Self-evolving Parameter-free Rule-based Controller

Pouria Sadeghi-Tehran*, Ana Belén Cara†, Plamen Angelov*, Héctor Pomares†, Ignacio Rojas† and Alberto Prieto†
* School of Computing and Communications
Infolab21, Lancaster University, United Kingdom
Email: p.sadeghi-tehran@lancaster.ac.uk, p.angelov@lancaster.ac.uk
†Department of Computer Architecture and Computer Technology
CITIC-UGR, University of Granada, Spain
Email: acara@atc.ugr.es, hector@ugr.es, irojas@atc.ugr.es

***Abstract*—In this paper, a new approach for Self-evolving PArameter-free fuzzy Rule-based Controller (SPARC) is proposed. Two illustrative examples are provided aiming a proof of concept. The proposed controller can start with no pre-defined fuzzy rules, and does not need to pre-define the range of the output or control variables. This SPARC learns autonomously from its own actions while performing the control of the plant. It does not use any parameters, explicit membership functions, any off-line pre-training nor the explicit model (e.g. in a form of differential equations) of the plant. It combines the relative older concept of indirect adaptive control with the newer concepts of (self-)evolving fuzzy rule-based systems (and controllers, in particular) and with the very recent concept of parameter-free, data cloud and data density based fuzzy rule based systems (and controllers in particular). It has been demonstrated that a fully autonomously and in an unsupervised manner (based only on the data density and selecting representative prototypes/focal points from the control hyper-surface acting as a data space) it is possible generate a parameter-free control structure and evolve it in on-line mode. Moreover, the results demonstrate that this autonomous controller is effective (has comparative error and performance characteristics) to other known controllers, including self-learning ones, but surpasses them with its flexibility and extremely lean structure (small number of prototypes/focal points which serve as seeds to form parameter-free and membership function-free fuzzy rules based on them). The illustrative examples aim primarily proof of concept.**

I. Introduction

During the last decades, fuzzy logic and fuzzy control have emerged as one of the most successful and commonly used areas of research on the foundation of fuzzy set theory. Although fuzzy controllers and fuzzy logic concept, in general, had many opponents at the beginning, these days the theory has been widely accepted. The importance of using such technique becomes clear when the model of the system is unknown or implementing classic analytical methods is not possible for the design of a controller. In reality, all possible events or the frequency of their occurrences cannot be identified while modelling a system. Lack of this knowledge imposes us to use an approximate model of a system.

There are many useful tools in control system theories for approximate modelling of systems as well as design of analytically founded control algorithms. In general, the more accurate modelling of a system, the better response of the controlled system can be achieved. However, the problem arises when the mathematical model of a system is not available; also, there are situations when systems contain unknown nonlinearities. Some of these difficulties can be addressed by using adaptive control methods introduced by H. Butler [1], V. Chalam [2] and many other researchers [3]. Nevertheless, their basic mathematical characteristic is rather complex and computationally expensive due to a large number of computing iterations.

Due to the fact that a fuzzy logic algorithm has the characteristic of a universal approximator, it is possible to model systems containing unknown nonlinearities using a set of IF-THEN fuzzy rules. Nonetheless, fuzzy logic algorithms are not limited to the applications when the mathematical model of a system is unknown. Sometimes the mathematical model is available but it is too complex and the design of a controller for such a system is extremely time-consuming. Besides, the level of knowledge required to design a fuzzy controller usually is not as high as for traditional mathematical-based controllers.

The main challenges in designing conventional fuzzy controllers are that they are sometimes designed to work in certain modelling conditions [4]. Moreover, fuzzy controllers include at least two parameters per fuzzy set, which are usually predefined in advance and tuned off-line [5]. Many techniques have been presented for auto-tuning of the controller's parameters in batch mode [6] mostly using genetic algorithms [7], or neural networks [8], [9] off-line. From a practical point of view, however, there is no guarantee that pre-training parameters have a satisfactory performance in online applications when the environment or the object of the controller changes. To tackle this problem, several approaches have been proposed for online adaptation of fuzzy parameters [10], [11], [12].

Nevertheless, only a few approaches have been introduced for online adaptation of fuzzy controller structure when no prior knowledge of the system is available. Possibly, the first paper on evolving fuzzy rule-based controller was presented in 2001 [13]. The author presented a method to design fuzzy controllers with evolving structure. Fuzzy rules are created based on collected data online. It is based on a combination of unsupervised online clustering and a version of recursive weighted least squared estimation of the parameters of the consequent part [14]. The proposed approach is applied to online design of fuzzy controllers combined with the indirect learning techniques [15]. The advantage of this method com-

U.S. Government work not protected by U.S. copyright

pared to original works on indirect learning based control is that the evolving rule based model is implemented to design the structure of the controller online; therefore, there is no need the system to be trained in batch mode. Also, the proposed approach was successfully implemented in air-conditioning systems as an example of wide potential applications. Recently, Cara et. al [16], [17] proposed a new online self-organising fuzzy controller without using a prior knowledge about the differential equations of the plant or implement any offline pretraining. The proposed method starts from very simple (even empty) topologies and the structure of the fuzzy controller is modified online based on the data obtained during the system's operation. Two main phases were introduced for parameter learning of the controller's consequents and modifying the structure of the controller [16]. The proposed approach was successfully applied to a nonlinear servo system consisting of a DC motor and showed a satisfactory performance [17], [18]. The drawback of this approach is that adding new membership functions increases the number of rules exponentially.

In this paper, a new self-evolving fuzzy controller is proposed which is parameter free and fully autonomous. It is based on a newly introduced simplified type of fuzzy rule-based (FRB) systems [19], which can be seen as the next form of FRB system types after the two well-known systems named Mamdani and Takagi-Sugeno-Kang (TSK). The proposed method offers a new way of defining the antecedent part without the need to pre-define the membership functions in an explicit manner, but using focal points/prototypes (selected descriptive actual data points) instead. The fuzzy rules are formed around selected representative points from the control surface; thus, there is no need to define the membership functions per variable. It has a much simplified antecedent part which is formed using so-called data clouds. Data clouds are sets of data samples which have no specific shape, parameters, and boundaries. The proposed method uses the relative density and takes into account the distance to all previous data samples calculated recursively. The fuzzy membership of a data sample is associated to more than one data cloud with different degrees of membership determined by the local density to all samples from the data cloud. In order to show the effective performance of the proposed controller, it is applied to two applications and the results are provided.

The remainder of the paper is organised as follows. In section II the new simplified FRB system is introduced, including the rule representation and its associated inference process. The evolving methodology used for the online learning of both the structure and the parameters of SPARC is described in section III. First, we present the mechanism for the online adaptation of the consequents in section III-A and then, we illustrate the structure evolution process in section III-B. In section IV, two simulation examples are presented as a proof of concept for the proposed methodology. Finally, the conclusions are drawn in section V.

II. Structure of the data cloud-based controller

ANYA [19] is a newly proposed type of fuzzy rule-based system characterized by the use of non-parametric antecedents. Unlike traditional Mamdani and TSK fuzzy systems, ANYA does not require an explicit definition of fuzzy sets (and their corresponding membership functions) for each input variable. On the contrary, ANYA applies the concepts of data cloud and relative data density to define antecedents that represent exactly the real data density and distribution and that can be obtained online from data streams.

Data clouds are subsets of previous data samples with common properties (closeness in the observation data space). Contrary to traditional membership functions (MFs), they represent directly and exactly all the previous data samples. A given data can belong to all the data clouds with a different degree $\gamma \in [0, 1]$, thus the fuzziness in the model is preserved. It is important to highlight that data clouds are different from traditional clusters in that they do not have specific shapes and thereby do not require the definition of boundaries.

It is possible to use ANYA to design fuzzy controllers in situations in which the lack of knowledge about the plant makes it difficult to define the rule antecedents. Thus, this ANYA controller will have a rule base with N rules of the form

$$\Re^i : \text{IF } \left(\vec{x} \sim \aleph^i\right) \text{ THEN } \left(u^i\right) \tag{1}$$

where $\sim$ denotes the fuzzy membership expressed linguistically as "is associated with", $\aleph^i \in \mathbb{R}^n$ is the ith data cloud defined in the input space, $\vec{x} = [x_1, x_2, ..., x_n]^T$ is the controller's input vector and u^i is the control action defined by the ith rule. It has to be noted that the antecedent part of the rule is not formed by a conjunction (or any other combination) of premises of the form IF x_j is X_j^k, as in traditional fuzzy systems. This means that no aggregation operator is required to combine the premises. All the remaining components of the FRB system (e.g., the defuzzification method) can be selected as in a traditional fuzzy system.

A rule base of the form (1) can describe complex, generally non-linear, non-stationary, non-deterministic systems that can be only observed through their inputs and outputs. Hence, ANYA controllers are suited to describe dependences of the type "IF X THEN U" based on the history of pairs of data observations of the form $z_j = [\vec{x}_j^T; u_j]^T$ (with $j = 1, ..., k-1$ and $z \in \mathbb{R}^{n+1}$), and the current kth input $\vec{x}_k^T$.

The degree of membership of the data sample $\vec{x}_k$ to the data cloud $\aleph^i$ is measured by the normalized relative density as follows [19]:

$$\lambda_k^i = \frac{\gamma_k^i}{\sum_{j=1}^{N} \gamma_k^i} \quad i = 1, ..., N \tag{2}$$

where γ_k^i is the local density of the ith data cloud for that data sample. This local density is defined by a suitable kernel over the distance between $\vec{x}_k$ and all the other samples in the data

cloud, i.e.,

$$\gamma_k^i = K\left(\sum_{j=1}^{M^i} d_{kj}^i\right) \quad i = 1, ..., N \tag{3}$$

where d_{kj}^i denotes the distance between the data samples $\vec{x}_k$ and $\vec{x}_j$, and M^i is the number of input data samples associated with the data cloud $\aleph^i$. In this general form, we consider that a sample is associated with the data cloud with the highest local density. In addition, we use the Euclidean distance, i.e., $d_{kj}^i = ||x_k - x_j||^2$. Nonetheless, any other type of distance could also be used [19].

In this work we select the Cauchy kernel. Thereby, (3) can be rewritten as

$$\gamma_k^i = \frac{1}{1 + \frac{\sum_{j=1}^{M^i}\left(d_{kj}^i\right)^2}{M^i}} = \frac{M^i}{M^i + \sum_{j=1}^{M^i}\left(d_{kj}^i\right)^2} \tag{4}$$

and can be computed recursively [20] by

$$\gamma_k^i = \frac{M^i}{M^i\left(\vec{x}_k^T\vec{x}_k + 1\right) - 2\alpha_k^i + \beta_k^i} \tag{5}$$

where

$$\begin{aligned} &\alpha_k = \vec{x}_k^T\xi_k^i; \\ &\xi_k^i = \xi_{k-1}^i + \vec{x}_{k-1}; && \xi_0^i = 0 \\ &\beta_k^i = \beta_{k-1}^i + ||\vec{x}_{k-1}||^2; && \beta_0^i = 0 \end{aligned}$$

As for the defuzzification, it has to be noted that ANYA can work with both Mamdani and TSK consequents. In this case, we use the latter type, as it is usual in control applications [21], [16], [18], [22]. Hence, if we consider the weighted average for the defuzzification, the output of the ANYA controller for a given input $\vec{x}_k$ is

$$\hat{G}(\vec{x}_k) = u_k = \sum_{i=1}^{N}\lambda_k^i Q^i = \frac{\sum_{i=1}^{N}\gamma_k^i Q^i}{\sum_{i=1}^{N}\gamma_k^i} \tag{6}$$

where Q^i denotes the ith rule consequent.

III. Methodology for evolving the SPARC

In this section we present the methodology applied for online evolving the structure and parameters of SPARC. Fig. 1 depicts the flowchart of the evolution process. Initially, the controller is empty, so it has to be initialized from the first data sample received. After this, the same steps are repeated for every incoming data: Firstly, the consequents of the current rules are updated according to the error at the plant's output. Then, a new control action is generated by applying the inference process described in section II. Finally, the controller's structure is updated. If the appropriate conditions are satisfied, a new data cloud (and hence, a rule) is created; otherwise, the new sample is used to update the information about the data density and the characteristics of the controller's current configuration.

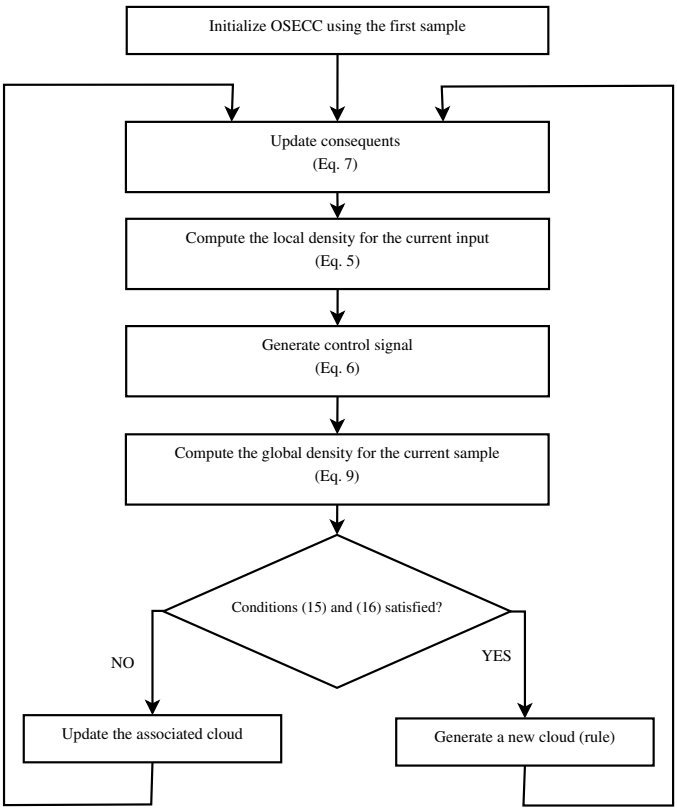

Fig. 1. Flowchart of the evolving methodology for the SPARC

In the following subsections, the entire process is described in more detail. Section III-A is devoted to the mechanism for the online adaptation of the rule consequents. In section III-B the process of adding a new data cloud is described.

A. Online adaptation of the rule consequents

From a "local" point of view, the goal of the controller is to bring the plant's output from its current value to the desired reference value as soon as possible, i.e., ideally $y_{k+1} = r_k$, where k and $k+1$ represent consecutive control steps.

Assuming that the plant is monotonic with respect to the control signal, then the partial derivative of the plant's output with respect to the control signal has a definite constant sign. Therefore, the combination of the error at the plant's output and the sign of the monotonicity of the plant w.r.t. the control signal, provides information about the right direction in which to move the rule consequents to achieve the local control objective [11].

Hence, we use this idea to adapt online the rule consequents, by applying the same method as in [22], [18], [16]. This adaptive method analyzes the error at the plant's output at each time step and applies a penalty or a reward to the rules that are responsible for the error. Since not all the rules contribute in the same degree to the plant's output value, the modification applied to each rule consequent is proportional to

their contributions. Thus, the following modification is applied to the consequent of the i-th fuzzy rule at instant k:

$$\Delta Q_k^i = C \cdot \lambda_{k-1}^i(\vec{x}) \cdot e_k(\vec{x}) = C \cdot \lambda_{k-1}^i(\vec{x}) \cdot (ref_{k-1} - y_k) \quad (7)$$

where $\lambda_{k-1}^i(\vec{x})$ is the activation degree of the i-th rule at the previous time step, when the rule was fired to obtain ref_{k-1}, and C is a normalization constant. C has the same sign as the monotonicity of the plant w.r.t. the control signal, and its absolute value is set offline as $|C| = \Delta u / \Delta ref$, where Δu is the range of variation of the control signal and Δref is the range of possible reference values. Both are known beforehand, as the user of a control system has to know what the operating ranges are.

Additionally, it has to be kept in mind that most real life controllers are limited in their operation and can only provide control actions within an specific range, namely, $[u_{min}, u_{max}]$. If the recommended control signal is $u_k^* > u_{max}$, the input finally applied to the plant will be u_{max} and $y_{k+1} < r_k$ will be output obtained. However, the rules should not be penalized, as they are already providing the best possible answer. The same reasoning is valid when $u_k^* < u_{min}$ and $y_{k+1} > r_k$. Hence, if the error is due to the actuator's limitations, the rules remain unchanged:

$$\Delta Q_k^i = \begin{cases} 0 & \text{if } u_{k-1} = u_{min} \;\&\; \Delta Q_i(k) < 0 \\ 0 & \text{if } u_{k-1} = u_{max} \;\&\; \Delta Q_i(k) > 0 \\ \Delta Q_k^i & \text{otherwise} \end{cases} \quad (8)$$

The consequent adaptation process is performed online, while the controller is operating over the real plant, so control is applied from the first moment, with increasing accuracy as the adaptation evolves. Nevertheless, adapting the consequents is not enough to obtain high accuracy if the rule base is not well chosen. This issue makes it necessary to define a mechanism for the online evolution of the controller's structure, i.e., for the addition of new antecedents and rules.

B. Structure evolution: adding new data clouds

For the evolution of the controller's structure, a second density measure, namely the global density Γ, is defined. Its definition is analogous to the one given for the local density, except that it takes into account the distance to all the previously observed samples z_j $(j = 1, ..., k-1)$. It has to be noted that the global density is computed for the points $z_k = [\vec{x}_k^T; u_k]^T$, whilst the local density is defined only for the input vectors $\vec{x}_k$. Using again the Cauchy kernel, this density can be defined as [20]

$$\Gamma_k = \frac{1}{1 + \frac{\sum_{j=1}^{k-1}(d_{kj})^2}{k-1}} = \frac{k-1}{k-1+\sum_{j=1}^{k-1}(d_{kj})^2} \quad (9)$$

and can be computed recursively by

$$\gamma_k^i = \frac{k-1}{k-1\left(z_k^T z_k + 1\right) - 2A_k + B_k} \quad (10)$$

where

$$\begin{aligned} A_k &= z_k^T \Xi_k; \\ \Xi_k &= \Xi_{k-1} + z_{k-1}; & \Xi_0 &= 0 \\ B_k &= B_{k-1} + ||z_{k-1}||^2; & B_0 &= 0 \end{aligned}$$

Since this measure considers all the existing samples, it provides an indication of how representative a given point z_k is with respect to the entire data distribution.

Additionally, and only for learning purposes, a focal point X_f^i and a radius r_{jk}^i are defined for each data cloud. The focal point is a real data sample that has high representative qualities. The fact that the focal point is always a real sample is important, as it avoids problems that may appear when only a descriptive measure is used instead (as in the case of the average of the points in the data cloud). In order to follow the philosophy of the proposed methodology (i.e., avoiding the need of predefining the rules' parameters), the focal point is updated online. Thus, for each new sample z_k, the following process is applied:

1) Find the associated data cloud $\aleph^i$, according to
$$\aleph^i = \arg\max_i \left(\gamma_k^i\right) \quad (11)$$
2) Check the representative qualities of the new data point using the following conditions:
$$\gamma_k^i > \gamma_f^i \quad (12a)$$
$$\Gamma_k > \Gamma_f^i \quad (12b)$$
where γ_f^i and Γ_f^i represent the local and global density of the current focal point, respectively.
3) If both conditions are satisfied, then replace the focal point by applying $X_f^i \leftarrow \vec{x}_k$.

The radius provides an idea of the spread of the data cloud. Since the data cloud does not have a definite shape or boundary, the radius represents only an approximation of the spread of the data in the different dimensions. It is also recursively updated as follows [23]:

$$r_{jk}^i = \rho \cdot r_{j(k-1)}^i + (1-\rho)\sigma_{jk}^i; \quad r_{j1}^i = 1 \quad (13)$$

where ρ is a constant that regulates the compatibility of the new information with the old one and is usually set to $\rho = 0.5$ [23]. The value σ_{jk}^i denotes the data cloud's local scatter over the input data space and is given by

$$\sigma_{jk}^i = \sqrt{\frac{1}{M^i}\sum_{l=1}^{M^i}\left|\left|X_f^i - \vec{x}_l\right|\right|_j^2}; \quad \sigma_{j0}^i = 1 \quad (14)$$

It is important to note that the radii and focal points are only used to provide an idea of the location and distribution of the data in the data clouds during the structure evolution process. However, they are not actually used to represent the data clouds or the fuzzy rules and do not affect the inference process at any point.

The structure-learning mechanism applied for the proposed SPARC is based on the following principles [19]:

A) Good generalization and summarization are achieved by forming new data clouds from data samples with high global density Γ.
B) Excessive overlap between data clouds is avoided by controlling the minimum distance between them.

Hence, the evolution of the structure is based on the addition of new data clouds and the associated rules. First, SPARC is initialized by creating a data cloud $\aleph^1$ from the first data sample $z_1 = [\vec{x}_1^T; u_1]^T$. The antecedent of the first rule is then defined by this data cloud and its consequent equals the value u_1. Next, for all the further incoming data samples z_k, the following steps are applied:

1) The sample $z_k = [\vec{x}_k^T; u_k]^T$ is considered to have good generalization and summarization capabilities if its global density is higher than the global density of all the existing data clouds. Thus, the following condition is defined:

$$\Gamma_k > \Gamma_f^i \quad \forall i = 1, ..., N \tag{15}$$

Note that this is a very restrictive condition that requires that the inequality is satisfied for all the existing data clouds, which is not very often for real data.

2) Check if the existing data clouds are sufficiently far from z_k with the following condition:

$$d_{ki} > r_{jk}^i/2 \quad \forall i = 1, ..., N \tag{16}$$

where d_{ki} represents the distance from the current sample to the focal point of the associated data cloud, X_f^i.

3) According to the result of the previous steps, take one of the following actions:

a) If conditions (15) and (16) are both satisfied, then create a new data cloud $\aleph^{N+1}$. The focal point of the new data cloud is $X_f^{N+1} = \vec{x}_k$. Its local scatter is initialised based on the average of the local scatters of the existing data clouds [23]:

$$\sigma_{jk}^{N+1} = \frac{1}{N}\sum_{l=1}^{N}\sigma_{jk}^l; \quad j = 1, ..., n \tag{17}$$

Additionally, the corresponding rule has to be added to the rule base. The antecedent of the new rule is defined by the newly created data cloud $\aleph^{N+1}$. For the consequent, we provide an initial value that guarantees that the output of the new controller when the input vector is equal to the focal point (i.e., $\vec{x} = X_f^{N+1}$) equals the controller's output under its previous configuration for that same input. Hence, the consequent for the new rule is computed as given by (6), i.e.,

$$Q^{N+1} = \hat{G}(X_f^{N+1}) \tag{18}$$

The rationale behind this initialization is to provide a smooth transition from the old to the new configuration of the controller. This avoids sudden changes in the output surface that could damage the controller's performance in the first time instants after the rule is added (and before the consequents are adapted) [16].

b) If the conditions are not satisfied, update the parameters of the data cloud $\aleph^i$ associated with z_k, as previously explained.

It is worth noting that the methodology presented for the evolution of the controller's structure starts from an empty controller. However, if an initial set of rules is known beforehand (e.g., provided by an expert or obtained from any other training method), it can be used for the initial controller. In this case, the algorithm's initialization step can be omitted.

IV. Simulation results

To test the newly proposed SPARC, two simulation examples are presented in this section. In addition, the performance of the controller is compared with the self-organising fuzzy controller presented in [17] (OSEFC). Although in both experiments we had a mathematical model of the system, we treated each model as a black box and no gain or any additional information was used during the control process.

First, in order to show its effectiveness, the proposed controller was applied in a water level problem for a sugar tank with a changing reference point. The discrete time equation of a sugar tank used in the experiment is:

$$y(k+1) = y(k) + T\left[\frac{-\sqrt{19.6y(k)}}{y^2(k)+1} + \frac{u(k)}{y^2(k)+1}\right] \tag{19}$$

where T is the sampling rate set to 0.5 seconds. In order to make sure that the water level never goes negative, which is physically impossible, the plant is simulated as follow:

$$y(k+1) = \max\left\{0, y(k) + T\left[\frac{-\sqrt{19.6y(k)}}{y^2(k)+1} + \frac{u(k)}{y^2(k)+1}\right]\right\} \tag{20}$$

In order to demonstrate that the controller has the ability to track a fast response plant, the desire output is chosen as:

$$ref(k) = \cos(0.05k) + sin(0.07k) + 3.7 \tag{21}$$

The error between the desired output and the real output ($e(k)$) and the derivative of the error ($de(k)$) were chosen as the inputs of the controller. The proposed controller started with an empty set of rules and new rules were added during the process if the two conditions mentioned in section III-B were satisfied. Fig. 2 shows the target output $ref(k)$, the actual output $y(k)$, and the tracking error $e(k)$. Additionally, Fig. 3 displays the distribution of the two inputs and the number of data clouds (fuzzy rules) generated at the end of the process. In this figure, the red stars represent the center of the data clouds.

At the beginning of the process, due to the lack of knowledge about the system and the use of an empty rule base, the performance of the controller was not satisfactory and the error is high. However, after few seconds the controller started minimising the error and tracking the reference in a satisfactory manner. As shown in Fig. 3, only four data clouds (fuzzy rules) were created during the control process.

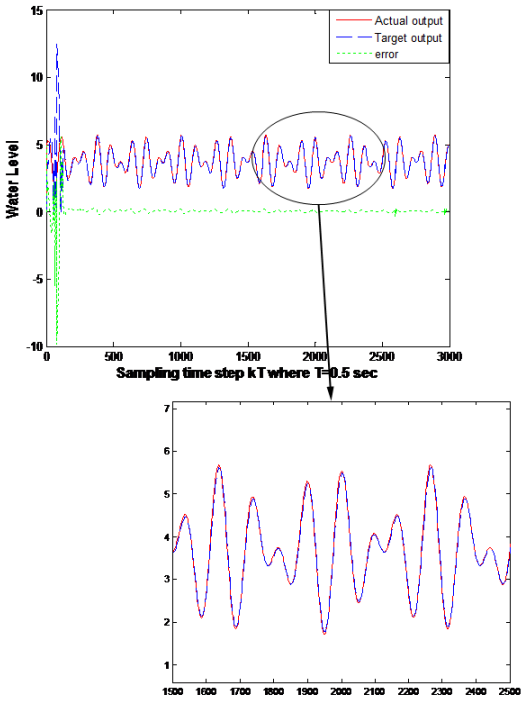

Fig. 2. Output tracking for the level of water in a sugar tank

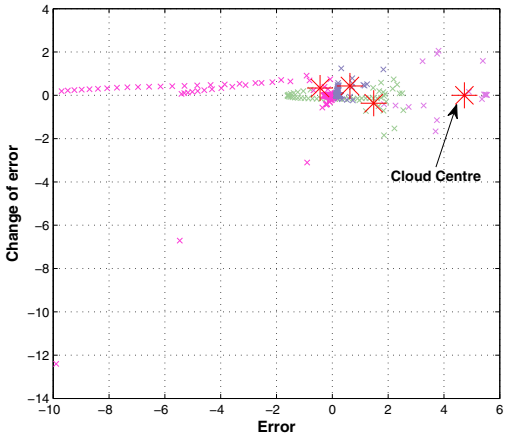

Fig. 3. Data clouds generated at the end of the control of the water level problem for a sugar tank. The red stars are the centers of the data clouds. Each data cloud represents a fuzzy rule

The second experiment consisted in the control of the temperature of a water bath. The plant chosen to be controlled is described by:

$$y(k+1) = a(T)y(k) + \frac{b(T)u(k)}{1+\exp(0.5y(k)-\gamma)} + (1-a(T))\,Y_0 \quad (22)$$

where $a(T) = \exp(-\alpha T)$ and $b(T) = \frac{\beta(1-\exp(-\alpha T))}{\alpha}$. The parameters of the plant are set to $\alpha = 10^{-4}$, $\beta = 8.7 \cdot 10^{-3}$, $\gamma = 40$, and $Y_0 = 25^{\circ}C$. The sampling period, T, is set as 25 seconds. The reference signal is a random step-wise function

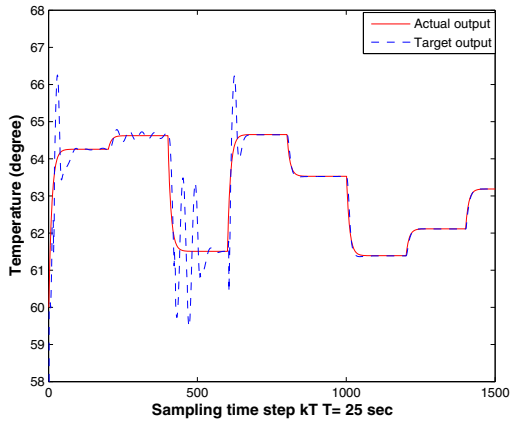

Fig. 4. Output tracking in the control of the temperature of a water bath during the initial moments

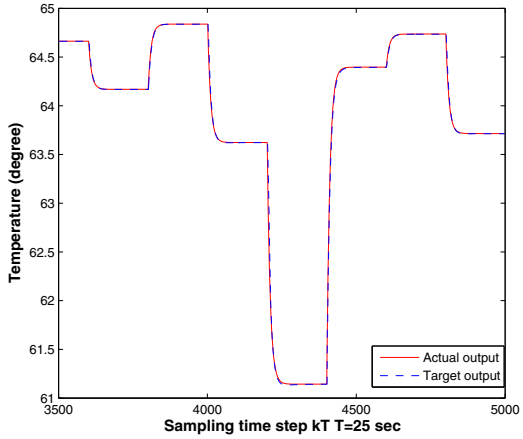

Fig. 5. Output tracking in the control of the temperature of a water bath after some data clouds have been generated

as shown in Fig. 4 (red line). As in the previous example, the input variables for the controller are the error $e(k)$ and the change of error $de(k)$.

Similarly to the previous simulation, the fuzzy controller started from zero fuzzy rules and membership functions and new rules were generated during the process. Fig. 4 and Fig. 5 show the tracking of the reference signal at two different moments of the process; also, Fig. 6 depicts the data clouds generated during the control process.

Almost the same scenario happened for the second experiment. At the beginning of the process, there is an oscillation in the plant's output due to the lack of initial knowledge. However, this oscillation is eventually removed when the controller started adding new rules and getting more information about the plant.

Table I illustrates the comparison between the newly proposed controller (SPARC), the conventional Mamdani-type fuzzy controller, and the self-organising controller (OSEFC) presented in [17]. The discrepancy between the real observa-

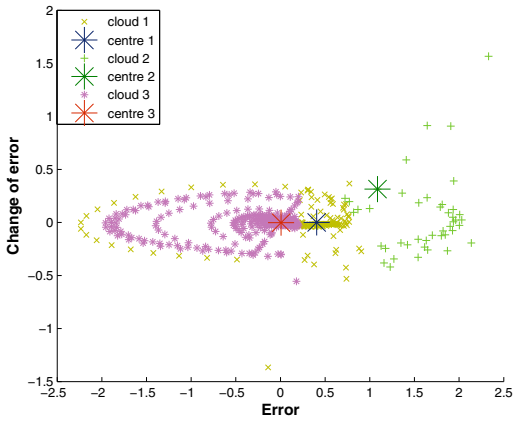

Fig. 6. Data clouds generated for the control of the temperature of a water bath. The red stars are the centers of the data clouds

TABLE I
COMPARISON OF SPARC WITH A CONVENTIONAL MAMDANI-TYPE FUZZY CONTROLLER AND A SELF-ORGANISING CONTROLLER (OSEFC)

	Water level control			Bath temperature control		
	AES	Rules	Params	AES	Rules	Params
Proposed Controller	405	4	12	132	3	9
OSEFC	389	9	54	98	6	30
Mamdani FC	518	49	686	205	49	686

tion and the target value has been used to calculate the errors. The sum of absolute error (AES) is used as the criteria for the comparison of the three controllers.

As it is clear from the table, the Mamdani-type controller requires the highest number of parameters (and, hence, computer memory) for both problems. On the other hand, the proposed SPARC presents a considerable improvement in terms of error and memory usage (parameters) compared to the Mamdani controller. OSEFC has a slightly better performance in terms of absolute error. However, we should note that at the beginning of the process for the OSEFC two membership functions were assigned for each input. Thus, four fuzzy rules formed the initial controller, although they were initialised to zero to create the situation of an empty controller. At the end of the process the number of rules increased to 9 rules for water level control experiment and 6 rules for bath temperature control. On the contrary, the proposed controller started with an empty rule set and no pre-defined parameters. Having this in mind, the proposed controller managed to track the water level of a sugar tank and temperature of the bath with only 4 and 3 rules, respectively, using less parameters and memory compared to OSEFC.

V. CONCLUSIONS

In this paper, a new approach for online self-evolving parameter-free (data cloud-based) fuzzy rule-based controller (SPARC) is proposed. Two illustrative examples are provided aiming a proof of concept. The proposed controller can start with no pre-defined fuzzy rules, no need to pre-define the range of output or control variable. This SPARC learns autonomously from its own actions while performing the control of the plant. It does not use any parameters, explicit membership functions, any off-line pre-training nor the explicit model (e.g., in a form of differential equations) of the plant. It combines the relative older concept of indirect adaptive control with the newer concepts of (self-)evolving fuzzy rule-based systems (and controllers, in particular) and with the very recent concept of parameter-free, data cloud and data density based fuzzy rule-based systems (and controllers in particular). It has been demonstrated that a fully autonomously and in an unsupervised manner (based only on the data density and selecting representative prototypes/focal points from the control hyper-surface acting as a data space) it is possible generate a parameter-free control structure and evolve it in on-line mode. Moreover, the results demonstrate that this autonomous controller is effective (has comparative error and performance characteristics) to other known controllers, including self-learning ones, but surpasses them with its flexibility and extremely lean structure (small number of prototypes/focal points which serve as seeds to form parameter-free and membership function-free fuzzy rules based on them). The illustrative examples aim primarily proof of concept.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Junta de Andalucía under Project no. TIC2906 and by the Spanish Ministry of Science and Innovation under Project no. SAF2010-02558.

REFERENCES

[1] H. Butler, *Model Reference Adaptive Control: From Theory to Practice*. New York: Prentice Hall, 1992.
[2] V. V. Chalam, *Adaptive Control Systems: Techniques and Applications*. New York and Basel: Marcel Dekker, Inc., 1987.
[3] H. Kaufman, I. Bar-Kana, and K. Sobel, *Direct Adaptive Control Algorithms: Theory and Applications*. New York: Springer-Verlag, 1994.
[4] S. Sugawara and T. Suzuki, "Applications of fuzzy control to air conditioning environment," *Journal of Thermal Biology*, pp. 456–472, 1993.
[5] Y. Liu and Y. Zheng, "Adaptive robust fuzzy control for a class of uncertain chaotic systems," *Nonlinear Dynamics*, vol. 57, no. 3, pp. 431–439, 2009.
[6] M. Mucientes and J. Casillas, "Quick design of fuzzy controllers with good interpretability in mobile robotics," *Fuzzy Systems, IEEE Transactions on*, vol. 15, no. 4, pp. 636–651, 2007.
[7] K. Shimojima, T. Fukuda, and Y. Hashegawa, "Self-Tuning modelling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm," *Journal on Fuzzy Sets and Systems*, vol. 71, pp. 295–309, 1995, 3.
[8] H. Mingzhi, W. Jinquan, M. Yongwen, W. Yan, L. Weijiang, and S. Xiaofei, "Control rules of aeration in a submerged biofilm wastewater treatment process using fuzzy neural networks," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 428–10 437, Sep. 2009.

[9] C. Li and C. Lee, "Self-organizing neuro-fuzzy system for control of unknown plants," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 1, pp. 135–150, Feb. 2003.

[10] H. Pomares, I. Rojas, J. González, F. Rojas, M. Damas, and F. J. Fernández, "A two-stage approach to self-learning direct fuzzy controllers," *International Journal of Approximate Reasoning*, vol. 29, no. 3, pp. 267–289, Mar. 2002.

[11] I. Rojas, H. Pomares, J. Gonzalez, L. Herrera, A. Guillen, F. Rojas, and O. Valenzuela, "Adaptive fuzzy controller: Application to the control of the temperature of a dynamic room in real time," *Fuzzy Sets and Systems*, vol. 157, no. 16, pp. 2241–2258, 2006.

[12] W. Wang, Y. Chien, and I. Li, "An on-line robust and adaptive T-S fuzzy-neural controller for more general unknown systems," *International Journal of Fuzzy Systems*, vol. 10, no. 1, pp. 33–43, 2008.

[13] P. Angelov, R. Buswell, J. A. Wright, and D. Loveday, "Evolving rule-based control," in *EUNITE Symposium*, Tenerife (Spain), 2001, pp. 36–41.

[14] P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE on System, Man, and Cybernetics*, pp. 484–498, 2004.

[15] D. Pasaltis, A. Sideris, and A. Yamamura, "A multilayer neural network controller," *IEEE Trans. on Control Systems Managements*, pp. 17–21, 1988.

[16] A. B. Cara, H. Pomares, and I. Rojas, "A new methodology for the online adaptation of fuzzy Self-Structuring controllers," *Fuzzy Systems, IEEE Transactions on*, vol. 19, no. 3, pp. 449–464, 2011.

[17] A. B. Cara, H. Pomares, I. Rojas, Z. Lendek, and R. Babuska, "Online self-evolving fuzzy controller with global learning capabilities," *Evolving Systems*, vol. 1, no. 4, pp. 225–239, Oct. 2010.

[18] A. B. Cara, Z. Lendek, R. Babuska, H. Pomares, and I. Rojas, "Online self-organizing adaptive fuzzy controller: Application to a nonlinear servo system," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1–8.

[19] P. Angelov and R. Yager, "Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density," in *2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, Apr. 2011, pp. 62–69.

[20] P. Angelov and X. Zhou, "On line learning fuzzy rule-based system structure from data streams," in *IEEE International Conference on Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence)*. IEEE, Jun. 2008, pp. 915–922.

[21] P. Angelov, "A fuzzy controller with evolving structure," *Information Sciences*, vol. 161, no. 1-2, pp. 21–35, Apr. 2004.

[22] H. Pomares, I. Rojas, J. Gonzalez, M. Damas, B. Pino, and A. Prieto, "Online global learning in direct fuzzy controllers," *Fuzzy Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 218–229, 2004.

[23] P. Angelov and X. Zhou, "Evolving fuzzy systems from data streams in Real-Time," in *2006 International Symposium on Evolving Fuzzy Systems*. IEEE, Sep. 2006, pp. 29–35.