the apexes when non-normal SMFs are used can be designed the same way. For example:

$$h(x) = \begin{cases} 1, & x < k_1 \\ h(k_i) + \frac{x-k_i}{k_{i+1}-x} \times (h(k_{i+1}) - h(k_i)), & k_i \leqslant x \leqslant k_{i+1} \\ 1, & x > k_n \end{cases} \tag{10}$$

This form is not identical to the principal function described in [36, p.86] or the fuzzy truth numbers proposed in [43] because for each x value, the SMF can be non-normal.

The lowest and highest FOU points (FOU_{lower} and FOU_{upper}) for each x can be defined by another function such as trapezoidal, Gaussian or triangular functions or any other functions used to define interval type-2 sets. An example of the proposed method is shown in figure 1 and an example of learning the secondary membership functions is shown in figure 2. The chosen form is based on the latest general type-2 literature using the vertical-slice representation and the novel method we proposed here to determine the apex locations and heights of SMFs. Although, this is not a new representation and can not be generalized for all forms of general type-2 sets, the aim of this method is to have general type-2 fuzzy sets simplified for practical usage.

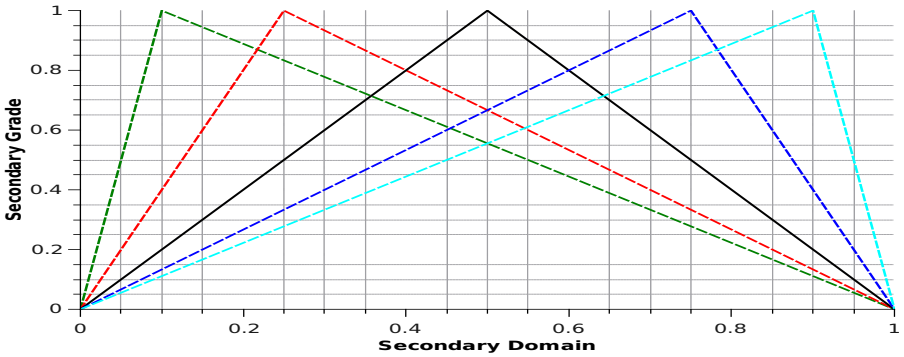
Figure 1: General type-2 fuzzy set defined by its FOU and SMF. The FOU is defined by two Gaussian functions while the SMF is a triangular shaped defined by linear interpolation of two piecewise linear functions. The green dotted line is the apex factor which has different values between 0 and 1.



3.2. The choice for the defuzzification method

The bottleneck part of the general type-2 fuzzy logic system is the defuzzification phase. This is due to the high computational burden associated with the type-reduction process. Therefore, special attention should be given to the choice of such methods. The aim of this subsection is to highlight this issue and its effects on the learning process. When using representation methods other than the vertical-slices representation, there are few proposed methods that have been used for this purpose. An example is a type-reducer proposed by [43] using triangular type-2 fuzzy sets which uses fuzzy

Figure 2: An example of learning a triangular SMF by adapting the apex location.



truth numbers where all the secondary membership functions are normal and convex for a unique entity in the secondary domain (primary membership functions). This type-reducer uses the iterative KM algorithm and some interpolation operations to get an approximate centroid for triangular type-2 fuzzy sets. Methods that use other representations include the one proposed in [32] which uses the iterative KM algorithm under the alpha-plane representation and the one based on z-slices in [44]. Based on our choice for the representation of general type-2 fuzzy sets using vertical-slices, the available defuzzifications options are :

1. The exhaustive brute-force highly expensive type-reduction method presented in [36, p.248-254] which computes the union of all the centroids of all the embedded type-2 fuzzy sets involved in the general type-2 fuzzy set. This method is impractical to use for our purpose. In practice, the number of embedded sets is normally astronomical and above the current data structure. For instance, for a general type-2 fuzzy sets discretized into our choice of 101 x-domain points and each vertical slice into 9 points, the number of embedded sets is 2.39×10^{96} which is far above the current data structure. In our chosen language (C++), the longest data structure size can be allocated is unsigned integer $= 2,14 \times 10^{9}$. This number of embedded sets are unioned to get *one* sample output in *one* fuzzy logic system evaluation in *one* iteration of the optimization process. Table 1 shows how type-reduction complexity evolves in our problem with some reasonable choices of fuzzy logic system input samples and reasonable number of simulated annealing iterations. Note that table 1 is for the type-reduction operations only (i.e. does not include fuzzification and other fuzzy logic system operations). Therefore, this choice is impractical for our work.
2. The recursive algorithm introduced by [17], which includes some interesting ideas to reduce these computations but whose complexity is still very high.
3. The vertical slice centroid type-reducer (VSCTR) which was initially proposed

Table 1: The number of centroid operations needed to type-reduce general type-2 fuzzy sets optimized by simulated annealing (SA).

X domain points	Y domain points	Number of embedded sets	FLS training samples number	SA iterations	Number of centroid operations needed
25	5	3e+17	200	10,000	6e+23
25	9	7.2e+23	200	10,000	1.44e+30
50	9	5.2e+47	200	10,000	1.04e+54
101	9	2.4e+96	200	10,000	4.8e+102

by [25] then detailed by [33]. It does not calculate the union for all the embedded sets involved in the general type-2 fuzzy sets. Although this method does not depend on the concept of embedded sets, it is a good approach for practical usage. This method works as follows:

- For each vertical slice, the centroid of each vertical slice is calculated exactly as type-1 set centroid calculation.
- The type-reduced set domain is the same as the vertical slices values. The membership grades of the type-reduced set are the centroids of these vertical slices in the type-reduced set.

When optimizing the FOU's and SMF's parameters using the non-deterministic sampling defuzzifier, the learning process is affected, to some degree, by the random errors and the fluctuations of the evaluation of the objective function. The effects come from the fact that the evaluation of one state will differ each time the sampling method approximates the type-reduced sets. Consequently, the outputs of the fuzzy logic system will be changed causing the objective function to get different energy values for the same state each time the evaluation is carried out. Whatever the objective function is, the outputs from the fuzzy logic system will affect that objective function. In fact, these random errors are small compared to the scale of the fuzzy logic system outputs and the scale of the objective function, but they nevertheless affect the learning performance as will be shown later in this paper. These effects can be ignored in the first exploration stages of the search when moves from state to state can bring relatively large differences, but this noise can deteriorate the search at the last stages when small effects of the objective function can be affected by this noise. In optimization, noise associated with the objective function has some effects on the quality of the solution.

In the simulated annealing literature, many papers have tackled the problem of noisy objective functions. All solutions proposed in the simulated annealing literature fall under these three categories [8]:

1. Solutions that adapt the convergence properties to allow better handling of noisy objective functions. These methods rely on storing all visited states and their evaluations or increasing the number of iterations according to a known schedule. This type of solution adds extra computations and needs larger memories to be executed.
2. Solutions that rely on revisiting each state a number of times to improve the approximation to the true objective function values, then using some statistical

approaches to calculate approximated objective functions. Again, this is computationally expensive, depending on the number of evaluations n needed for each state. Hence, the computations will be multiplied by n.

3. Solutions that adapt the acceptance function to maintain an adequate thermodynamic equilibrium.

Unfortunately, the three solutions require more computations and do not guarantee the exact objective functions. In general, when dealing with noisy objective functions, we are not interested in the exact best solutions. Rather, we are interested in alternatives to the best energy value that are nearly equally good [42, p.64]. The use of these methods adds another computational burden and does not lead to more accurate solutions. In order to get a fair comparison with interval type-2 fuzzy logic system, the use of such methods is not the best choice for our purpose. Therefore, a solution can be sought from the general type-2 fuzzy logic system side. A deterministic approach can be used to get the type-reduced set such as the vertical slice centroid type-reducer (VSCTR).

3.3. The Proposed Method of Learning

Our research proposal is to solve a two-stage optimization problem where in the first stage we search for possible configurations of FOU which can be used as bounds for the secondary domain of a general type-2 fuzzy set. The second stage is associated with the search through all the apex factors representing apex locations of the secondary membership function of a general type-2 fuzzy set. Using the proposed form of general type-2 set presented in section 3, we can design GT2FLS using the following two-stage procedure:

- The first step is to design the FOU of the general type-2 set while fixing the secondary membership function. This is done by defining FOU using any function used to define interval type-2 fuzzy sets. The lower and upper membership functions that bound the FOU in interval type-2 fuzzy sets can bound the FOU in general type-2 fuzzy sets. To get a good FOU, expert opinions or automated learning can be applied exactly as the case when designing IT2FLS.

- The second step involves learning the secondary membership functions of general type-2 sets. By fixing the optimal FOU, the secondary membership functions can be optimized. This is done by adapting the apex location indicators by a suitable value.

This two stage-method seems to be logical as the definition of the uncertainty boundaries (primary memberships) should precede the definition for how much secondary membership grades (uncertainty distribution) will be given to each primary membership.

The complexity of this problem stems from the size of the solution space, the nature of the functions to be optimized and the size of the rule base. The size of the solution space arises from the product of the possible apex factor domains and the number of possible partitions of the continuous domain for discretization of the apex factors. A solution is an array of apex factors which reside within FOU and minimizes the total error of modeling of the data into a general type-2 fuzzy logic system. Mathematically

if n observed/input values of x_1, x_2, x_n and output/target value of x^* are given, for any given step size of t_k and any possible partition of $x_{k1}, x_{k2}, ...x_{kn}$. $k = 1, 2, ., n$, our search process finds all apex factors $g_{ki}(x)$, $ki = 1, 2, , n$ which identify the secondary membership functions to generate a new output of the fuzzy logic system. Our aim is to minimize the objective function of the optimized problem. For example, an objective function that measures the total error of the output of the proposed method from given actual values for observed data.

4. Methodology

The experiments are divided into two main stages as described in subsection 3.3. In each stage the experiment is carried out in four steps: preparing data, constructing the initial interval and general type-2 fuzzy systems, learning the FOU parameters and learning the secondary membership functions. Hence, the optimization processes in the flowchart are repeated twice; one for IT2FLS and the second for GT2FLS. The flowchart of all stages stage is illustrated in Figure 3.

4.1. Data

4.1.1. Mackey-Glass time-series

The Mackey-Glass Time Series is a chaotic time series proposed in [34]. It is obtained from this non-linear equation :

$$\frac{dx(t)}{dt} = \frac{a * x(t-\tau)}{1 + x^n(t-\tau)} - b * x(t)$$

where a, b and n are constant real numbers, t is the current time and τ is the difference between the current time and the previous time $t - \tau$. To obtain the simulated data, the equation can be discretized using the Fourth-Order Runge-Kutta method. In the case where $\tau > 17$, it is known to exhibit chaos and has become one of the benchmark problems in soft computing [36, p.116]. To get the time series, firstly, the noise-free time series is generated with the following parameters : $a = 0.2$, $b = 0.1$, $\tau = 17$ and $n = 10$. The Runge-Kutta method is used to obtain the values of $x(t)$ at each time point with a time step of 0.1 and the initial condition $x(0) = 1.2$ where $x(t) = 0$ for $t < 0$. The input-output samples are extracted in the form $x(t - 18)$, $x(t - 12)$, $x(t - 6)$ and $x(t)$ from $t = 118$ to $t = 417$ using a step size of 6. Then the generated data are divided into 200 data points for training and the remaining 200 data points for testing. Using a step size of 6, the input values to the fuzzy system are the previous points $x(t - 18)$, $x(t - 12)$, $x(t - 6)$ and $x(t)$ while the output from the fuzzy system is the predicted value $x(t + 6)$. Four initial input values $x(114)$, $x(115)$, $x(116)$ and $x(117)$ are used to predict the first four training outputs. We have chosen 200 training samples only to complete the learning process in an acceptable time.

Adding some noise to the time series produces more challenges to the prediction task. In this experiment, a noisy time series will be used to test our models. The noisy Mackey-Glass time series will be generated by adding noise to Mackey-Glass time series that are generated as described above. The amount of noise will be 20db added to all inputs and outputs. The noise is measured by signal-to-noise ratio (SNR). Again, the number of training samples used here is 200.

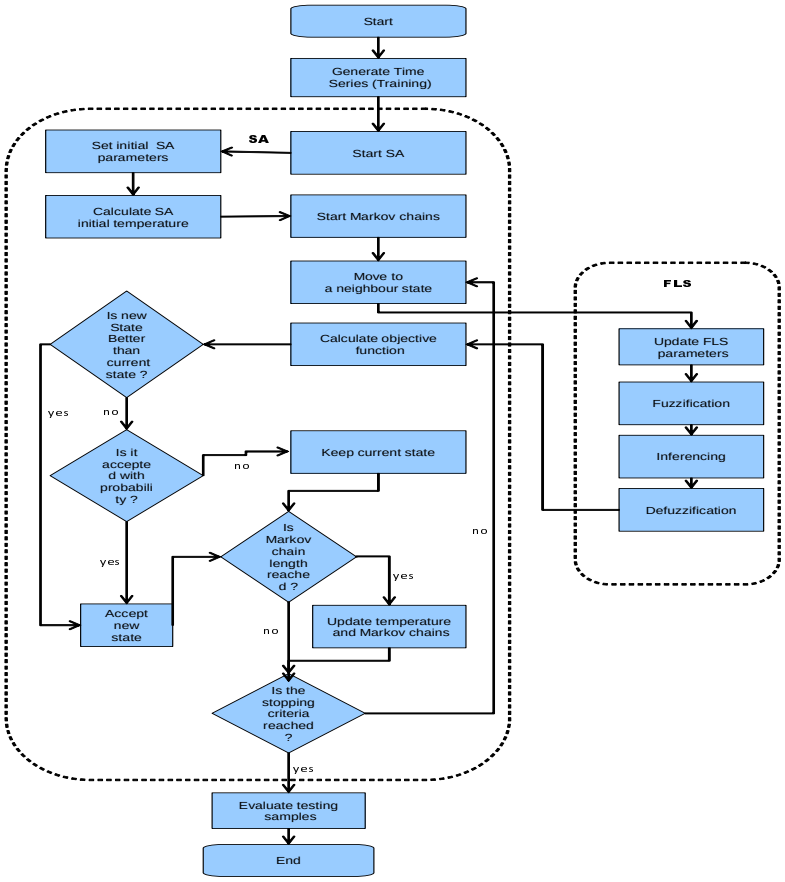


Figure 3: A flowchart of the method of using simulated annealing to optimize IT2FLS or GT2FLS.

4.1.2. Estimation of the low voltage electrical line length and maintenance costs in rural towns

Two problems concerning electrical distribution were proposed in [11] and serve as benchmark real-world problems in the fuzzy logic community. The first is concerned with finding a model that estimates the total length of low-voltage line installed in a rural town using some available information. The data consist of 495 samples in which the real data were measured by a company. Each sample has two inputs which are the number of inhabitants in the town and the mean of the distances from the center of the town to the three furthest clients in it while the output is the estimated length of low-voltage line. The data set has been taken from [9]. The data samples were divided into two sets labeled training and testing sets which are randomly selected from the whole sample as reported in [13] and [14]. As with other authors, 396 samples are used for

training while the other 99 samples are used for testing. The number of training samples was chosen to be the same as the others without a reduction due to the smaller number of inputs involved in this problem (2 instead of 4) which reduces the computational burden. The second related problem is to estimate the minimum maintenance costs of the medium-voltage electrical line based on a model of the optimal electrical network for some Spanish towns [11]. The problem has four input variables: sum of the lengths of all streets in the town, total area of the town, area that is occupied by buildings, and energy supply to the town while the output is the minimum maintenance cost. The data set consists of 1056 samples and has been taken from [9]. The data samples were randomly divided into two sets labeled training and testing sets which are randomly selected from the whole sample as reported in [13] and [14]. In order to reduce the training computations and time, the number of samples has been reduced from the one used by other authors. 400 data samples from the whole set were divided into two sets labeled training and testing sets with 200 samples for each set. Therefore, 400 samples have been used instead of 1056 samples.

4.2. The initial fuzzy logic systems

First we consider the interval case. The fuzzy model consists of a number of independent input fuzzy sets and one independent output fuzzy set for each rule. There are four rules while each rule is characterized by a number of fuzzy sets equal to the number of inputs (i.e. four antecedent fuzzy sets and one consequent fuzzy set). The system is built from scratch rather than using the optimized type-1 sets to initialize the interval type-2 fuzzy sets. Each type-2 fuzzy set is described by Gaussian primary membership functions with uncertain means represented by two means and one standard deviation as follow [36, p.91]:

$$\tilde{f}(x) = \exp^{-(\frac{x-m}{2\sigma})^2} \; m \in [m_1, m_2] \tag{11}$$

Therefore the upper $\overline{\mu}_{\tilde{A}}(x)$ and lower $\underline{\mu}_{\tilde{A}}(x)$ membership functions are defined by following mathematical functions [36, p.91]:

$$\overline{\mu}_{\tilde{A}}(x) = \begin{cases} \exp^{-(\frac{x-m_1}{2\sigma})^2} & \text{if } x < m_1 \\ 1 & \text{if } m_1 \leq x \leq m_2 \\ \exp^{-(\frac{x-m_2}{2\sigma})^2} & \text{if } x > m_2 \end{cases} \tag{12}$$

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} \exp^{-(\frac{x-m_2}{2\sigma})^2} & \text{if } x \leq \frac{m_1+m_2}{2} \\ \exp^{-(\frac{x-m_1}{2\sigma})^2} & \text{if } x > \frac{m_1+m_2}{2} \end{cases} \tag{13}$$

where the upper $\overline{\mu}_{\tilde{A}}(x)$ and lower $\underline{\mu}_{\tilde{A}}(x)$ membership functions in this equation are used to define FOU_{lower} and FOU_{upper}. All the means and standard deviations are initialized for all the input fuzzy sets by partitioning each input space into the chosen number of fuzzy sets and enabling enough overlapping between them while the output fuzzy sets are initialized randomly around the average value of training outputs. The fuzzification process is based on the minimum t-norm while the center-of-area has been chosen for type-reduction. The collapsing method proposed by [18] has been used to calculate the centroids of the interval type-2 sets needed to compute the center-of-area. This is done by using the composite outward right-left variant of the collapsing method

as it is described in [18]. The training procedure aims to learn the parameters of the antecedent parts and the consequent parts of the fuzzy system rules. The parameters found are then used to predict the next testing data points. Only the FOU's parameters are optimized in interval type-2 fuzzy sets. The initial general type-2 fuzzy logic system is built by using the proposed parameterization method described in section 3. The fuzzy model consists of a number of independent input fuzzy sets and one independent output fuzzy set for each rule. There are four rules while each rule is characterized by a number of antecedent fuzzy sets equal to the number of inputs (four antecedent fuzzy sets and one consequent fuzzy set). However, the number of rules was chosen heuristically and any number of rules can be chosen but we are interested in reducing the system's complexity and saving computations and time. The system is built from scratch rather than using optimized type-1 or interval type-2 fuzzy sets to initialize general type-2 fuzzy sets. The general type-2 sets are defined using their $FOU's$ and $SMF's$ functions as follows:

- FOU: The same membership functions used to define interval type-2 fuzzy sets in previous subsection (4.2) are used to define FOU parameters. The upper $\overline{\mu}_{\tilde{A}}(x)$ and lower $\underline{\mu}_{\tilde{A}}(x)$ membership functions in this equation are used to define FOU_{lower} and FOU_{upper}. All the means and standard deviations are initialized for all the input fuzzy sets by partitioning each input space into the chosen number of fuzzy sets and enabling enough overlapping between them while the output fuzzy sets are initialized randomly around the average value of training outputs.

- SMF : Our choice for the SMFs in this work is to use a triangular SMF with a normal apex initialized in the middle between (FOU_{lower} and FOU_{upper}) for $k_1, k_2, ...k_n$ points $(n = 9)$ by choosing their apex factors $g(k_1) = g(k_2) = ... = g(k_n) = 0.5$ and then calculating the apex locations for other x points using the linear interpolation function proposed in section 3. This method to parametrize the general type-2 set is shown in figure 1.

The configurations of IT2FLS and GT2FLS used in this experiment are detailed in Table 2. The initial general type-2 fuzzy logic system stages will be as follows:

- **Fuzzification** The fuzzification process will fuzzify each x value into a type-1 fuzzy set (SMF) which is a triangular function as described above. The fuzzified SMF is described by its FOU_{upper} and FOU_{lower} which are derived from the two Gaussian functions for x and its apex location indicator. The output from each fuzzification process is a triangular SMF.

- **Combination of antecedents** The combination between all antecedent fuzzified values is done using the meet operation proposed by [15]. The output from this phase is a convex SMF that might be non-normal.

- **Implication** To do the implication phase, firstly, the consequent sets space is discretized into $n = 101$ points $y_1, y_2, ..., y_n$ in Y domain. Then the implication is done using the same meet operation proposed by [15]. The third step is to do a join between all secondary membership grades for each $y \in Y$ using the join operation proposed in [15].

- **Type-Reduction** Two methods for type-reduction have been used: the embedded sets based sampling method and VSCTR method. In the sampling method, we used 100 samples of the embedded sets. The rationale for using two type-reduction methods is to test the true effects of learning SMF in general type-2 fuzzy sets without been distracted by the stochastic evaluation using sampling. The output from this phase is a type-1 fuzzy set.
- **Defuzzification** The center of area (centroid) defuzzification has been used in this part.

Table 2: The configurations of IT2FLS and GT2FLS used in this experiment

Stage	**IT2FLS**	**GT2FLS**
Membership Function	Gaussian	Gaussian + triangular SMF
Number of parameters (with four inputs)	60	60+180=240
fuzzification	singleton	singleton
Antecedent combination t-norm	minimum	minimum using Coupland's meet
Implication t-norm	minimum	minimum using Coupland's meet
Join t-conorm	maximum	maximum using Coupland's join
SMF discretized points	none	9
Type-reduction method	centroid by collapsing method	centroid by sampling and VSCTR
Defuzzification method	centroid	centroid
Y Descritization points	101	101

4.3. Learning of the FOU parameters

The training procedure aims to get the best parameters of the antecedent parts and the consequent parts of the fuzzy system rules. Then, the found parameters are used to predict the next testing data points. The total number of FOU parameters is 4 rules $*4$ antecedent fuzzy sets $*3$ parameters $+4$ rules $*3$ consequent set parameters $= 60$ in all problems except the line length problem where it is $4 * 2 * 3 + 4 * 3 = 36$ parameters. The learning process is done using the simulated annealing algorithm that searches for the best configuration of the parameters by trying to modify one parameter each time and evaluate the cost of the new state. The cost function that is used to measure the cost of the new state is the Root Mean Square Error (RMSE), defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}[f(k) - f(k^*)]^2} \tag{14}$$

From an optimization perspective, the only constraint to the variables of the optimization problem is that all standard deviations of all fuzzy sets must be ≥ 0. The simulated annealing algorithm is initialized with a temperature equal to the standard deviation of the mean RMSE for 1000 runs for the training samples as proposed by [48]. The cooling schedule is based on a static cooling rate of 0.9 updated for each Markov chain where this number is chosen from typical range to allow a fair exploration of the search space. Each Markov chain has a length related to the number of variables in the search

space as recommended by [2] which equals 5 times the number of variables. The search ends after 40 Markov chains which is enough to allow good convergence. The new states for a current state are chosen from neighboring states randomly by adding a small number (step size) to one of the antecedent parameters or the consequent parameters. The step size value is related to the maximum and minimum value for each input space and equal to max-min$/25$ so that the search space is divided into 25 discretized points for each value which allow a fair exploration with the available number of search iterations. The direction of the search is chosen randomly from right or left. After that, the new state is evaluated by examining the 200 output data points. Then, the average and the minimum of the cost function of the training and testing results have been calculated.

4.4. Learning of the SMF parameters

The learning process in this stage aims to get the optimal locations of apexes for all the SMF's parameters where the other two points for each triangular SMF are fixed. The optimized parameters in this case are the apex location factors $g(k_1), g(k_2), ..., g(k_n)$ for each general type-2 set involved in the system. The learning is done using simulated annealing algorithm with the same configuration used above apart from the following :

1. The constraints for each variable (apex factors $g(k_i)$ for each k_i) are defined by their ($FOU_{lower}(k_i)$ and $FOU_{upper}(k_i)$) points which constitute the primary memberships (secondary domain) boundaries for each k_i.
2. The step size is the value that changes the apex location indicator. The new value must be between $[0, 1]$ and the step size should be large enough to make a difference in the cost function as small values might not change the outputs when it does not overcome the next discretization step in the primary memberships (secondary domain). The chosen step size is 0.225.
3. The length of each Markov chain is equal to 5 times the number of variables in the search space. The search ends after 10 Markov chains. These choices are made to reduce the experiment's time. The choices of simulated annealing algorithm and Markov chains configurations are limited by the high computations and impracticality as justified in section 3.2.

The number of all parameters being optimized in this stage for each fuzzy set is $n = 9$. Therefore, the total number of all parameters being optimized in the system in this stage is (the number of fuzzy sets $* n$) parameters. That is 4 rules $*5$ sets $*9 = 180$ parameters in problems with 4 inputs and $4 * 3 * 9 = 108$ parameters in the length line problem. Therefore, the total number of parameters optimized in general type-2 fuzzy logic system is the sum of FOU and SMF optimized parameters. The experiment has been carried out 20 times and the average and the minimum of the cost function of the testing data results have been calculated. In addition, due to space limitation, we included only a sample of the data for the maintenance cost estimation problem for more clarification in Table 6 and samples of representative parameters for the maintenance cost estimation problem using VSCTR method as follows:

- Fuzzy sets parameters before FOU optimization in all rules for one run (Table 3).

- Fuzzy sets parameters after FOU optimization in all rules for one run (Table 4).
- SMF parameters after SMF optimization using VSCTR for the first rule only (Table 5).

Table 3: Interval type-2 fuzzy sets parameters before FOU optimization using SA in all rules for one run for the maintenance cost estimation problem.

Initial Gaussian first mean					
Rule	fs_{mean-1}	fs_{mean-2}	fs_{mean-3}	fs_{mean-4}	$fs_{consequent-mean}$
1	3.125	2.25	36.855	36.375	2318.478
2	5.75	4.35	72.07	71.75	2119.751
3	8.375	6.45	107.285	107.125	2086.63
4	11	8.55	142.5	142.5	2163.913
Initial Gaussian second mean					
Rule	fs_{mean-1}	fs_{mean-2}	fs_{mean-3}	fs_{mean-4}	$fs_{consequent-mean}$
1	3.25625	2.355	38.61575	38.14375	2689.434
2	5.88125	4.455	73.83075	73.51875	2490.707
3	8.50625	6.555	109.0458	108.8937	2457.586
4	11.13125	8.655	144.2608	144.2688	2534.869
Initial Gaussian standard deviations					
Rule	fs_{std-1}	fs_{std-2}	fs_{std-3}	fs_{std-4}	$fs_{consequent-std}$
1	2.625	2.1	35.215	35.375	370.9564
2	2.625	2.1	35.215	35.375	370.9564
3	2.625	2.1	35.215	35.375	370.9564
4	2.625	2.1	35.215	35.375	370.9564

5. Results and Discussion

The experiments were developed using the C++ language and have been carried out 20 times on a number of PCs with an equal CPU speed of 3 GHz and a memory of 4GB. The results are shown for each problem below and are summarized for all problems in tables 12 and 11. Extra insights into the convergence behaviors and acceptance ratios in both stages will be discussed to explain some new results. The acceptance ratio is the proportion of moves that are accepted. In typical implementations of simulated annealing, acceptance ratios start close to 1 and decrease towards zero.

5.1. Mackey-Glass time series results

The results of learning Mackey-Glass time series are detailed in table 7 where the average RMSEs curves and the acceptance ratios during search are depicted in figures 4 and 5 respectively. The main observations are :

1. The best average RMSE in testing samples was obtained by a general type-2 fuzzy logic system with VSCTR defuzzification (GT2FLS-VSCTR) followed by interval type-2 fuzzy logic system (IT2FLS).
2. The best average RMSE in training samples was obtained by a general type-2 fuzzy logic system with VSCTR defuzzification followed by IT2FLS.

Table 4: Interval type-2 fuzzy sets parameters after FOU optimization using SA in all rules for one run for the maintenance cost estimation problem.

Optimized Gaussian first mean					
Rule	fs_{mean-1}	fs_{mean-2}	fs_{mean-3}	fs_{mean-4}	$fs_{consequent-mean}$
1	2.705	4.602	70.6614	-31.545	-56.3592
2	4.49	6.702	128.414	54.77	7887.21
3	7.535	3.762	180.532	39.205	7854.09
4	11	12.918	181.941	216.08	-5978.39
Optimized Gaussian second mean					
Rule	fs_{mean-1}	fs_{mean-2}	fs_{mean-3}	fs_{mean-4}	$fs_{consequent-mean}$
1	4.93625	1.011	72.4222	55.1237	-363.928
2	4.62125	5.463	62.562	39.5588	7918.91
3	7.66625	2.523	86.5082	193.794	-595.775
4	4.41125	14.703	211.874	161.249	2195.61
Optimized Gaussian standard deviations					
Rule	fs_{std-1}	fs_{std-2}	fs_{std-3}	fs_{std-4}	$fs_{consequent-std}$
1	4.305	9.156	29.5806	29.715	1049.48
2	4.305	1.764	18.3118	205.175	1728.01
3	3.045	0.084	23.9462	58.015	2067.27
4	8.505	2.772	181.709	7.075	710.219

3. The average RMSEs curves when learning FOUs (training samples) have exhibited similar performances by the three models. However, IT2FLS obtained the best average RMSEs in testing phase followed by GT2FLS-VSCTR which was the best in training phase followed by IT2FLS.
4. The learning of SMFs using GT2FLS-VSCTR adds about 11.7% to the average testing RMSEs and about 17.7% to the average training RMSEs over the FOU's learning best results. The learning of SMFs using GT2FLS-Sampling adds about 0.86% to the training RMSEs but worsened the testing RMSEs by about -0.059.
5. The learning curves of SMFs showed a clear difference in performance between GT2FLS-VSCTR and GT2FLS-Sampling models. GT2FLS-VSCTR shows continuous improvements compared to very small improvements obtained by GT2FLS-Sampling.
6. The acceptance ratio curves when learning FOUs show similar behaviors between GT2FLS-VSCTR and IT2FLS better than the narrower acceptance behavior obtained by GT2FLS-Sampling. The last one shows poor performance where it converges to values close to 0% quickly in less than 30 Markov chains which means no improvements were observed in the rest of iterations.
7. The acceptance ratio curves when learning SMFs show a clear difference in behaviors between GT2FLS-VSCTR and GT2FLS-Sampling models. The GT2FLS-Sampling shows undesirable very wide acceptance behavior compared to a narrower one by GT2FLS-VSCTR. Interestingly, the acceptance ratios curves of GT2FLS-Sampling model show a different behavior when learning FOUs from its behavior with SMF. However, as mentioned above, the initial temperatures were set separately in each stage to be proportional to the objective function differences brought by these moves in the two parameters groups (FOU and SMF). This is important to avoid starting with very large or very small initial tempera-

Table 5: A sample of general type-2 fuzzy sets parameters after SMF optimization in the first rule for one run for the maintenance cost estimation problem. The K points are n=9 points for each fuzzy set and associated with values that are distributed equally in the the primary domain. The apex factor values (AF) are shown before and after optimization.

Rule	Fuzzy set	K point #	K value	SMF lower	Initial AF	optimized AF	SMF upper
1	1	1	-10.21	0.00205148	0.5	0.725	0.011109
1	1	2	-6.70234	0.0258751	0.5	0.05	0.0918518
1	1	3	-3.19469	0.168027	0.5	0.95	0.391005
1	1	4	0.312969	0.561768	0.5	0.275	0.856957
1	1	5	3.82063	0.966979	0.5	0.275	1
1	1	6	7.32828	0.561768	0.5	0.5	0.856957
1	1	7	10.8359	0.168027	0.5	0.275	0.391005
1	1	8	14.3436	0.0258751	0.5	0.275	0.0918518
1	1	9	17.8513	0.00205148	0.5	0.275	0.011109
1	2	1	-26.457	0.00317161	0.5	0.05	0.011109
1	2	2	-19.1411	0.0346561	0.5	0.275	0.0887311
1	2	3	-11.8253	0.19999	0.5	0.95	0.374287
1	2	4	-4.50938	0.609487	0.5	0.5	0.833802
1	2	5	2.8065	0.980956	0.5	0.05	1
1	2	6	10.1224	0.609487	0.5	0.275	0.833802
1	2	7	17.4383	0.19999	0.5	0.5	0.374287
1	2	8	24.7541	0.0346561	0.5	0.5	0.0887311
1	2	9	32.07	0.00317161	0.5	0.95	0.011109
1	3	1	-18.0804	0.00927583	0.5	0.725	0.011109
1	3	2	4.32514	0.0706658	0.5	0.95	0.0809004
1	3	3	26.7307	0.303322	0.5	0.95	0.331944
1	3	4	49.1362	0.733562	0.5	0.275	0.767392
1	3	5	71.5418	0.999557	0.5	0.275	1
1	3	6	93.9473	0.733562	0.5	0.275	0.767392
1	3	7	116.353	0.303322	0.5	0.725	0.331944
1	3	8	138.758	0.0706658	0.5	0.275	0.0809004
1	3	9	161.164	0.00927583	0.5	0.95	0.011109

tures and to have acceptable curves of best results and acceptance ratios. In other words, the observed acceptance behaviors for the GT2FLS-Sampling model are not related to the settings of simulated annealing. This behavior can be easily explained by the effects of the defuzzification method which is the only difference between the two models of GT2FLS. As explained in section 3.2, the effects of the stochastic objective function when using sampling method can be ignored when moves from state to state can bring relatively large differences compared to the random noise but this noise can deteriorate the search when moves bring improvements comparable to that noise. In other words, when learning FOU, the differences brought by moves are large enough to accept very small errors of approximated objective functions due to the larger contributions of the FOU's parameters on the objective functions compared to the SMF contributions. Hence, we do not expect large contribution from learning SMF's parameters compared to FOU's learning due to the fact that SMF is dependent on FOU and bounded by its endpoints. This behavior of acceptance ratios when using GT2FLS-Sampling have been observed with all problems and this explanation is applied to them.

Table 6: A sample of the data used for the maintenance cost estimation problem.

Input 1	Input 2	Input 3	Input 4	Output
11	3.3	54.959999	55	4329.330078
4	1.2	19.98	40	2016.439941
0.9	0.27	4.5	1.8	249.419998
2	1.2	19.98	10	1044.219971
2	1.8	19.98	30	1761.920044
2.5	1.5	24.959999	25	2028.640015
9.5	2.85	47.459999	19	3093.179932
5	1.5	16.65	10	964.52002
6.5	5.85	97.5	65	5782.939941
5	1.5	24.959999	25	2101.409912
2.5	0.75	12.48	25	1445
9.5	5.7	94.980003	95	6857.439941

8. The time taken by IT2FLS was the shortest, i.e. 5.8 times faster than GT2FLS-VSCTR and 21.8 times faster than GT2FLS-Sampling. Therefore, IT2FLS is preferred in terms of speed.

5.2. Mackey-Glass time series with added noise results

The results of learning a Mackey-Glass time series with added noise are detailed in table 8 where the average RMSE's curves and the acceptance ratios during search are depicted in figures 6 and 7 respectively. The main observations from the results are :

1. The best average RMSE in the testing samples was obtained by GT2FLS-VSCTR followed by GT2FLS-Sampling.
2. The best average RMSE in training samples was obtained by GT2FLS-VSCTR followed by GT2FLS-Sampling.
3. The average RMSE's curves for learning FOUs (training samples) have exhibited similar performances by the three models. However, GT2FLS-VSCTR model obtained best average RMSEs in training and testing followed by GT2FLS-Sampling.
4. The learning of SMFs using GT2FLS-VSCTR adds about 1% to the average testing RMSEs and about 3% to the average training RMSEs over the FOU's learning best results. The learning of SMFs using GT2FLS-Sampling adds about 0.32% to the training RMSEs but worsened the testing RMSEs by about -0.1.
5. The learning curves of SMFs showed a clear difference in performance between GT2FLS-VSCTR and GT2FLS-Sampling models. GT2FLS-VSCTR shows continuous improvements compared to relatively small improvements obtained by GT2FLS-Sampling.
6. The acceptance ratio curves when learning FOUs show similar behaviors between GT2FLS-VSCTR and IT2FLS, better than the narrower acceptance behavior obtained by GT2FLS-Sampling. The latter converges to acceptance ratios close to 0% in less than 25 Markov chains.

Table 7: The forecasting results for noise-free Mackey-Glass time series by simulated annealing with GT2FLS

Stage	$Mean_{RMSE}$	Std_{RMSE}	$Minimum_{RMSE}$
IT2FLS			
Training	0.04980955	0.0200348	0.026242
Testing	0.0433439	0.010239	0.027117
Time (seconds)	332.55	21.027488	295
GT2FLS with Sampling Defuzzification			
After FOU's Learning			
Training	0.0553228125	0.01243	0.03761027
Testing	0.0518645455	0.0107249	0.03617023
After SMF's Learning			
Training	0.0548446	0.0119293	0.0372725
Improvement by SMF	0.86%	-	-
Testing	0.051895285	0.010721	0.0362123
Improvement by SMF	-0.059269 %	-	-
Time (seconds)	7,259.9	992.126	5,724
GT2FLS with VSCTR Defuzzification			
After FOU's Learning			
Training	0.0483079765	0.01089	0.03428513
Testing	0.0446682685	0.0121448	0.02823214
After SMF's Learning			
Training	0.03975027	0.0115896	0.0240021
Improvement by SMF	17.7%	-	-
Testing	0.03943346	0.0116557	0.024325
Improvement by SMF	11.7%	-	-
Time (seconds)	1,945.45	368.392	1,217

Figure 4: The average convergence of the method using the three models for noise-free Mackey-Glass time series problem when learning FOU (left) and SMF (right).

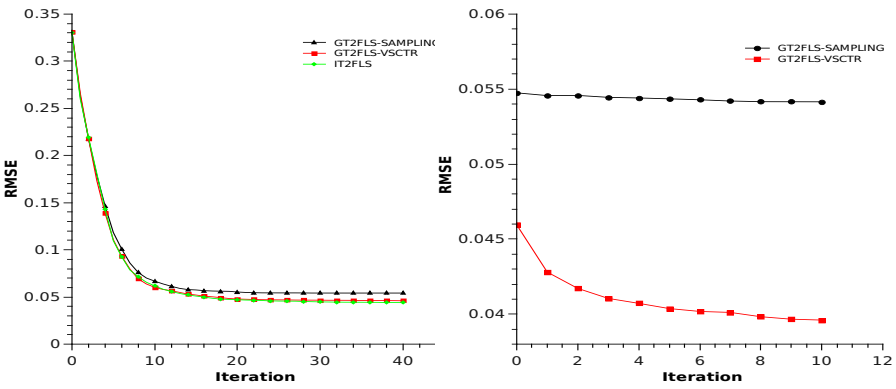
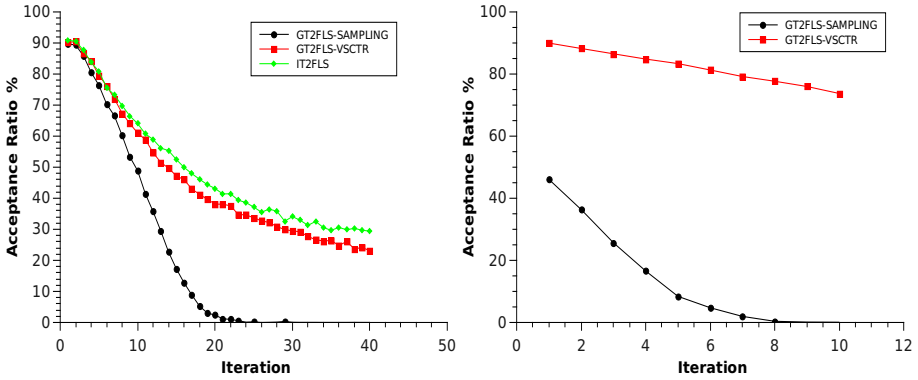


Figure 5: The average acceptance ratios of the three models when learning FOU (left) and SMF (right) for noise-free Mackey-Glass time series problem.



7. The acceptance ratio curves when learning SMFs show a clear difference in behaviors between GT2FLS-VSCTR and GT2FLS-Sampling models. The GT2FLS-Sampling shows very wide acceptance behavior compared to a narrower one by GT2FLS-VSCTR.
8. The time taken by IT2FLS was the shortest at 6.3 times faster than GT2FLS-VSCTR and 20 times faster than GT2FLS-Sampling.

5.3. The low voltage electrical line length results

The results of the learning low voltage electrical line length problem are detailed in table 9 where the average RMSEs curves and the acceptance ratios during search are depicted in figures 8 and 9 respectively. The main observations from the results are :

1. The best average RMSE in testing samples was obtained by GT2FLS-VSCTR followed by GT2FLS-Sampling.
2. The best average RMSE in training samples was obtained by GT2FLS-VSCTR followed by GT2FLS-Sampling.
3. The average RMSE's curves for learning FOUs (training samples) have exhibited similar performances by the three models. However, the GT2FLS-VSCTR model obtained the best average RMSEs in training and testing. The second in testing was GT2FLS-Sampling while IT2FLS was the second in training.
4. The learning of SMFs using GT2FLS-VSCTR adds about 0.88% to the average testing RMSEs and about 4.9% to the average training RMSEs over the FOU's learning best results. The learning of SMFs using GT2FLS-Sampling adds about 0.05% to the testing RMSEs and about 3.15% to the training RMSEs after FOU's learning.

Table 8: The forecasting results for Mackey-Glass time series with added noise by simulated annealing with IT2FLS and GT2FLS

Stage	$Mean_{RMSE}$	Std_{RMSE}	$Minimum_{RMSE}$
IT2FLS			
Training	0.1468528	0.02459	0.125778
Testing	0.1525942	0.014847	0.126217
Time (seconds)	350	57.217	285
GT2FLS with Sampling Defuzzification			
After FOU's Learning			
Training	0.13835616	0.00835	0.1282838
Testing	0.14152867	0.008688	0.1242401
After SMF's Learning			
Training	0.13790745	0.008148	0.128318
Improvement by SMF	0.32%	-	-
Testing	0.1416725	0.0086397	0.124408
Improvement by SMF	-0.1%	-	-
Time (seconds)	6,999.45	1,107.276	4,645
GT2FLS with VSCTR Defuzzification			
After FOU's Learning			
Training	0.132636905	0.0045462	0.123189
Testing	0.138335225	0.004243	0.1287115
After SMF's Learning			
Training	0.12860905	0.004316	0.120457
Improvement by SMF	3%	-	-
Testing	0.136965	0.0043126	0.128493
Improvement by SMF	1%	-	-
Time (seconds)	2,197.5	426.4706	1,460

Figure 6: The average convergence of the method using the three models for noise-free Mackey-Glass time series with added noise problem when learning FOU (left) and SMF (right).

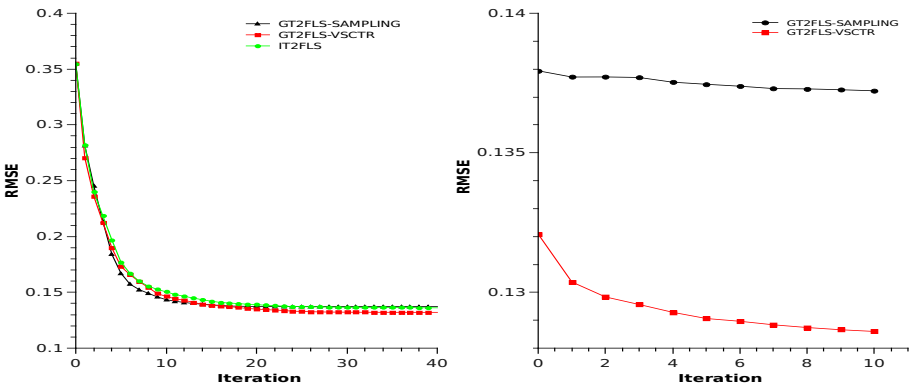
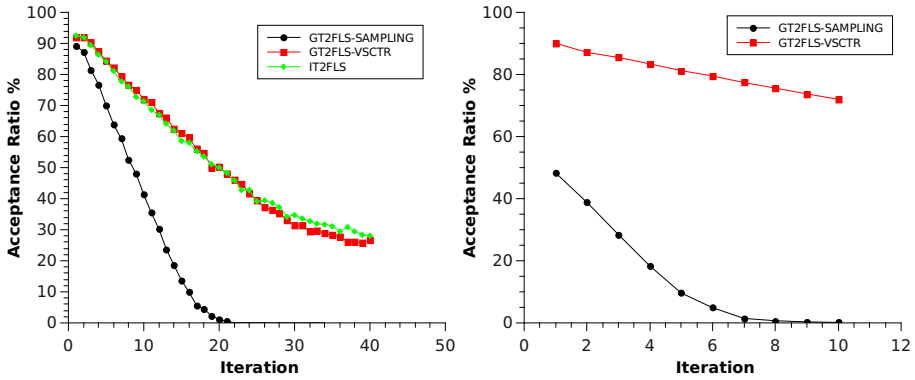


Figure 7: The average acceptance ratios of the three models when learning FOU (left) and SMF (right) for noisy Mackey-Glass time series problem.



5. The learning curves of SMFs showed a clear difference in performance between GT2FLS-VSCTR and GT2FLS-Sampling models. GT2FLS-VSCTR shows continuous improvements compared to relatively small improvements obtained by GT2FLS-Sampling.
6. The acceptance ratio curves when learning FOUs show similar behaviors between GT2FLS-VSCTR and IT2FLS better than acceptance behavior obtained by GT2FLS-Sampling. The last one converges to acceptance ratios close to 0% in less than 25 Markov chains.
7. The acceptance ratio curves when learning SMFs show a clear difference in behaviors between GT2FLS-VSCTR and GT2FLS-Sampling models. The GT2FLS-Sampling shows very wide acceptance behavior compared to a narrower one by GT2FLS-VSCTR.
8. The time taken by IT2FLS was the shortest at 3.8 times faster than GT2FLS-VSCTR and 17.3 times faster than GT2FLS-Sampling.

5.4. The maintenance cost problem results

The results of learning the maintenance cost problem are detailed in table 10 where the average RMSE's curves and the acceptance ratios during search are depicted in figures 10 and 11 respectively. The main observations are :

1. The best average RMSE in testing samples was obtained by GT2FLS-VSCTR followed by interval type-2 fuzzy logic system.
2. The best average RMSE in training samples was obtained by GT2FLS-VSCTR followed by interval type-2 fuzzy logic system.
3. The average RMSE's curves for learning FOUs (training samples) have exhibited similar performances by the three models. Again, GT2FLS-VSCTR model obtained best average RMSEs in both training and testing followed by IT2FLS.

Table 9: The estimation results for low voltage electrical line length by simulated annealing with IT2FLS and GT2FLS

Stage	$Mean_{RMSE}$	Std_{RMSE}	$Minimum_{RMSE}$
IT2FLS			
Training	627.816	64.10956	580.319
Testing	606.84075	62.6282	568.15
Time (seconds)	530.8	47.5987	463
GT2FLS with Sampling Defuzzification			
After FOU's Learning			
Training	632.080425	50.4127	595.8498
Testing	594.33905	16.46317	562.3377
After SMF's Learning			
Training	612.13475	10.24457	593.864
Improvement by SMF	3.15%	-	-
Testing	594.02365	16.2959	560.929
Improvement by SMF	0.05%	-	-
Time (seconds)	9,162.3	2,521.752	4,377
GT2FLS with VSCTR Defuzzification			
After FOU's Learning			
Training	618.412695	52.09535	577.1892
Testing	596.185465	19.2559	571.3894
After SMF's Learning			
Training	588.01895	11.6174	564.773
Improvement by SMF	4.9%	-	-
Testing	590.90565	18.40509	559.914
Improvement by SMF	0.88%	-	-
Time (seconds)	2,005.65	367.299	1,474

Figure 8: The average convergence of the method using the three models for low voltage electrical line length problem when learning FOU (left) and SMF (right).

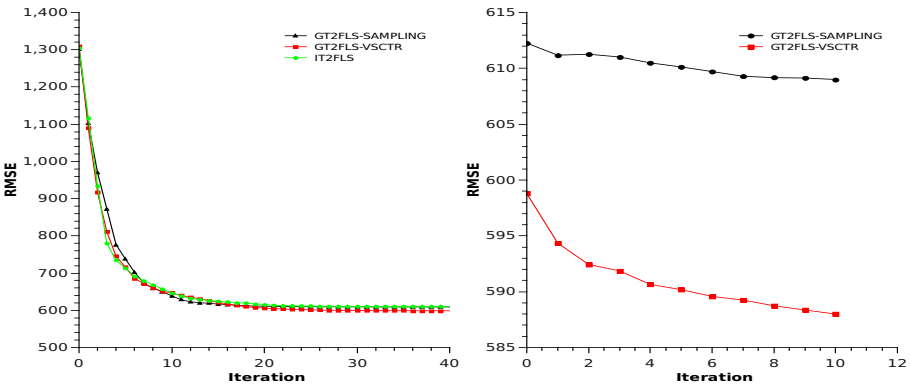
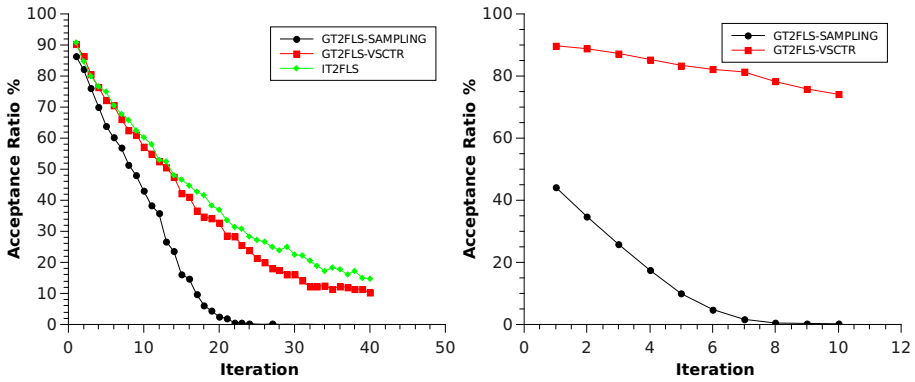


Figure 9: The average acceptance ratios of the three models when learning FOU (left) and SMF (right) for low voltage line problem



4. The learning of SMFs using GT2FLS-VSCTR adds about 6.9% to the average testing RMSEs and about 14.9% to the average training RMSEs over the FOU's learning best results. The learning of SMFs using GT2FLS-Sampling adds about 3.35% to the training RMSEs but worsened the testing RMSEs by about -0.05%.
5. The learning curves of SMFs showed a clear difference in performance between GT2FLS-VSCTR and GT2FLS-Sampling models. GT2FLS-VSCTR shows continuous improvements compared to relatively very small improvements obtained by GT2FLS-Sampling.
6. The acceptance ratio curves when learning FOUs show similar behaviors between GT2FLS-VSCTR and IT2FLS better than acceptance behavior obtained by GT2FLS-Sampling. The latter converges to acceptance ratios close to 0% in less than 25 Markov chains.
7. The acceptance ratio curves when learning SMFs show a clear difference in behaviors between GT2FLS-VSCTR and GT2FLS-Sampling models. The GT2FLS-Sampling shows very wide acceptance behavior compared to a narrower one by GT2FLS-VSCTR.
8. The time taken by IT2FLS was the shortest at 3.77 times faster than GT2FLS-VSCTR and 14.4 times faster than GT2FLS-Sampling.

5.5. Results summary

The main conclusions from the results for the four problems are :

1. The GT2FLS-VSCTR model obtained the best results in all cases for both training and testing results (average RMSEs).

Table 10: The estimation results for the maintenance cost problem by simulated annealing with IT2FLS and GT2FLS

Stage	$Mean_{RMSE}$	Std_{RMSE}	$Minimum_{RMSE}$
IT2FLS			
Training	304.8366	92.320619	145.985
Testing	353.99755	106.36379	207.672
Time (seconds)	410.95	44.00535	341
GT2FLS with Sampling Defuzzification			
After FOU's Learning			
Training	347.56075	93.88004	172.6323
Testing	424.139295	107.8478	230.5775
After SMF's Learning			
Training	335.91655	81.6298	172.514
Improvement by SMF	3.35%	-	-
Testing	424.3692	108.0096	229.275
Improvement by SMF	-0.05%	-	-
Time (seconds)	5,936.6	937.4517	4,037
GT2FLS with VSCTR Defuzzification			
After FOU's Learning			
Training	281.416145	96.938	124.3031
Testing	341.1021	112.5648	155.5801
After SMF's Learning			
Training	239.4284	76.2998	109.223
Improvement by SMF	14.9%	-	-
Testing	317.43325	104.8642	145.222
Improvement by SMF	6.9%	-	-
Time (seconds)	1,556.25	183.7191	1,260

Figure 10: The average convergence of the method using the three models for the maintenance cost problem when learning FOU (left) and SMF (right).

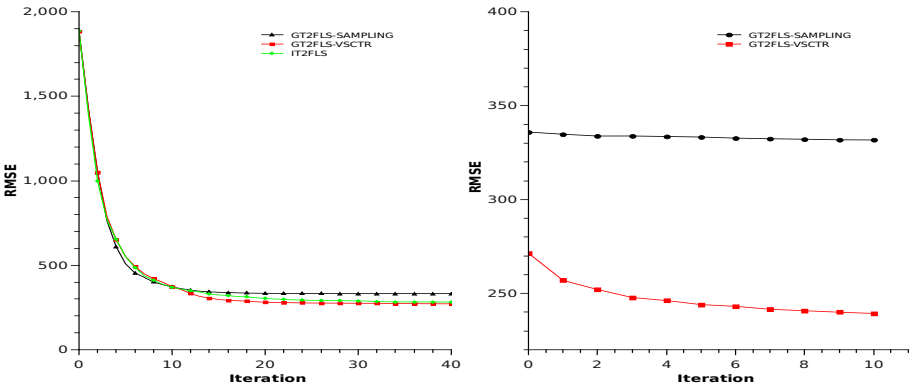
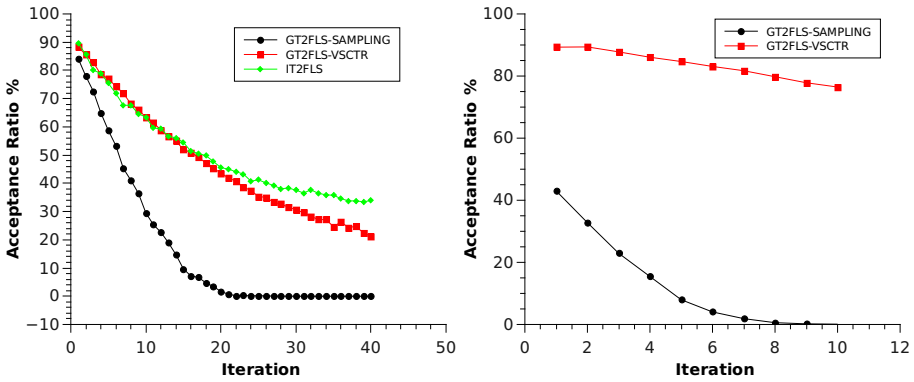


Figure 11: The average acceptance ratios of the three models when learning FOU (left) and SMF (right) for maintenance cost problem



2. GT2FLS-Sampling and IT2FLS were overlapping the second position and therefore showing similar results. However, the learning of SMF when using sampling defuzzification was adversely affected by the stochastic behavior for the objective function. In fact, it is unfair to compare a model with a stochastic evaluation (GT2FLS-Sampling) with a model with a deterministic evaluation (IT2FLS) of the objective function. Therefore, the question of whether general type-2 fuzzy logic systems or IT2FLS is better in handling uncertainties should not be based on such a case.
3. When learning FOU, GT2FLS-VSCTR obtained the best results in all training cases and most testing cases (three out of four) against IT2FLS. This might be explained intuitively by the uncertainties in practice that are centered and distributed around some points in the SMF. In interval type-2 fuzzy set, all uncertainties are given the same amount of possibilities in their SMF. Therefore, in general, general type-2 fuzzy sets should be able in practice to handle more information than interval type-2 fuzzy sets even without learning their SMF.
4. The learning of SMFs using GT2FLS-VSCTR adds between 0.88% and 11.7% to the average testing RMSEs and between 3% to 17.7% to the average training RMSEs over the FOU's learning best results. The learning of SMFs using GT2FLS-Sampling adds between 0.32% and 3.35% to the training RMSEs and up to 0.5% to the testing RMSEs but also worsen some results by up to -1%. Again, the comparison against GT2FLS-Sampling should not be tackled between such models. In other words, the learning of SMFs has brought noticeable improvements when allowing a deterministic method of evaluation of objective functions meaning that general type-2 fuzzy logic systems add more abilities and flexibilities to modeling than interval type-2 fuzzy logic systems. This observation enforces the assumption that the third dimension in GT2FS should enhance the modeling ability over the uniform and restricted type-2 in-

Table 11: Results summary for the three models ordered by accuracy (1= the best).

Model	Mackey-Glass		Noisy Mackey-Glass		Line length		Maintenance cost	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
IT2FLS	2	2	3	3	3	3	2	2
GT2FLS-Sampling	3	3	2	2	2	2	3	3
GT2FLS-VSCTR	1	1	1	1	1	1	1	1

Table 12: The results summary for all problems by the three models to model testing samples (the best are in bold font)

Problem	IT2FLS		GT2FLS-Sampling		GT2FLS-VSCTR	
	$Mean_{RMSE}$	Time	$Mean_{RMSE}$	Time	$Mean_{RMSE}$	Time
Mackey-Glass	0.0433439	**332.55**	0.051895285	7,259.9	**0.03943346**	1,945
Noisy Mackey-Glass	0.1525942	**350**	0.1416725	6,999.45	**0.136965**	2,197.5
Line length	606.84075	**530.8**	594.02365	9,162.3	**590.90565**	2,005.65
Maintenance cost	353.99755	**410.95**	424.3692	5,936.6	**317.43325**	1,556.25

terval fuzzy sets when a good configuration is chosen through an optimization process. The problem of type-reduction in general type-2 fuzzy logic systems should be investigated further to find embedded sets based on practical and stable methods to help the automatic design of general type-2 fuzzy logic systems.

5. The stochastic evaluations of centroids using the sampling method affects the learning performance of general type-2 fuzzy logic systems especially when learning SMF. These effects are shown through their learning and acceptance behavior curves.
6. The time taken by the interval type-2 fuzzy logic system was the shortest at $3.77 - 6.3$ times faster than GT2FLS-VSCTR and $14.4 - 21.8$ times faster than GT2FLS-Sampling. Therefore, in terms of speed, IT2FLS is preferred followed by GT2FLS-VSCTR.
7. Due to the complexity of the solution space in GT2FLS, most researchers have only focused on modeling practical problems using more limited type-1 fuzzy logic system (T1FLS) and IT2FLS. In our previous work [5], we presented a comparison between IT2FLS and T1FLS results and, for this reason, in this paper we only focus on comparing different versions of GT2FLS with IT2FLS. Our research shows that an efficient optimization algorithm for finding and fine-tuning parameters enables the use of GT2FLS to model even more complex problems within a comparable time. Tables 11 and 12 clearly show that our final GT2FLS-VSCTR version achieved best accuracy in all cases and so provides a clear indication of the strength of the algorithm compared to others.

6. Conclusion and Future Work

Work on the incorporation of learning in general type-2 fuzzy logic systems (GT2FLSs) using simulated annealing (SA) has been reported. The learning has been applied to the

configuration of all general type-2 fuzzy logic system parameters in two stages in both the footprint of uncertainty (FOU) and the secondary membership functions (SMF) parts. The learning process starts from scratch rather than using optimized interval type-2 fuzzy logic systems (IT2FLSs) to initialize general type-2 fuzzy logic systems. The novel parametrization approach presented in this work has been used to design two models of general type-2 fuzzy logic systems. These two models used two type-reduction techniques: non-deterministic (the sampling method) and deterministic (the vertical-slices centroid type-reduction (VSCTR). The rationale for using the two type-reduction techniques has been described. In addition, both models as well as interval type-2 fuzzy logic system model have been been applied to model the four problems.

The question of whether general type-2 fuzzy logic systems can enhance the ability to handle information has been tackled. The stochastic defuzzification method of sampling embedded sets affects the learning performance in both FOU and SMF learning stages. However, general type-2 fuzzy logic systems with sampling defuzzification have achieved similar performance to interval type-2 fuzzy logic systems but in a very long time. The best results achieved in all problems have been accredited to general type-2 fuzzy logic systems with VSCTR defuzzification. The results showed that when using the deterministic defuzzification method (VSCTR), the learning of general type-2 fuzzy logic systems can provide extra abilities to handle more information and uncertainties than interval type-2 fuzzy logic systems that use uniform SMFs. Although the use of VSCTR is not based on the concept of using embedded sets to calculate the exact centroids of type-2 sets, the method allows the learning process to be carried out in a practical manner. This achievement opens the door to using other learning methods to obtain greater modeling capabilities from general type-2 fuzzy logic systems in real-world applications.

7. References

[1] E. Aarts, J. Lenstra, Local search in combinatorial optimization, Princeton Univ Press, 2003.

[2] E. H. L. Aarts, H. M. M. T. Eikelder, Simulated annealing, in: P. Pardalos, M. Resende (eds.), Handbook of applied optimization, Oxford University Press, 2002, pp. 209–220.

[3] M. Almaraashi, R. John, Tuning of type-2 fuzzy systems by simulated annealing to predict time series, in: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2011 , WCE 2011, vol. 2, Newswood Limited, London, U.K, 2011, pp. 976–980.

[4] M. Almaraashi, R. John, Tuning type-2 fuzzy systems by simulated annealing to estimate maintenance cost, in: proceedings the UKCI 2011, Manchester, 2011.

[5] M. Almaraashi, R. John, S. Ahmadi, Electrical engineering and intelligent systems book, in: S. I. Ao, L. Gelman (eds.), Learning of Type-2 Fuzzy Logic Systems by Simulated Annealing with Adaptive Step Size, vol. 130 of Lecture Notes in Electrical Engineering, chap. 5, Springer, 2012.

[6] M. Almaraashi, R. John, S. Coupland, Designing generalised type-2 fuzzy logic systems using interval type-2 fuzzy logic systems and simulated annealing, in: Fuzzy Systems (FUZZ), 2012 IEEE International Conference on, IEEE, 2012.

[7] M. Almaraashi, R. John, A. Hopgood, Automatic learning of general type-2 fuzzy logic systems using simulated annealing, in: Fuzzy Systems (FUZZ), 2014 IEEE International Conference on, IEEE, 2014, accepted.

[8] J. Branke, S. Meisel, C. Schmidt, Simulated annealing in the presence of noise, Journal of Heuristics 14 (6) (2008) 627–654.

[9] G. Casillas, Fuzzy modeling library (fmlib), Available at `http://decsai.ugr.es/ casillas/fmlib/index.html`, [Accessed: 28 March 2011] (2011).

[10] H. H. Christian Wagner, Novel methods for the design of general type-2 fuzzy sets based on device characteristics and linguistic labels surveys, in: 2009 IFSA World Congress, EUSFLAT World Conference, Lisbon, Portugal, 2009, pp. 537–543.

[11] O. Cordón, F. Herrera, L. Sánchez, Solving electrical distribution problems using hybrid evolutionary data analysis techniques, Applied Intelligence 10 (1) (1999) 5–24.

[12] O. Cordón, F. Herrera, P. Villar, Analysis and guidelines to obtain a good uniform fuzzy partition granularity for fuzzy rule-based systems using simulated annealing, International Journal of Approximate Reasoning 25 (3) (2000) 187–215.

[13] O. Cordón, F. Herrera, P. Villar, Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base, Fuzzy Systems, IEEE Transactions on 9 (4) (2001) 667–674.

[14] O. Cordón, F. Herrera, I. Zwir, Linguistic modeling by hierarchical systems of linguistic rules, Fuzzy Systems, IEEE Transactions on 10 (1) (2002) 2–20.

[15] S. Coupland, R. John, Geometric type-1 and type-2 fuzzy logic systems, Fuzzy Systems, IEEE Transactions on 15 (1) (2007) 3 –15.

[16] L. Drack, H. Zadeh, Soft computing in engineering design optimisation, Journal of Intelligent and Fuzzy Systems 17 (4) (2006) 353–365.

[17] C. Gafa, S. Coupland, A new recursive type-reduction procedure for general type-2 fuzzy sets, in: Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), 2011 IEEE Symposium on, 2011, pp. 44 –49.

[18] S. Greenfield, F. Chiclana, S. Coupland, R. John, The collapsing method of defuzzification for discretised interval type-2 fuzzy sets, Information Sciences 179 (13) (2009) 2055–2069, iSSN: 0020-0255.

[19] S. Greenfield, R. John, S. Coupland, A novel sampling method for type-2 defuzzification, in: Proceedings of UKCI 2005, London, 2005, pp. 120–127.

[20] F. Guely, R. La, P. Siarry, Fuzzy rule base learning through simulated annealing, Fuzzy Sets and Systems 105 (3) (1999) 353 – 363.

[21] H. Hagras, Type-2 flcs: A new generation of fuzzy controllers, Computational Intelligence Magazine, IEEE 2 (1) (2007) 30–43.

[22] H. Hamrawi, S. Coupland, R. John, A novel alpha-cut representation for type-2 fuzzy sets, in: FUZZ IEEE 2010 (WCCI 2010), IEEE, IEEE, Barcelona, Spain, 2010, pp. 1 – 8.

[23] L. Ingber, Simulated annealing: Practice versus theory, Mathematical and computer modelling 18 (11) (1993) 29–57.

[24] W. Jeng, C. Yeh, S. Lee, General type-2 fuzzy neural network with hybrid learning for function approximation, in: Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on, IEEE, 2009, pp. 1534–1539.

[25] R. John, Perception modelling using type-2 fuzzy sets / r. i. john., Ph.D. thesis, De Montfort University (2000).

[26] R. John, S. Coupland, Extensions to type-1 fuzzy: type-2 fuzzy logic and uncertainty, Computational Intelligence: Principles and Practice (2006) 89–102.

[27] R. John, S. Coupland, Type-2 fuzzy logic: A historical view, Computational Intelligence Magazine, IEEE 2 (2007) 57 –62.

[28] R. John, C. Czarnecki, A type 2 adaptive fuzzy inferencing system, in: Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on, vol. 2, IEEE, 1998, pp. 2068–2073.

[29] N. Karnik, J. Mendel, Operations on type-2 fuzzy sets, Fuzzy Sets and Systems 122 (2) (2001) 327–348.

[30] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, 1983, Science 220 (1983) 671–680.

[31] O. Linda, M. Manic, Importance sampling based defuzzification for general type-2 fuzzy sets, in: Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, 2010, pp. 1 –7.

[32] F. Liu, An efficient centroid type-reduction strategy for general type-2 fuzzy logic system, Information Sciences 178 (9) (2008) 2224–2236.

[33] L. Lucas, T. Centeno, M. Delgado, General type-2 fuzzy inference systems: Analysis, design and computational aspects, in: Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International, 2007, pp. 1 –6.

[34] M. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.

[35] P. Melin, C. Gonzalez, J. Castro, O. Mendoza, O. Castillo, Edge-detection method for image processing based on generalized type-2 fuzzy logic, Fuzzy Systems, IEEE Transactions on 22 (6) (2014) 1515–1525.

[36] J. Mendel, Uncertain rule-based fuzzy logic systems: introduction and new directions, Prentice Hall, 2001.

[37] J. Mendel, Fuzzy sets for words: a new beginning, in: Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on, vol. 1, 2003.

[38] J. Mendel, R. John, Type-2 fuzzy sets made simple, Fuzzy Systems, IEEE Transactions on 10 (2) (2002) 117 –127.

[39] J. Mendel, R. John, F. Liu, Interval type-2 fuzzy logic systems made simple, Fuzzy Systems, IEEE Transactions on 14 (6) (2006) 808 –821.

[40] J. Mendel, F. Liu, On new quasi-type-2 fuzzy logic systems, in: Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on, IEEE, 2008, pp. 354–360.

[41] J. Mendel, F. Liu, D. Zhai, Alpha plane representation for type-2 fuzzy sets: Theory and applications, Fuzzy Systems, IEEE Transactions on 17 (5) (2009) 1189 –1207.

[42] P. Salamon, P. Sibani, R. Frost, Facts, conjectures, and improvements for simulated annealing, Society for Industrial Mathematics, 2002.

[43] J. T. Starczewski, Efficient triangular type-2 fuzzy logic systems, International Journal of Approximate Reasoning 50 (5) (2009) 799–811.

[44] C. Wagner, H. Hagras, zslices based general type-2 flc for the control of autonomous mobile robots in real world environments, in: Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on, 2009, pp. 718 –725.

[45] C. Wagner, H. Hagras, Toward general type-2 fuzzy logic systems based on zslices, Fuzzy Systems, IEEE Transactions on 18 (4) (2010) 637 –660.

[46] C. Wagner, H. Hagras, Uncertainty and type-2 fuzzy sets and systems, in: Computational Intelligence (UKCI), 2010 UK Workshop on, 2010, pp. 1 –5.

[47] C. Wagner, S. Miller, J. Garibaldi, D. Anderson, T. Havens, From interval-valued data to general type-2 fuzzy sets, Fuzzy Systems, IEEE Transactions on 23 (2) (2015) 248–269.

[48] S. White, Concepts of scale in simulated annealing, in: American Institute of Physics Conference Series, vol. 122, 1984, pp. 261–270.

[49] T. Yanar, Z. Akyrek, Fuzzy model tuning using simulated annealing, Expert Systems with Applications 38 (7) (2011) 8159–8169.

[50] L. Zadeh, The concept of a linguistic variable and its application to approximate reasoning. i, Inform. Sciences 8 (1975) 199–249.