Models-based Optimization Methods for the Specification of Fuzzy Inference Systems in Discrete EVent Simulation

P.-A. Bisgambiglia[1] B. Poggi[1] C. Nicolai[1]

[1]University of Corsica, CNRS UMR SPE 6134 Quartier Grossetti, Bat 018, FST 20250 CORTE

Abstract

Fuzzy Inference Systems (FIS) have the advantage of relying on the properties of Fuzzy Logic to represent imperfect information so gradually, and manipulate them from a linguistic description. This flexibility of representation is more significant for the study of complex systems. Our aims are to propose a formal approach for describing FIS as a Discrete Event System (DES), and to extend a DES in order to use the many advantages offered by FIS: flexibility, easy implementation, robustness... In this paper, we present the extension of Discrete EVent system Specification (DEVS) formalism to represent FIS, and we propose a modular approach (DEVFIS) to use several optimization methods. We focus mainly on the used new aproach about using genetic algoritm in order to optimize the FIS.

Keywords: DEVS, Fuzzy Sets Theory, FIS, optimization models, iDEVS, DEVFIS

1. Introduction

In the fields of decision support, the amount of knowledge to be considered and their qualities are very important. This information comes from field observation or measurement instrument (sensors) which are unreliable, or subject to human interpretation. In these fields the methods of computational intelligence, as fuzzy modeling, are adapted. They can represent and manipulate imperfect knowledge. For this, we are working to combine these two modeling approaches, approximate methods (fuzzy) and systemic methods (DES). Works on discrete event systems (DES) and fuzzy discrete event systems (FDES) has allowed making major advances in the fields of modeling and simulation [1, 2]. In the 70s Zeigler [3] was introduced DEVS (Discrete EVent system Specification) as an abstract formalism for discrete event modeling. DEVS is a universal formalism for discrete event system, which can be used to specify complex systems described from behavioral functions, states and inputs/outputs. The original implementation of the DEVS formalism considers systems as perfectly known, but in the case of complex systems, such as natural systems, knowledge is often imperfect. To consider this issue, we looked at several computational intelligence methods, as fuzzy logic [4], genetic algorithms and neural networks. Moreover, these methods are proven to be useful to take into account imperfect knowledge. For example, fuzzy logic control [5, 6] provides a human-like decision making methodology, which has been used widely in the field of control process.

Knowledge representation and the inclusion of imperfect knowledge is, in our view, a key step in an effort to optimally incorporate computational intelligence methods in a modeling and simulation framework [7]. We define imperfect knowledge as: uncertain, there is a doubt about their validity; imprecise, it is difficult to quantify, inconsistent: there is inconsistencies in the data. The fuzzy theories include: the Fuzzy Sets Theory [8], which extends the sets theory and allow manipulating imprecise knowledge; the possibility theory [9], which extends the probabilities theory for unclear and imprecise systems, and can handle uncertainty and fuzzy logic [4], which extends the binary logic and allows exploiting incomplete data. These different theories all aim to provide concepts, techniques, and formally rigorous methods to collect, represent and process fuzzy knowledge and data. In recent years our team is interested in computational intelligence methods and fuzzy theories, which take into account imprecision and uncertainty: Theory of evidence [10], Possibility Theory [9], Fuzzy Logic [4], etc. We have defined two new data types to represent such knowledge then, we have proposed a new implementation to integrate our data types in the DEVS formalism. The first step of this approach was the definition of a modeling method for representing and simulating imprecisions. This method, called iDEVS [11], associates the DEVS formalism [3] and Fuzzy Set Theory [8]. On this basis, we then introduced a new DEVS extension for describing FIS as DEVS models [12]. FIS are one of the most used applications of Fuzzy Logic and Fuzzy Sets Theory. They can be helpful to achieve classification tasks, data mining, process simulation, diagnosis, decision support, and process control. They have the advantage of relying on the properties of Fuzzy Logic to represent imperfect information so gradually, and manipulate them from a linguistic description, as would a human expert. This flexibility of representation is more significant for the study of complex systems. FIS can describe sys-

tems with partially known parameters, and provide a digital framework for representing and manipulating gradual information express by a human expert. The FIS is composed of three parts (c.f. 1): (1-fuzzification part) inputs represent a fuzzy partitions (several fuzzy sets), they describe the set of values that the system can take; (2-inference part) the inference engine consists of rules describing the behavior of the system and an operator (AND, OR, MAX, MIN); (3-defuzzification part) outputs can be described by a set of discrete values or fuzzy partitions.

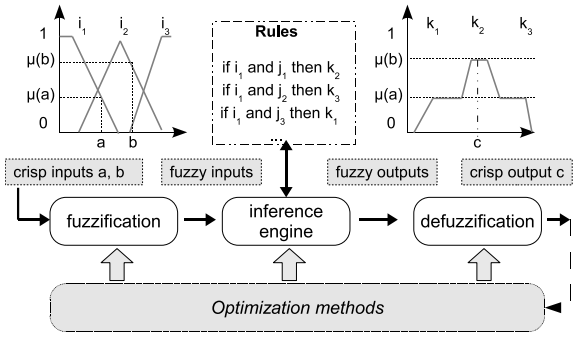

Figure 1: Structure of optimize Fuzzy Inference System

Globally there are many research works proposing an implementation of FIS [13, 14, 15]. Despite the wealth and quality of literature in the field, none of these works fully meet our expectations and goals. In the specific context of the DEVS formalism, the literature is much less abundant [16], and the proposed approach seems to us incomplete, and limited to one field of application: process control. We wish to propose a modular and generic modeling approach based on association of DEVS formalism and the Fuzzy Sets Theory to represent the FIS and make the event driven simulation. Everything must be grouped in a multi-modeling framework, open source, and features a graphical user interface (GUI) that allows to link experts and computer scientists. Indeed, we propose to integrate different optimization methods in DEVS models, they can be coupled to the structure of the FIS to optimize its parameters. The standard structure of a FIS: input; inference engine; output; can be completed, for its optimization, by other methods such as: the stochastic gradient, genetic algorithms, neural networks, etc.

To achieve our objectives, we followed several steps: (1) we worked on taking into account imprecision. We defined a first extension of the formalism DEVS (iDEVS [11]). It is based on fuzzy sets theory to represent imprecision: as a fuzzy interval form (class FuzzyInt); then, we have proposed a theoretical method for taking into account the uncertainties, method based on possibilities theory; and finally, we proposed an implementation of FIS as DEVS model (DEVFIS [12]). Those steps have led us to work on optimization methods and to propose their integration in the DEVFIS extension.

In the first part, we will offer a brief overview of the basic tools of our work: DEVS formalism, DEVSimPy: multi-modeling framework developed in our laboratory. In the second section, we describe: the iDEVS method; the DEVFIS approach, then we will present the addition of optimization methods before concluding with a pedagogical example. Finally, before concluding our research perspectives in the fields of modeling and simulation and computational intelligence, we will discuss the goals and limitations of our methods.

2. Discrete Event Modeling

Among the different methods developed and inspired by the System and Control Theory, we have chosen to work on the Discrete Event system Specification formalism [3]. The word discrete does not mean that time is discrete, or that state is discrete because state variables may assume continuous values. This word refers to the fact that the system evolution is based on events; these events may possibly have a continuous evolution but most importantly, an end event can cause new beginnings.

2.1. DEVS formalism

The DEVS formalism introduced by Zeigler [3] has been developed and upheld for over thirty years by an International community of researchers [17, 16, 7]. The work undertaken is part of the effort to develop an approach which will facilitate the modeling, simulation and validation phases of the study of complex systems. This approach is based on the development of a software architecture enabling us, on the one hand to use the same multi-modeling environment to analyze different systems and fields and on the other hand to implement generic simulation techniques in order to simulate the corresponding models.

The DEVS formalism can be defined as a universal and general methodology which provides tools to model and simulate systems. It is based on a mathematical paradigm with well-defined concepts of coupling of components, hierarchical, modular model construction, support for discrete event approximation of continuous systems and support of object-oriented paradigms.

The DEVS formalism is based on the definition of two types of components: atomic models and coupled models. The atomic model provides an autonomous description of the behavior of the system, defined by states, input/output functions and transition functions. The coupled model is a composition of atomic models and/or coupled models. It is modular and presents a hierarchical structure which enables the creation of complex models from basic models.

$$Atomic\ Model :< X, Y, S, t_a, \delta_{ext}, \lambda, \delta_{int} > \quad (1)$$

Where: X: is the set of input events, event is characterized by a couple (port, time, value), where the port means the input on which the event occurs, the time is the date of occurrence of the event, it is blank for internal events, and the value symbolizes the data from the event; Y: is the set of output events; S: is the set of partial or sequential states, which includes the state variables; $t_a : S \to \mathrm{T}^{\infty}$: is the time advance function which is used to determine the lifespan of a state; $\delta_{ext} : Q \times X \to S$: is the external transition function which defines how an input event X changes a state of the system, where $Q = \{(s, t_e) | s \in S, t_e \in (\mathrm{T} \cap [0, t_a(s)])\}$ is the set of total states, and t_e is the elapsed time since the last event; $\lambda : S \to Y^{\iota}$: is the output function where $Y^{\iota} = Y \cup \{\iota\}$ and $\iota \notin Y$ is a silent event or an unobserved event. This function defines how a state of the system generates an output event, when the elapsed time reaches to the lifetime of the state; $\delta_{int} : S \to S$: is the internal transition function which defines how a state of the system changes internally, when the elapsed time reaches to the lifetime of the state.

In a framework based on DEVS formalism, the user only has to worry about design of its model, the simulation algorithms are automatically generated. This property coupled with modular and hierarchical aspects of formalism are a very powerful tool for studying complex system of any type. However, depending on the studied system it is necessary to define specific models. One of the main advantages of DEVS formalism is its capacity for openness, so it can be easily extended to many fields of application. This led him to be described as multi-formalism: it brings together in a consistent manner, several methods or modeling formalisms. Our aim is to build on these properties to define a new modeling approach to take into account the imperfect knowledge. To do this, we want to integrate into the DEVS formalism various tools related to the computational intelligence.

2.2. DEVSimPy framework

The framework DEVSimPy allows a simple graphical interface to create and use DEVS models. It is a WxPython based environment for the simulation of complex systems. Its development is supported by the CNRS (National Center for Scientific Research) and the SPE research laboratory team.

The main goal of this framework is to facilitate the modeling of DEVS systems using the GUI library and the drag and drop approach. The interface is designed to help the implementation of DEVS model in form of blocks. The modeling approach of DEVSimPy is based on UML Software, and there is a separating between the GUI part and the implementation part of DEVS formalism.

With DEVSimPy we can: (1) describe a DEVS model and save or export it into a library; (2) edit the code of DEVS model to modify behavior's also during the simulation; (3) import existing library of models which allows the specific domain modeling (Power System, Fuzzy, etc.); (4) automatically simulate the system and perform its analysis during the simulation; etc.

Modeling and simulation have emerged as essential tools for analyzing complex systems behavior. Several methods have been proposed to improve the process of analyzing the behavior of these systems. Those proposals attempt to achieve more realistic models, relatively simple and highly flexible. Our interest focuses on DEVS is formalism for modeling discrete event systems. Efforts were taken to adapt this formalism to different areas and situations [3, 17, 16, 7] In this large niche, we look to the problem of fuzzy parameter systems.

3. Proposal and methodology

As we show: DEVS formalism because of its flexibility and scalability is a very interesting tool for studying complex systems and even fuzzy systems, the FIS is one of the methods of fuzzy logic most used and robust, it helps to describe systems like a human expert. The combination these two approaches seem highly relevant in order to offer an approach to modeling flexible and generic to take account of imperfect knowledge.

3.1. iDEVS method

Initially, we developed a method to associate the DEVS formalism and the fuzzy sets theory, this method is called iDEVS for inaccuracy DEVS [11]. The iDEVS method meets all constraints of formalism, and is applicable in many fields. iDEVS is based on the same properties of modeling and simulation that the DEVS formalism. It extends its application fields to take into account imprecise data. The iDEVS method is based on the fuzzy sets theory for the representation and the manipulation of fuzzy quantities.

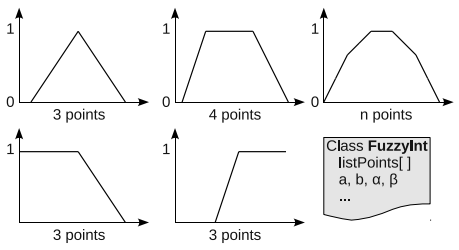

Figure 2: Data types represented by the FuzzyInt class

To make the link between the DEVS formalism and the fuzzy sets theory, we created a class,

called FuzzyInt to build objects representing imprecise variables, figure 2. This class is based on the definition of a fuzzy set, and was subsequently incorporated into the DEVS formalism to give birth to iDEVS.

It is possible to instantiate the class from a list of points, or characteristic points of the interval a, b the kernel $\mu(a) = \mu(b) = 1$, ψ the left boundary, α the radius $\alpha = a - \psi$ and ω the right boundary, β the radius $\beta = \omega - b$. We also added the possibility to describe the interval from a linguistic variable. A fuzzy interval is a simplified representation of fuzzy set, to describe denumerable quantities. In this class we have implemented several mathematical operators from the fuzzy sets theory and interval arithmetic. These operators allow manipulating FuzzyInt objects as classic object of Integer or Float type.

The next step is the creation of a new method to allow classification tasks, data mining, process simulation, diagnosis, decision support, and process control, by fuzzy inference. It must associate the DEVS formalism and a part of fuzzy logic called Fuzzy Inference System (FIS). This new method, called DEVFIS for Discrete EVent Fuzzy Inference system Specification, must take into account the incomplete knowledge.

3.2. DEVFIS method

DEVFIS method is an extension of DEVS to represent FIS. The interest of this approach are multiple: to extend the DEVS formalism to other application fields, e.g. control, learning, decision support; continue our work on the association of DEVS formalism and fuzzy logic; coupled DEVS with computational intelligence methods; develop and propose new DEVS models, and thus enhance the quantity and diversity in our framework (DEVSimPy); provide users (experts or specialists) method of modeling simple and intuitive, and which their expectations.

The theoretical approach and a first application to define real-time coefficient of fires spreading was proposed in [7]. The first part of its implementation was presented in [12]. In this paper, we present the addition of optimization methods. The parameters optimization of FIS is an important step in view of their use. FIS are generally based on human expertise, often qualitative, expressed in natural language and therefore subject to widely varying interpretations. The optimization methods allow a passage from the qualitative to quantitative, so it is possible to refine the knowledge and fully exploit the system. Figure 1 shows the structure of an FIS with the addition of an optimization method. This method will receive an evaluation criterion and will run to optimize the structure or parameters of the FIS. The different optimization methods can be grouped into three categories. (1) Analytical methods: they are based on gradient descent or stochastic gradient for learning. The typical use is the optimization of the conclusions of the rules [18]. (2) Evolutionary methods: these methods include the genetic algorithms and evolutionary strategies [19]. They allow the optimization of the structure and parameters of the FIS. Evolutionary methods have therefore a wider scope than analytical methods, and are particularly interesting for structural optimization. However, they may be slower and less accurate. (3) Classification methods can be used for structural optimization. Two methods are widely used for the classification, method of Kohonen [20] and that of Fuzzy C-Mean, FCM [21].

There are many optimization methods, we will present, through an approach based models, the most used. For optimization, we will use the stochastic gradient methods, and for learning or optimization we will use the genetic algorithms or neural networks. Stochastic gradient is an optimization method for minimizing an objective function that is written as a sum of differentiable functions. The genetic algorithms are used to solve different type of problems. The resolution is based on theory of evolution: natural selection, reproduction, mutations.

3.2.1. Methodology

DEVFIS method is based on the integration of fuzzy inference techniques in the DEVS formalism. The DEVFIS method is independent of iDEVS method; it is based on the same mechanism, the integration of a class: FuzzySets (FuzzyInt list) with DEVS framework. This new method is intended to extend the application fields of the DEVS formalism in order to make learning, optimization and control.They operate from fuzzy reasoning rules, which have the advantage of managing the progressive phenomena. As for iDEVS method, a new class was defined, FuzzySets class; it contains a list of objects of FuzzyInt type and different methods for fuzzy sets handling. For example, it is possible to describe fuzzy partitions, and several defuzzification methods are implemented.

Figure 3 presents an input and output fuzzy partition. In the class, there are several functions, e.g. search a linguistic variable associated with an input, or execute a defuzzification method.

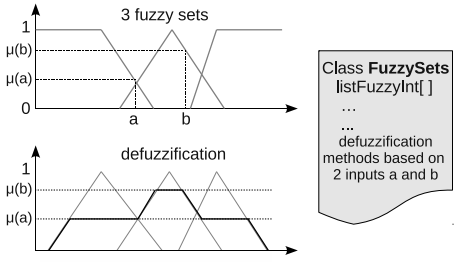

Figure 3: Data described by the FuzzySets class, fuzzy partitions

Starting of this class we can represent the inputs and outputs of the system (fuzzy partition), and the results obtained following the application of fuzzy operators or inferences methods (figure 3). We propose accordance with the structure of a fuzzy inference system (figure 1), to define a model DEVFIS for each stage of the process of inference, namely:

- one or more atomic models based on a fuzzification method to represent the input of the system;
- a coupled model describing the inference engine, which includes an atomic model that represents all the rules describing the system, an atomic model describing the fuzzy operators employees, and an atomic model that represents the inference method;
- one or more atomic models based on a defuzzification method to represent the output of the system;

The models are described using the following properties, figure 4.

Example: the atomic model for input is given $M_I =< X, Y, S, s_0, t_a, \delta_{ext}, \delta_{int}, \lambda >$ such that:

$X = receive(crisp\_value)$;

$Y = send(membership\_degree, linguistic\_label)$;

$S = \{(fp, md, ll, \sigma)$ with $fp : fuzzy\_partition$, $md : membership\_degree$, $ll : linguistic\_label$, $\sigma \in \mathrm{T}^{\infty}\}$ and $s_0 = \{FuzzyInt(list), 0, '', \infty\}$;

$t_a(s) = \sigma \, \forall \, s \in S$;

$\delta_{ext}(((fp, md, ll, \sigma), t_e), receive(crisp\_value))$:

- $find\ if\ crisp\_value\ is\ in\ fp$
- $S = (fp, fp.\mu(crisp\_value), fp.label(crisp\_value), 0)$

$\delta_{int}(fp, md, ll, \sigma) = (fp, md, ll, \infty)$;

$\lambda(fp, md, ll, \sigma) = send(md, ll)$;

The general problem is to consider the best way to put an inference system in DEVS models, to choose the best inference algorithms to use, and to represent the sets of inputs and outputs. For this it is possible to add to the structure of FIS optimization methods. The optimization methods can occur at several levels, their goal is to optimize the FIS parameters, so they can be fully parametrized. In the case of simple FIS, it is possible to parametrize and optimization by successive trials (by hand), but in the case of complex FIS, this solution is too costly in time, you need to use methods of optimization and learning.

3.2.2. Optimization methods

FIS are based on human expertise, often qualitative, expressed in natural language, it is their strong point. In the first part of our work [7, 12], we relied on that knowledge to define the FIS parameters. But this knowledge is subject to many interpretations as there are respondents. We therefore wish to add optimization methods to supplement and improve the setting of FIS used. This action will help to refine knowledge and is learning. There are several types of learning: supervised, weakly supervised, unsupervised, and learning with criteria. We will work on learning with the criteria in this case the qualities of the FIS are evaluated by a criterion that rewards or punishes according to the achieved objectives.

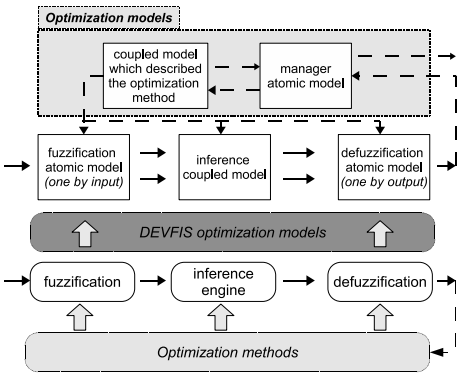

Figure 4: Optimization models to DEVFIS

In general, it is possible to apply optimization methods that if there are data to optimize. A first step is to choose the FIS part to optimize: structure, rules, or partitions. We must then define the optimization constraints to ensure consistent results. For example, for optimization of fuzzy partitions of inputs and outputs, we have defined as constraints: (1) optimized partitions must be strong fuzzy partition, (2) the order of fuzzy sets should be stored; (3) and the degree of overlapping of the membership functions (overlap between functions) should be neither too large nor too small.

Once part of the FIS to optimize and the chosen constraints, we can apply an optimization method. They are grouped into 3 categories: analytical methods (stochastic gradient), evolutionist methods (genetic algorithms) and classification methods. In our case, each of these methods has been integrated in a DEVS coupled model. Usually it involves running an algorithm. Figure 4 shows the general structure of our optimization model. There is a coupled model by method; it specifies the execution of the optimization algorithm. It is coupled to two models, the model to optimize (fuzzification, rules, defuzzification) and an atomic model that is used as a switch. The manager or switch model must be connected to the model that computes the evaluation criteria in order to retrieve it. Its purpose is making the system break during the optimization phase. During the optimization phase, this model manages and directs the messages; it sends the evaluation criterion to the optimization model. During

Algorithm 1 Evolutionary methods algorithm

1: Choose an initial individual FS_i
2: Initialize the population by mutation of FS_i
3: Calculate the performance of each individual
4: Stop = false, iteration = i
5: Do {
6: Select the new generation (choose the n best and delete the n worst)
7: Apply the cross function
8: Apply the mutation function
9: Calculate the performance of the new population
10: If the performance==evaluation criterion or iteration==0
11: { Stop = true}
12: iteration–
13: } while stop == false

the classical phase, it cuts the communication with the optimization model and forwards the messages to the following model.

Both models can be grouped in a coupled model of the highest level. We must think about adding an input port to the part of the FIS to be optimized, its aim is to replace the initial parameters for the optimized parameters. All these models come complete our library [7].

The stochastic gradient method is very interesting to optimize the conclusion of the rules, but is not very suitable for optimizing fuzzy partitions.

Classification methods must be adapted to reflect the specific aspect of FIS. They are generally used in a supervised or unsupervised context. Among the classification methods, we used a hybrid model: neuro-fuzzy. The interest of these systems is that the Neural natworks (NN) and the FIS are complementary. NN are used for their learning abilities and FIS for their readability and flexibility. There are several methods of combinations: the merger of FIS and NN, that is to say, the FIS is set as an NN; NN are used to replace a FIS components; in series variables entry of FIS to determine the output of the NN; and in parallel is the case the most effective because it will provide a knowledge representation as large as possible and a rule base effectively.

The evolutionary methods manipulate data, which are the encoded representation of an object. This object will be one of the parameters to optimize FIS: partition or rule. For example, a fuzzy partition representing an individual in a population and a part of this partition (fuzzy set or membership function) a gene. The goal of evolutionary methods is, over generations, to find the best individual for a given criterion. Evolutionary methods are implemented: (1) a population of N individuals; (2) an evaluation function of individuals (perf(i,test)); (3) operators of mutation and crossover to create new individuals; (4) a policy of population replacement; (5) a stopping criterion. We will use the following algorithm:

3.2.3. Example for an output partition:

The system we will study is a controller for heating. The System is heating, a sensor is used to return the room temperature; the Data module is the set point temperature.

FIS takes as input an error: the set point temperature sent by the Data module, minus the room temperature sent by the System. Input partition is defined as follows: negative, zero, positive. On output, it returns the thermostat (0,1, ... 8).

The rules are:

If negative then we lower the temperature (C)

If zero then we leave unchanged (B)

If positive then we increase the temperature (A)

Consistent with our approach, the system is described from several DEVS models. The methodology described above and presented in the paper [12]. We used two coupled models (figure 4). A coupled model encapsulates the optimization method. It contains the switch, that is to say, the manager model, and a model that contains the optimization algorithms. A second coupled model describes the FIS.

3.2.4. Application of genetic algorithms:

1. The definition of the genetic is equivalent to select the FIS part to optimize, us an output partition FS: {A=[], B=[], C=[]}.

- The discourses universe (all abscissa values) = [0; 8]
- Linguistic variables A = 'Heat', B = 'None', C = 'Lower'

2. From this information, we can create an initial individual who is a suitable solution FS_i: {A=[0, 2, 4], B=[2, 4, 6], C=[4, 6, 8]}. Step 1 of the algorithm 1.

3. We will then automatically generate the population (N = 60 individuals) by applying a function of mutation (mutation(min_disturbance, max_disruption)), this function was added in the FuzzySets class. As shown in figure 5, from the initial individual, we will modify the bounds of fuzzy sets (ψ, a, b, ω) from a coefficient or a radius randomly determined (us between 0.0 and 2.0).

Disturbance example: $FS_i^{n=1}$={A=[0.1, 1.0, 5.4], B=[0.4, 3.0, 6.8], C=[3.1, 7.1, 9.4]}. The constraints that we presented here are applied to select the new population. A new individual must meet the following constraints:

(1) A, B and C must be a strong fuzzy partitions and standardized $\alpha < a < b < \beta$;

(2) A <B <C;

(3) A, B and C must overlap;

(4) For A, B and C, $\alpha \geq 0$ and $\beta \leq 8$.

All partitions that do not meet these constraints are rejected. Step 2 of the algorithm.

4. The rest of the population, in our example 29 of the 60 individuals generated, are then evaluated

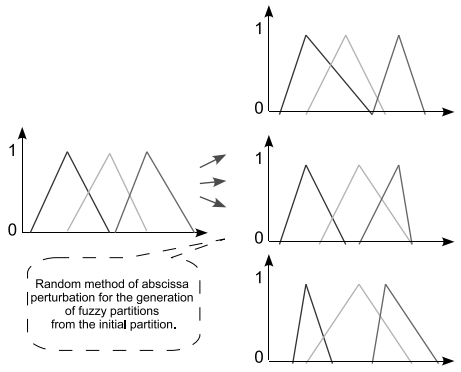

Figure 5: Example of individuals random generation

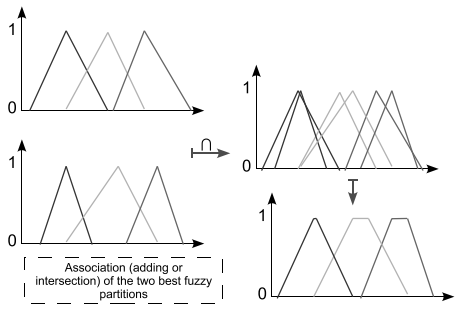

Figure 6: Crossover example between two fuzzy partitions

based on a performance criterion defined in the optimization coupled model. In our case we wanted to get the fastest possible set point temperature. The evaluation function sets a score between 0.0 and 1 to each individual. After evaluation, we keep the two best. Step 3 of the algorithm. The two individuals are crossed to give birth to a son. To implement this crossing, we use intersection operator between fuzzy sets, c.f. figure 6 then, we apply a smoothing function to find a standardized form. The smoothing function used in this example, executes the intersection operator (max) between two fuzzy sets, then standardizes the modal valuesââ. From this son, a perturbation is applied to obtain two children must respect the constraints set. Steps 7 and 8 of the algorithm 1.

6. Finally, we evaluate the new population and if the stop condition is not satisfied the algorithm continues until the desired result.

7. After obtaining a good result or i iteration, i set by the user (we tested 20 iterations), the parameter is considered optimized. Finally, we obtain the following partition FS_f = {A=[0.1, 0.7, 3.6, 4.1], B=[3.7, 3.9, 4.5], C=[4.1, 4.7, 7.2, 8]}. We can note that the two fuzzy sets A and C were enlarged, while B is reduced. This phenomenon is due to our evaluation criterion. As we wanted to quickly reduce the error, the partition parts that have a direct impact on the error have increased. The fuzzy set B, which involves no change in reading rules, was reduced in favor of A and C.

3.3. Discussion and remarks

The stochastic gradient method was used to approximate an equation $y = x^2$ on the interval $[-1, 1]$. For this example, the results are quite good, but their presentation has limited interest in discrete event simulation. In [12] we presented the implementation of a temperature controller and pressure for a boiler. Without going into details, the application is already well detailed in the article, we found that the addition of optimization models based on genetic algorithms do not allow to accelerate the control process. Despite the use of optimization methods, obtaining good results is always subjected to empirical testing, we must test the various methods, choosing the right parameters to optimize, etc. Thus, our efforts, to make our approach fully generic, are in party remained unsuccessful. To properly optimize a FIS we must take into account, the specificity of the system and the FIS. However, we think it is very useful to have such models, to complement the DEVFIS approach, and to complement our fuzzy toolbox integrated into DEVSimPy. Our contribution must be completed, we still need: add, test, and evaluate the impact on the results of different operators and smoothing methods used.

Despite, there are many frameworks that provide tools to model FIS [13, 16, 22, 14, 15], and the quality of literature in this field. Some tools do not allow for simulation: FisPro, other do not make discrete simulation: MatLab. We believe that our approach has its place. We wish to develop a modular, intuitive and generic approach, based on the DEVS formalism to make the event driven simulation, and computational intelligence methods to provide an adaptive framework close to the mode of human reasoning. Our approach should be integrated in a multi-modeling framework, open source, and features a graphical user interfaces (GUI) to create a direct link between expert and computer scientist. In our view none of these tools has all these qualities.

4. Conclusion and perspectives

This work should be seen in a more general context, which aims to combine the DEVS formalism and computational intelligence methods: DEVS for approximate reasoning. Our objectives are to extend the DEVS formalism: proposing adaptive extensions; taking into account the complexity of systems; providing intelligent methods, flexible, and intuitive. We want to replace the expert at the center of the modeling process, and provide the necessary tools to describe their systems so natural.

In this paper, we presented part of our work, and especially a method to represent fuzzy inference systems (FIS) as DEVS models. This is the third part of our work on the DEVFIS extension. We presented a modular approach to add optimization methods to the standard structure of the FIS. On

the one hand, the FIS can represent systems, a flexible and robust way, their uses within the DEVS formalism has therefore many interests. On the other hand, optimization methods can facilitate the calibration or setting phases, moreover they are used to improve the structure of the FIS: inputs, outputs and rules.

Our approach is simple to use, after selecting the parameters to optimize, it is sufficient to couple, or add optimization models to the structure of a FIS. Several models are available; they are based on different optimization methods, stochastic gradient, genetic algorithms and neural network. Subsequently, other methods will add to enrich the optimization library. In the short term, we wish to propose two extensions to this paper: study the impact of operators and smoothing functions, and a comparison between the several models of optimization. This work enables us to complete our idea of fuzzy toolbox for the DEVS formalism [7]. Subsequently, we will continue our research in the fields of computational intelligence, for example, we work to define an extension of the DEVS formalism for representing Bayesian networks.

References

[1] C.G. Cassandrass and S. Lafortune. Introduction to Discrete Event Systems. 1999.

[2] Feng Lin and Hao Ying. Fuzzy discrete event systems and their observability. In IEEE, editor, Proceedigns of IFSA World Congress and 20th NAFIPS International Conference, Vancouver, (Canada), volume 3, pages 1271 – 1276, 2001.

[3] Bernard P. Zeigler. Theory of Modeling and Simulation. Academic Press, 1976.

[4] Lofti A. Zadeh. Fuzzy logic. Computer, 21(4):83–93, 1988.

[5] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies, (7):1–13, 1975.

[6] T. Takagi and M. Sugeno. Fuzzy indentification of systems and its applications to modeling and control. In IEEE, editor, proceedings in IEEE Trans On S.M.C, 01 1985.

[7] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.-F. Santucci. Fuzz-idevs: Towards a fuzzy toolbox for discrete event systems. In ACM, editor, Proceedings of the SIMUTools'09, Rome (Italie). SIMUTools, 3 2009.

[8] L.A. Zadeh. Fuzzy sets. Information Control, 8:338–353, 1965.

[9] L.A. Zadeh. Fuzzy sets as a basics for a theroy of possibility. Fuzzy Sets and Systems, pages 3–28, 1978.

[10] G. Shafer. A Mathematical Theory of Evidence. 1976.

[11] P.-A. Bisgambiglia, E. de Gentili, P.A. Bisgambiglia, and J.-F. Santucci. idevs: new method to study inaccurate systems. In IEEE, editor, Proceedings of the The Annual IEEE International Conference on Fuzzy Systems, Jeju Island (Korea). IEEE, 8 2009.

[12] P.-A. Bisgambiglia, L. Capocchi, P.A. Bisgambiglia, and S. Garredu. Fuzzy Inference Models For Discrete EVent Systems. In IEEE, editor, Proceedings of the 2010 IEEE World Congress on Computational Intelligence (WCCI 2010) - IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Barcelona (SPAIN). IEEE, 7 2010.

[13] A. Abraham. Evonf: a framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation. In IEEE, editor, Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium, number 2, pages 327–332, 02 2002.

[14] Ying-Shen Juang, Shui-Shun Lin, and Hsing-Pei Kao. Design and implementation of a fuzzy inference system for supporting customer requirements. Expert Systems with Applications, 32(3):868 – 878, 2007.

[15] Lee ChangSu Lee, A. Zaknich, and T. Braunl. A framework of adaptive t-s type rough-fuzzy inference systems (arfis). In Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008) - IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pages 567–574, 06 2008.

[16] S. Sheikh-Bahaei, J. Liu, M. Jamshidi, and P. Lino. An intelligent discrete event approach to modeling, simulation and control of autonomous agents. 10(4):337 – 348, 2004.

[17] Y. Kwon, H. Park, S. Jung, and T. Kim. Fuzzy-DEVS Formalisme : Concepts, Realization and Application. Proceedings AIS 1996, pages 227–234, 1996.

[18] P.Y. Glorennec. Fuzzy Q-Learning and Dynamical Fuzzy Q-Learning. In IEEE, editor, Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Orlando, USA, page 474, 1994.

[19] T. Presberger and M. Koch. Comparison of evolutionary strategies and genetic algorithms for optimisation of a fuzzy controller. In Proceedings of EUFIT 95 Aachen, Germany, 1995.

[20] T. Kohonen. The self-organizing map. 78:1464 – 1480, 1990.

[21] M. Sugeno and Takahiro Yasukawa. A fuzzy-logic-based approach to qualitative modeling. 1(1):1464 – 1480, 1993.

[22] E. Kilic, C. Karasu, and K. Leblebicioglu. Fault diagnosis with dynamic fuzzy discrete event system approach. 3949:117 – 124, 2005.