Knowledge Acquisition in Fuzzy-Rule-Based Systems With Particle-Swarm Optimization

R. P. Prado, S. García-Galán, J. E. Muñoz Expósito, and A. J. Yuste

***Abstract*—Knowledge acquisition is a long-standing problem in fuzzy-rule-based systems. In spite of the existence of several approaches, much effort is still required to increase the efficiency of the learning process. This study introduces a new method for the fuzzy-rule evolution that forms an expert system knowledge: the knowledge acquisition with a swarm-intelligence approach (KASIA). Specifically, this strategy is based on the use of particle-swarm optimization (PSO) to obtain the antecedents, consequences, and connectives of the rules. To test the feasibility of the suggested method, the inverted-pendulum problem is studied, and results are compared for two of the most extensively used methodologies in machine learning: the genetic-based Pittsburgh approach and the Q-learning-based strategy, i.e., state-action-reward-state-action (SARSA). Moreover, KASIA is analyzed as a learning strategy in fuzzy-rule-based metascheduler design for grid computing, and performance is compared with other scheduling strategies based on genetic learning and existing scheduling approaches, i.e., EASY-backfilling and ESG+local periodical search. To be more precise, simulation results prove the fact that the proposed strategy outperforms classical learning approaches in terms of final results and computational effort. Furthermore, the main advantage is the capability to control convergence and its simplicity.**

***Index Terms*—Genetic fuzzy systems, grid computing, knowledge acquisition, reinforcement learning, swarm intelligence.**

I. Introduction

FUZZY-RULE-BASED systems (FRBSs) incorporate expert knowledge in their knowledge bases (KBs), and thus, they are considered to be expert systems [1]. This kind of systems has been used in real applications in a wide range of fields: speech and music discrimination in audio [2], heating control, ventilation, and air conditioning systems [3], computational network management [4], and other sciences, such as risk modeling and decision making [5], [6]. FRBS success strongly depends on the quality of their acquired knowledge or KB. However, the incorporation of an expert knowledge is not a feasible option in most FRBS application areas, and it is generally necessary to resort to learning systems that are able to acquire knowledge. One way to solve this problem may consist of an optimization method, which makes it a self-learning system. To be precise, the learning system is responsible for generating rules that are incorporated into an FRBS.

There exist several schemas for the learning of fuzzy rules. In this sense, the role of genetic algorithms (GAs) must be pointed out [7]. Application of GA fields range from optimization in neural networks [8] and fuzzy systems design [9] to production scheduling [10]. They encode every candidate individual as a chromosome, and they define a population as those individuals taking part of the considered problem search space. Every individual consists of a rule or a set of rules that compete for being included in a successful population of the selection process that bases its decision on a fitness or an objective function through iterations [11]. Specifically, there exist two successful genetic fuzzy learning strategies: the Pittsburgh [12] and Michigan approaches [13].

On the other hand, particle swarm optimization (PSO) has proved effective in optimizing complex multidimensional discontinuous problems in a wide range of research areas, including renewable energies [14] and electromagnetics [15]. It has been experimentally shown that PSO is able to achieve a fast convergence, it is not highly sensitive to the population size, and it scales well [16]. Moreover, PSO is a stochastic evolutionary algorithm based on Swam Intelligence, which has proved to outperform other optimization procedures such as GAs [15]. One of the major advantages of PSO over GAs is related to its simple implementation and reduced number of fixing parameters. PSO does not need genetic operators, namely, crossover or mutation, but particles update themselves, considering their internal velocities. In addition, particles have memory, and they consider unidirectional information exchanges, which notably reduces the number of required communications among particles. Furthermore, a major control over particle convergence can be exercised in comparison with GAs, thereby decreasing the required number of evaluations in a large way.

Due to its features, PSO is not only extensively used for science research but for engineering applications as well. In this way, recent works on ad hoc wireless networks [17], artificial neural network training [18], or simultaneous localization and mapping problems [19] can be shown. In addition, PSO has been used together with fuzzy logic in nonlinear identification techniques [20], [21], collective robotic search applications [22], and power system stabilizers [23]. However, as far as the authors' knowledge is concerned, PSO has never been used for knowledge acquisition in FRBSs in such a way that a rule base (RB) as a whole represents a swarm individual. In this study, the utilization of PSO is suggested to obtain more efficient KBs for these systems. Specifically, the proposed algorithm, i.e., knowledge acquisition with a swarm-intelligence approach (KASIA), is

Manuscript received October 23, 2009; revised March 8, 2010, May 28, 2010, and July 1, 2010; accepted July 19, 2010. Date of publication August 3, 2010; date of current version December 3, 2010. This work was supported by the Andalusian Government under Research Project P06-SEJ-01694.

The authors are with the Telecommunication Engineering Department, Jaén University, 23071 Jaén, Spain.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2010.2062525

compared with one of the most successful strategies for knowledge acquisition in FRBS, i.e., the Pittsburgh approach. The Pittsburgh approach considers a set of RBs to be the evolution population that is also the situation of the proposed PSO-based strategy. This way, both methodologies' performance can be compared, bearing in mind the same number of RB evaluations at every algorithm step. Moreover, KASIA is compared with another successful reinforcement-learning (RL) strategy: state-action-reward-state-action (SARSA). SARSA is a model-free algorithm founded on an "on-policy" control introduced in [24] as a modification of the present temporal difference technique Q-learning strategy [25]. In SARSA, the possible system states are encoded, taking into account a finite range of controlled variables values, and each state s is associated with an action a. The goal is to find the most suitable action for every state. The strategy operation essentially consists of updating the expected reinforcement, $Q(s, a)$, by considering action a in state s through a set of steps within a range of episodes. Thus, in every episode, there is a final state, and the episode finishes when the final state is reached.

As far as performance evaluation is concerned, the inverted-pendulum problem [26] is considered to incorporate a knowledge-based system to evaluate the acquired knowledge [27]. Simulation results indicate that the suggested schema improves the genetic-based strategy and SARSA in its final result, and it requires a more reduced computational effort. Moreover, a major advantage is given by its ability to control convergence behavior and its implementation simplicity. In addition, KASIA is used as a knowledge-acquisition strategy in a well-known problem of practical importance, i.e., the design of metaschedulers for the emerging computational grids. Grid computing is a high-performance computational framework based on the share of geographically distributed and heterogeneous computational resources with the aim of solving of large-scale problems in science, engineering, and technology [28]. Due to the inherent dynamism of grids, resource coordination and cooperation are two of the most-challenging problems, and with regard to this fact, a lot of research has been carried out in scheduling. To be precise, FRBS models are increasingly attracting scheduling research community to achieve near-optimal schedules. However, FRBS performance highly depends on acquired knowledge quality and, thus, with the learning strategy. In this study, the KASIA application is suggested for the self-learning of fuzzy-rule-based metaschedulers for grid computing, and the strategy is compared with fuzzy schedulers made up with genetic learning and two existing scheduling approaches: *EASY-Backfilling* and *ESG+Local search periodical* [29]. Thus, this work represents a new effort toward the implementation of more efficient reinforcement knowledge-acquisition strategies.

The rest of this paper is organized as follows. In Section II, a brief introduction to knowledge acquisition strategies applied to FRBSs and PSO fundamentals is presented. The suggested learning strategy, i.e., KASIA, is introduced in Section III. Section IV analyzes the feasibility of KASIA through the control of inverted pendulum and provides a comparison to a genetic-learning strategy for rule evolution, i.e., the Pittsburgh approach, and the other unsupervised RL strategy, i.e., SARSA. In addition, KASIA is evaluated as a learning strategy for fuzzy-rule-based metaschedulers for computational grids, and results are compared with other scheduling approaches. Finally, Section V concludes the paper.

II. Background

Machine-learning strategies can be broadly categorized into supervised and unsupervised ones [25]. In the former, learning strategies are based upon known models or previous examples [30], whereas in the latter, a reward and punishment process considering performance feedback is taken into account. RL is an unsupervised learning strategy mainly concerned with the way to achieve the best response when a role model is, at first, unknown. Therefore, a machine learns through trial-and-error iterations where performance information is retrieved from environment. As mentioned earlier, this study is focused on the analysis and improvement of RL schemas for the evolution of fuzzy rules.

In this regard, automatically defining an FRBS is considered to be a search or optimization problem, and GAs are well-known evolutionary computation techniques for global optimization. Furthermore, these techniques have been proved to be able to find quasioptimal solutions in complex search spaces [1]. Genetic FRBSs (GFRBSs) are FRBS designing methods that consider genetic-learning strategies as evolutionary algorithms to optimize or generate part or all its KB. Thus, the KB makes up the solution space, and it must be coded in a suited genetic representation to face genetic operations ranging from selection to crossover and mutation. In relevant FRBSs, such as descriptive Mamdani-type FRBS, the KB consists of two essential components. On the one hand, there is the Data Base (DB), keeping in the scale factors and memberships functions, and on the other hand, there is the fuzzy RB. Deciding which parts of a given FRBS must be evolved is generally achieved on the basis of a wide range of FRBS design criteria. It is to be noticed the fact that reducing search space generally leads to faster and simpler optimization procedures. Despite this, these results are considered to be sub-optimal. A similar reasoning can be followed for increasing search space. In this study, attention is focused on the RB optimization problem or knowledge acquisition.

In the genetic learning of RBs, it is assumed that there exists a predefined set of membership functions that refer to rules by means of linguistic labels. An RB can be encoded either taking each RB as an individual of the population or considering each single rule within an RB as a candidate or chromosome. The first case corresponds to the so-called Pittsburgh approach [12], which obtains a population of RBs where the best suited one is selected as the final RB. However, the so-called Michigan approach [13] encodes and evolves each rule as an individual, and a single RB is achieved.

The goal of a genetic rule learning system is to evolve efficiently a set of coadapted rules that cooperate with each other to solve a given problem. One of the most challenging problems in Michigan approach is to solve the conflict between individual and collective interests, namely, the *cooperation versus competition* problem (CPP) [1]. In the classical Michigan approach,

every rule cooperates to achieve the best fitness; it receives payoff on the basis of its contribution to the present state, and it competes to be selected for the next generation. Nevertheless, the conflict among rule interests as individuals does not occur in Pittsburgh approach where competition appears among complete sets of rules, not among individual rules. It is to be noticed, on the other hand, that the evaluation of a complete set of rules usually requires a greater computational effort, i.e., memory and processing time, than the Michigan approach. Furthermore, a major problem of the Pittsburgh approach is given by the little reinforcement information of every set of rules. Typically, it consists of a single scalar fitness for every collection. The Pittsburgh approach is not able to take advantage of the credit apportion systems in contrast with Michigan approach. In the Pittsburgh approach, selection and replacement favor sets of coadapted rules; however, crossover can dramatically affect the set of cooperating rules, as good rule associations are not considered by this genetic operator.

Hence, considering the recent importance of FRBSs, lot of effort must be made toward the development of new knowledge-acquisition strategies that outperform the existing ones. This study suggests the application of PSO to optimize FRBS knowledge. PSO is a stochastic optimization technique inspired by social behavior patterns related to organisms that interact with its neighbors [31], [32]. Originally, the aim was to simulate graphically the choreography of birds within a flock. Nonetheless, a swarm model that could be used as an optimizer was derived from these studies, and several modifications, such as adaptive PSO [33], Euclidean PSO [34], and fuzzy PSO [35], have recently been suggested to improve its performance.

In canonical PSO, each individual is known as a "particle" and it moves within a multidimensional space representing the search space. The system is initialized with a set of M particles randomly distributed in the n-dimensional search space $\mathbb{R}^N$. Furthermore, a real function f is defined in such space constituting the fitness function $f : \mathbb{R}^N \rightarrow \mathbb{R}$. Each particle position is modified through iterations with the aim of finding the optimum position, where an optimum value for the fitness function or optimum state is achieved. This modification is carried out on the basis of three components leading the whole process: an inertial component, a self-recognition component or an inner tendency to return to its best position, and a social component representing the swarm particle leaning to move toward the best position found by its neighbors. Thus, at every iteration $(t+1)$, position x and velocity v of particle i are updated by

$$v_i^{(t+1)} = \omega v_i^t + d_1 r_1 (Pb_i^t - x_i^t) + d_2 r_2 (Gb^t - x_i^t) \quad (1)$$

$$x_i^{(t+1)} = x_i^t + v_i^{(t+1)} \quad (2)$$

where d_1 and d_2 are constant weight factors, Pb_i represents the best position achieved by particle i, Gb_i indicates the best position found by the neighbors of particle i, r_1 and r_2 are random factors in the [0, 1] interval, and ω denotes inertia weight. Moreover, to ensure the algorithm convergence, v values are constrained to the interval $[-v_{\max}, v_{\max}]$. Furthermore, another key factor for convergence is to set efficiently the inertia weight ω. A high value for ω favors the global search, whereas a low value favors local search. In other words, ω can be configured to balance both types of search and, thus, to reduce the number of required operations to reach the optimum value. Hence, ω is usually associated with a decreasing value through iterations in a way that local searches are intensified once the whole space has been explored. Typically, ω is initialized with a value that is close to the unity.

In addition, it is relevant to configure optimally cognitive and social weights, d_1 and d_2, respectively. Despite the fact that it is not decisive for convergence, a good fixing of these two parameters can lead to a reduction of the required number of iterations and local optimum avoidance [36]. Furthermore, the number of iterations is a critical factor [15]. A high number of iterations may lead the particles to stagnate in a location pending a decrease in the inertia weight to further exploring the maximum. On other hand, if the number of iterations is not high enough, particles may not be able to explore the search space and, thereby, reach the global optimum. In addition, it is relevant to set the population size efficiently. Although large populations allow a thorough exploration of the search space, they also increase the number of evaluations and, thereby, the computational effort. It is shown the fact that PSO can sufficiently explore search space considering a population size around 30 [37]. Therefore, to compare KASIA with genetic strategies fairly, the Pittsburgh approach is considered in this study. As stated before, the Pittsburgh approach considers every candidate population or RB evaluation at every iteration as it is seen in the proposed PSO-based strategy, and therefore, a step-by-step comparison is possible in terms of computational effort and convergence behavior.

Moreover, one of the most relevant advantages of PSO over GAs is related to its simplicity. GAs generally resort to three main operators: selection, crossover, and mutation. Different configurations of these operators lead to a wide range of implementations. However, in PSO, there only exist a major operator, i.e., velocity. Reducing the number of operators is usually translated into decreasing computational effort and into eliminating the need for selecting the best operator for a given optimization problem as well. Moreover, an important difference between both algorithms is the ability to control convergence [15]. Although crossover and mutation configuration can affect the population converge behavior, they cannot achieve as high a level of control as PSO through inertia weight ω. As mentioned before, ω allows the converge rate and the stagnation level in PSO to be set. In GAs, the stagnation occurs when every individual presents the same genetic code, and thus, crossover has little or no effect on the population. On the other hand, PSO can prevent or even control this phenomenon. High inertia weight allows the particles in the swarm stay in a sawing motion around the best global solution, and therefore, best fitness locations may be found.

III. Knowledge Acquisition With a Swarm-Intelligence Approach

The suggested standard PSO-based algorithm, i.e., KASIA, is focused on knowledge acquisition in FRBSs. As stated earlier,

PSO is a well-known optimization strategy. However, due to the novelty of PSO schema for Mamdani-type fuzzy-rule evolution, its adaptation to an FRBS environment is explained.

First, the considered search-space definition must be set up, together with the considered structure definition for each particle in the swarm. As mentioned before, an expert system knowledge based on fuzzy rules is located in its KB. Hence, in this study, each particle represents an RB and an RB is represented by a matrix, where every row describes a fuzzy rule. Each rule is characterized by the antecedents, the consequences, and the connector (AND/OR). The structure for particle i is presented as follows:

$$P_i = \begin{bmatrix} a^i_{1,1} & a^i_{1,2} & \dots & a^i_{1,n} & b^i_1 & c^i_1 \\ a^i_{2,1} & a^i_{2,2} & \dots & a^i_{2,n} & b^i_2 & c^i_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a^i_{m,1} & a^i_{m,2} & \dots & a^i_{m,n} & b^i_m & c^i_m \end{bmatrix} \tag{3}$$

where m is the number of rules of the KB, n is the number of input variables, a represents the input variables, and b denotes the consequences (in this case, a single consequence is considered). Finally, c represents the connector.

Antecedent and consequence representation: Every membership function of every variable is enumerated consecutively starting from 1. This way, 0 value indicates the variable absence in the rule, and negative values represent the complementary set. It must be highlighted here that to keep the coherence of the fuzzy rules, a rule presenting 0 value for all its antecedents is not allowed, i.e., it is considered as a nonsense to contribute to controller output, regardless of all the controller inputs. On other hand, however, consequences are allowed to be set to zero in this study as a way of representing their associated rule absence.

Connector representation: If connector is set to 1, it indicates that the connector for the antecedents of the rule is AND. However, if it is set to 2, it indicates that the connector for the antecedents of the rule is OR.

Note that the fuzzy-set membership functions remain the same during the whole learning process and that the particle has no information about the shape of the membership functions; therefore, the rule interpretability is not affected through the whole learning process.

Therefore, the problem consists of the way to obtain a matrix that represents a set of fuzzy rules making up a high-quality KB for the FRBS. It must be pointed out that each element within the position matrix P_i is subject to the solution space of its group, namely, antecedents, consequences, or connectives. Thus, the following constraints are borne in mind for each particle i in the swarm:

$$a^i_{j,k} \in [-NF_{\text{in}},\, NF_{\text{in}}], \quad j \in \{1, 2, \dots, m\} \quad k \in \{1, 2, \dots, n\} \tag{4}$$

$$b^i_j \in [-NF_{\text{out}},\, NF_{\text{out}}], \quad j \in \{1, 2, \dots, m\} \tag{5}$$

where NF_{in} and NF_{out} represent the number of membership functions for the input and the output, respectively, where $NF_{\text{in}},\, NF_{\text{out}} \in \mathbb{N}$. Thus, the connector value must satisfy $c^i_j \in \{1, 2\}$. Moreover, the swarm-position-updating process must be considered. With this aim, the velocity matrix is introduced for this approach. The velocity at every iteration is represented as a matrix with the particle dimensions for each particle i

$$V_i = \begin{bmatrix} v^i_{1,1} & v^i_{1,2} & \dots & v^i_{1,n} & v^i_{1,n+1} & v^i_{1,n+2} \\ v^i_{2,1} & v^i_{2,2} & \dots & v^i_{2,n} & v^i_{2,n+1} & v^i_{2,n+2} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ v^i_{m,1} & v^i_{m,2} & \dots & v^i_{m,n} & v^i_{m,n+1} & v^i_{m,n+2} \end{bmatrix} \tag{6}$$

where $v^i_{j,k} \in [V_{\min},\, V_{\max}]$, $j \in \{1, 2, \dots, m\}$, and $k \in \{1, 2, \dots, n+2\}$. $V_{\max}$ and $V_{\min}$ represent the maximum and the minimum values allowed for the velocity, respectively. Hence, once all the particles or RB population and their associated velocities have been defined, they are subject to evaluation. As a result, a fitness value that estimates each individual's quality is obtained, which drives the updating process. This way, on the basis of this performance index, the velocity matrix modification is undertaken, considering the best RB for the particle $P^{\#}(t)$ and the best position found in the swarm $P^*(t)$ so far:

$$V(t+1) = \omega \otimes V(t) \oplus (d_1 * r_1) \otimes (P^{\#}(t) \ominus P(t)) \oplus (d_2 * r_2) \otimes (P^*(t) \ominus P(t)) \tag{7}$$

where $\otimes$ denotes multiplication, and $\oplus$ and $\ominus$ represent the regular addition and subtraction between matrices, respectively. Finally, the particle position is updated as follows:

$$P(t+1) = P(t) \oplus V(t+1). \tag{8}$$

Thus, the process is repeated until the stopping criteria are met. One of the most extended criteria is the number of iterations, which is dealt with in Section IV. In addition, previous to the earlier step, both Gbest- and Pbest-associated particles must be recorded, i.e., $P^{\#}(t)$ of each particle and $P^*(t)$ of the swarm.

Furthermore, a range of constraints are needed to make up an RB coherently. Initially, all the particles are randomly generated within the search-space-allowed boundaries. In spite of this, particles update their position cyclically through iterations, as described in (8). As this operation may lead the particle to exceed the solution space, some constraints must be imposed. Concretely, each particle considers the following expressions within the updating process:

$$a^i_{j,k} = \begin{cases} NF_{\text{in}}, & \text{if } a^i_{j,k} > NF_{\text{in}} \\ -NF_{\text{in}}, & \text{if } a^i_{j,k} < -NF_{\text{in}} \end{cases} \tag{9}$$

$$b^i_{j,k} = \begin{cases} NF_{\text{out}}, & \text{if } b^i_{j,k} > NF_{\text{out}} \\ -NF_{\text{out}}, & \text{if } b^i_{j,k} < -NF_{\text{out}} \end{cases} \tag{10}$$

$$c^i_j = \begin{cases} 1, & \text{if } c^i_j < 1 \\ 2, & \text{if } c^i_j > 2. \end{cases} \tag{11}$$

As stated before, to keep the RB base coherence, all the antecedents of a given rule cannot be allowed to equal zero simultaneously. In this case, whether in the RB initialization or in the updating process, other considerations are contemplated. Specifically, a rule j of RB i is initialized as a consequence of violating the coherence conditions of a fuzzy rule (if $\sum_{k=1}^{n} a^i_{j,k} = 0 \Rightarrow \text{init}\, a^i_j$). Consequences, however, are

Algorithm 1 Knowledge Acquisition with a Swarm Intelligence Approach, KASIA.

Initialization
1. Swarm: Num_particles, RB_Num_rules (m),
Num_iter, Init_rate (r_0), Inertial_weight ω, Factors c_1 and c_2.
2. Random setting of RB-Swarm position.
$for\ every\ particle/RB\ i\ and\ every\ rule\ j,$
(a) No zero antecedents (rules coherence):
$while(\sum_{k=1}^{n} a_{j,k}^{i} == 0)$ (a rule with no antecedents)
Random setting of $a_{j,k}^{i}\ for\ random\ k$
end
(b) $if\ \mid a_{j,k}^{i} \mid > NF_{in}\ then\ Eq.\ 9$
(c) $if\ \mid b_{j,k}^{i} \mid > NF_{out}\ then\ Eq.\ 10$
(d) $if\ c_j^i \notin \{1,2\}\ then\ Eq.\ 11$
end
3. Random setting of velocity.
4. Velocity Constraints.
5. Initialize Gbest(P^*)/Pbest($P^{\#}$)
Do
Do
1. Update position. Eq. 8.
2. Constraints RB-Swarm position.
(a) No zero antecedents (rules coherence):
$while\ (\sum_{k=1}^{n} a_{j,k}^{i} == 0)$(a rule with no antecedents)
Random setting of $a_{j,k}^{i}\ for\ random\ k$
end
(b) $if\ \mid a_{j,k}^{i} \mid > NF_{in}\ then\ Eq.\ 9$
(c) $if\ \mid b_{j,k}^{i} \mid > NF_{out}\ then\ Eq.\ 10$
(d) $if\ c_j^i \notin \{1,2\}\ then\ Eq.\ 11$
3. Evaluate-Fitness. Evaluation system.
Particles ++
While(Num_particles)
Update Gbest (P^*).
Do
1. Update Pbest ($P^{\#}$).
2. Update velocity. Eq. 7.
3. Velocity Constraints.
Particles ++
While(Num_particles)
iter++
While(Num_iter)
Return solution: Gbest (P^*)

allowed to equal zero in such a way that their associated rule has no effect on the controller output, as mentioned earlier.

On the other hand, it must be highlighted that particle size remains unchanged through the whole learning process, i.e., each particle size and, hence, the KB size (i.e., rule number) are not modified. This is also the case of the Michigan approach, where the KB is fixed at the beginning of the algorithm; therefore, the successful base rules number is known *a priori* [1]. The reason for this can be found on the metric used (i.e., difference $\ominus$) for calculating the distance of every particle to its best position and to the best global position. However, modified difference metrics could also be considered to let the algorithm work with several RB sizes. In addition, it is to be noted that KASIA does not face fuzzy-rule weight modification. Nevertheless, this could be considered later through the tuning of the fuzzy KB. The proposed algorithm is illustrated in Algorithm 1.

Regarding computational effort, it must be pointed out that optimization algorithms are computationally rated generally according to the number of function evaluations (FEs) required [33]. Hence, because an FE in KASIA is associated with an RB evaluation (as in the Pittsburgh approach), computational effort is measured interchangeably in RB evaluations or FE in this study. Specifically, in KASIA, each particle or RB is evaluated at every iteration, and thus, computational effort for an experiment can be expressed as $\mathrm{CE}_{\mathrm{KASIA}} = \mathrm{NP} * \mathrm{num}_{\mathrm{iter}}$, where NP denotes the size of the swarm, and $\mathrm{num}_{\mathrm{iter}}$ is the number of iterations or stopping condition. Note that KASIA considers the evolution of a fuzzy RB as a whole at every iteration, where antecedents, consequences, and connectors are obtained simultaneously by position updating expression, i.e., (8), and analog RB coherence constraints, i.e., (9)–(11). Hence, additional computational effort of KASIA at every iteration is equally distributed for all rule components.

Finally, it must be mentioned that KASIA convergence control properties are derived from standard PSO analysis. Control convergence is one of the more relevant aspects of PSO, and analytic results have been presented to prove the algorithm efficiency in this sense. A generalized model of PSO was proposed in [38], including strategies to control the convergence properties of the swarm. In this study, Clerc and Kennedy analyze how a particle searches a complex problem space in discrete time (i.e., the algebraic view) and continuous time (i.e., the analytical view), and this analysis leads to a generalized model of PSO that defines a set of coefficients to control the system-convergence behavior in ways that are advantageous for optimization. Specifically, the definition of constriction coefficients is derived. Constriction coefficients enable control over the swarm dynamics, including its exploration and exploitation tendencies. It should be mentioned that the analysis indicates that no problem-specific parameters may need to be specified. The study initially suggests a reordering of the original velocity equation, i.e., (1)

$$v(t+1) = \chi(v(t) + \varphi_1(Pb - x(t)) + \varphi_2(Gb - x(t))) \quad (12)$$

and defines φ, p, and $y(t)$ as

$$\varphi = \varphi_1 + \varphi_2, \qquad p = \frac{\varphi_1 Pb + \varphi_2 Gb}{\varphi_1 + \varphi_2}, \qquad y(t) = p - x(t). \quad (13)$$

Thus, substituting (1) and (2) into the original system, the system is redefined as

$$v(t+1) = \chi(v(t) + \varphi y(t)) \quad (14)$$

$$y(t+1) = -v(t) + (1-\varphi)y(t). \quad (15)$$

Assuming a continuous process, a classical second-order differential equation is deduced as

$$\frac{\partial^2 v}{\partial t^2} + \ln(e_1 e_2)\frac{\partial v}{\partial t} + \ln(e_1)\ln(e_2)v = 0$$

$$\begin{cases} e_1 = & 1 - \dfrac{\varphi}{2} + \dfrac{\sqrt{\varphi^2 - 4\varphi}}{2} \\ e_2 = & 1 - \dfrac{\varphi}{2} - \dfrac{\sqrt{\varphi^2 - 4\varphi}}{2} \end{cases} \quad (16)$$

where e_1 and e_2 are the roots of $\lambda^2 + (\varphi - 2)\lambda + 1 = 0$. Hence, the general solution is formulated as

$$v(t) = c_{01}e_1^t + c_{02}e_2^t \quad (17)$$

$$y(t) = \frac{1}{\varphi}(c_{01}e_1^t(e_1 - 1) + c_{02}e_2^t(e_2 - 1)) \quad (18)$$

where coefficients c_{01} and c_{02} depend on initial conditions $v(0)$ and $y(0)$:

$$\begin{cases} c_{01} = \dfrac{-\varphi y(0) - (1-e_2)v(0)}{e_2 - e_1} \\ c_{02} = \dfrac{\varphi y(0) + (1-e_2)v(0)}{e_2 - e_1} \end{cases}, \qquad e_1 \neq e_2$$

$$\begin{cases} v(0) = c_{01} + c_{02} \\ y(0) = -\dfrac{c_{01} + c_{02}}{2} \end{cases}, \qquad e_1 = e_2 \ (\varphi = 4). \tag{19}$$

Therefore, we must have $v(0) + 2y(0) = 0$, in a way that discontinuity is avoided. From these results, a guideline for preventing the explosion of the system can be obtained. As can be inferred, it depends on whether it satisfies $\max(|e_1,||e_2|) > 1$. Hence, the convergence criterion is met when $\chi < \min(1/|e_1|, 1/|e_2|)$. Under this consideration, the constriction coefficient χ is formulated as $\chi = \kappa/\max(1/e_1, 1/|e_2|)$ or alternatively considering $|e_1,| \leq |e_2|, \chi = \kappa/|e_2|, \kappa \in]0, 1[$. In addition, substituting $|e_2|$, the constriction factor can be expressed as

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \qquad \varphi > 4, \qquad \kappa \in]0, 1[. \tag{20}$$

Finally, coming back to the original system, the following expression is obtained for position:

$$x(t+1) = \chi v(t+1) + \chi x(t) + (1-\chi)\frac{\varphi_1 Pb + \varphi_2 Gb}{\varphi_1 + \varphi_2}. \tag{21}$$

Bearing in mind the earlier expression, the constriction coefficient can be seen as a recommendation to the particle to *"take shorter steps"* to convergence point ($v = 0$, $x = (\varphi_1 Pb + \varphi_2 Gb)/(\varphi_1 + \varphi_2)$). Note that v is the particle velocity, and it must be equal to zero in its convergence position. It must be highlighted that Clerc and Kennedy's velocity [see (12)] is just a special case of the original velocity expression [see (1)], where d_1, d_2, and ω are selected according to (20) and (13) [15], and that results can be easily extended to KASIA strategy

$$P(t+1) = \chi V(t+1) \oplus \chi P(t) \oplus (1-\chi)\frac{\varphi_1 P^{\#} \oplus \varphi_2 P^{*}}{\varphi_1 + \varphi_2}. \tag{22}$$

Hence, from this analysis, the fact that not only can KASIA explosion be controlled with no need for the specification of arbitrary or problem-specific parameters is shown, but in addition, the algorithm can be parametrized in such a way that the swarm converges consistently to local optima (with some special type of functions as a exception, convergence to global optima is not proven). In addition, the generalized model of KASIA allows users to control the convergence degree by configuring constriction parameter κ to various values. Note, however, that κ setting is not a requisite for convergence. On the basis of Kennedy and Eberhart's original work [31], φ_1 and φ_2 are typically fixed to 2.05 in a way that χ or ω are 0.729, and value for constants d_1 and d_2 is 0.729×2.05=1.494. However, there exist studies that suggest an adaptive fine tuning according to the problem under consideration [39].

IV. Simulation Results

A. Control of Inverted Pendulum

In this section, the classical problem in the fuzzy-logic controllers (FLCs) design [40], i.e., the inverted pendulum, is simulated to test the feasibility of the suggested rule-evolution strategy. First introduced by Barto *et al.* [26], this model presents a wide range of appropriate features for controller benchmarking. The inverted-pendulum system is an inherently unstable system (it is highly nonlinear when far from the equilibrium). It is extensively used to test control systems for a variety of uncertain nonlinear conventional systems [41], [42].

Basically, the inverted-pendulum system consists of a motor-driven cart with a vertical pole on its top [26], [43]. The cart is placed on a short rail track, and it is able to move in a single direction, i.e., toward its right or its left. Furthermore, the pole has a single freedom grade to rotate around its hinge pivot point. The main aim of the controller is to keep the pole balanced vertically and within a specified location as long as possible using impulses to move the cart. In addition, the cart cannot exceed the rail track boundaries. The inverted-pendulum system dynamics are driven by a series of parameters, such as acceleration due to gravity, pole length and mass, cart mass, rubbing the pole on the cart, and the cart on the track. Specifically, these equations are modeled as presented in [26].

In this study, the inverted pendulum is controlled by means of a fuzzy controller, which incorporates the acquired knowledge. Hence, it is initially necessary to choose the controller inputs. In the classical inverted-pendulum system, there exist four states, namely, x (POS), $\dot{x}$(VEL), θ(ANG), $\dot{\theta}$(WANG), and they are selected as the controller inputs. On the other hand, the inverted pendulum has only one input, the applied force F, which is used as the controller output. Precisely, in this study, every input is described by three Gaussian-shaped membership functions that represent *left, center*, and *right* positions for x and θ, as well as *low, medium*, and *high* values for $\dot{x}$ and $\dot{\theta}$. Output F considers nine fuzzy sets. Membership functions for inputs and output are presented in Fig. 1. Thus, every rule in the RB consists of four antecedents and a single consequence, with seven and 19 possible configurations, respectively (note the negation sets). Initially, the RBs must be randomly generated in such a way that no previous information is required. In addition, motion equations are assumed to be unknown.

With the aim of evaluating the KBs, an expert system based on fuzzy rules is considered [27]. The general structure of the evaluation system within the learning system is illustrated in Fig. 2. In this study, the typical structure for the FLC, i.e., fuzzification, inference, and defuzzification, is considered to evaluate the obtained knowledge quality. In the evaluation system [27], each input, namely, x (POS) and θ (ANG), is depicted by three triangle-shaped membership functions corresponding to left, center, and right. Output (EVAL), however, is represented by nine fuzzy sets. The evaluation system knowledge or RB is presented in [27]. The evaluation rule antecedents are

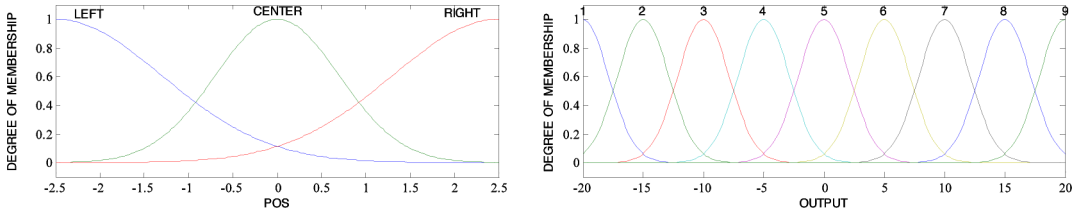

Fig. 1. FLC fuzzy sets. Input (POS) and output (OUTPUT).

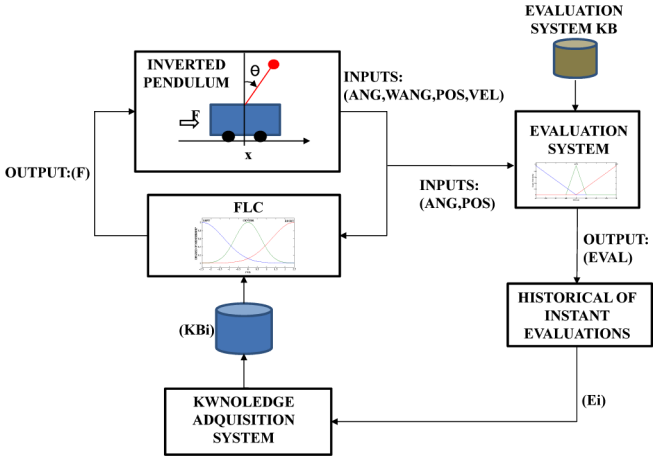

Fig. 2. Fuzzy control of the inverted pendulum with fuzzy-knowledge evaluation.

interpreted as the states of the controlled system states (in the evaluation system, the state is given by variables x and θ), and the rule consequence represents the system instantaneous evaluation (EVAL). Instant evaluations for a given time interval, i.e., N, are stored shaping a history of instantaneous values. Considering this history, only a partial evaluation of the control system is obtained as the system is only analyzed from a single initial state or starting point. Hence, to test the KB efficiently, specifying an initial condition range and obtaining partial evaluations are necessary. In this study, 16 random starting points, i.e., $n = 16$, within a four-dimension space, corresponding to the inverted-pendulum system inputs $(x, \theta, \dot{x}, \dot{\theta})$, are taken into account. In addition, the simulation step is set to 0.01 for 100 s for every starting point, i.e., $N = 10\,000$. Obtaining a numeric value in the evaluation of the KB of the control system is achieved by applying the arithmetic mean to all the histories n, establishing a penalty for those where the pole has fallen past the considered angle from the vertical position ($\theta_n < \theta < \theta_m$), or in case the cart has exceeded the track boundaries ($-x_n < x < x_m$) and a positive 80% weight for the successful ones $q_1 = 0.8$. In addition, a major 20% weight, i.e., so-called completeness $q_2 = 0.2$, is considered for those cases where the pole has always success ($\theta_n < \theta < \theta_m$ and $-x_n < x < x_m$) for each starting point [44]. In other words, if the system manages to keep the pole in a vertical position, i.e., the pole does not exceed its angle boundaries, and the cart does not exceed the previously defined track limits, the RB receives a complementary award. The final evaluation for each particle, E^i, is included in the interval [0,1]. A particle i evaluation can be mathematically formulated in the following way:

Algorithm 2 Evaluation of the acquired knowledge strategy.

Initialization. Evaluation of RB i.
1. Number of starting points for the inverted pendulum (n=16)
2. Random generation of n starting_points.
3. Timestep=0.01; Time interval=100s. N=Time step·Time_interval.

Do
1. Update inverted pendulum initial conditions.
2. Partial RB Evaluation j. One starting point for N time points. Eq. 24.
   if (angle and position satisfy boundaries)
   $O(j) = 1$
   else
   $O(j) = 0$
3. Store Partial RB Evaluation j.
   j++

While(j≤n)
if $O(j) = 1, \forall j,\ j \in [1, 2, \ldots, n]$
$\{C(i) = 1\}$
else
$\{C(i) = 0\}$
Global RB Evaluation:
Ei=Evaluate RB from the n partial evaluations. Eq. 23.

Return: *Global RB Evaluation* (Ei)

$$E^i = q_1 \cdot \frac{\sum_{j=1}^{n} \bar{E}_j^i \cdot O(j)}{n} + q_2 \cdot C(i) \tag{23}$$

where

$$\bar{E}_j^i = \frac{\sum_{k=1}^{N} E_j^i[k]}{N} \tag{24}$$

$$O(j) = \begin{cases} 1, & \text{if angle and position satisfy boundaries} \\ 0, & \text{in other case} \end{cases} \tag{25}$$

$$C(i) = \begin{cases} 1, & \text{if } O(j) = 1 \qquad \forall j,\ j \in [1, 2, \ldots, n] \\ 0, & \text{in other case.} \end{cases} \tag{26}$$

The evaluation process for a given RB is summarized in Algorithm 2, and the inverted pendulum and the evaluation system configuration in this study can be summarized as mass of the cart, $m_c = 1$ kg, mass of the pole, $m_p = 1$ kg, half-pole length $l = 0.5$ m, angle θ tolerance $-\pi/4$ rad $< \theta < \pi/4$ rad, position x boundaries -2.5 m $< x < 2.5$ m, and impulsive-driven force F interval -20 N $< F < 20$ N.

With regard to learning-strategy configuration, the population size for the Pittsburgh approach (PS) is set at 30, with selection rate $\lambda = 0.8$, elitism, two-point crossover and initial maximum size for RB, and initmaxRB$_{\text{size}}$, -20 rules-.

On the other hand, 24 particles or RBs are considered at each iteration (i.e., NP $= 24$) in the PSO-based strategy with RB sizes (RB$_{\text{size}}$), -10 rules-, inertia weight $\omega = 0.7$, and weight factors $d_1 = 2$ and $d_2 = 3$. This way, the genetic

TABLE I
LEARNING STRATEGIES RESULTS FOR 30 SIMULATIONS (TRAINING FITNESS E)

Results	Max	Min	Average	Standard Deviation	Confidence Interval (95%)
KASIA	0.9357	0.6638	0.7949	0.0984	0.7597, 0.8301
GA-Pittsburgh	0.9147	0.6909	0.7582	0.0893	0.7262, 0.7902
SARSA	0.9069	0.2185	0.6249	0.2060	0.5512, 0.6986

population and the swarm-convergence behavior can be compared with every generation (except starting iteration) with the same number of RB evaluations or FEs and, therefore, require the same computational effort. Note that the Pittsburgh-learning strategy evaluates the entire population in the first generation, and then, it evaluates a number of individuals regarding selection rate λ. Thus, total computational effort for every simulation for the Pittsburgh approach can be formulated as $\mathrm{CE}_{\mathrm{Pitts}} = \mathrm{PS} + \mathrm{PS} * \lambda * (\mathrm{num}_{\mathrm{iter}} - 1)$, where PS denotes the number of individuals or RB population size. Hence, total computational effort for $\mathrm{num}_{\mathrm{iter}} = 500$ iterations is 12 006 FEs and 12 000 FEs for Pittsburgh and KASIA, respectively, where the difference is due to initial iteration (i.e., 30 versus 24 FE). Moreover, the Pittsburgh approach is configured with a mutation operator following an exponential expression. It is known that mutation is employed in GAs to avoid local maximums, and considering a mutation that decreases through iterations, it allows an increase of the maximum location exploration when the algorithm is close to the stopping condition. In addition, as regard KASIA convergence control or constriction factor χ setting, random factors r_1 and r_2 with an exponential decrease are considered in such a way that it favors a wider search-space exploration in initial swarm evaluations, and a thorough examination of the locations close to the achieved maximum $r(\mathrm{iter}) = \mathrm{Uniform}(0,\ r_0 \cdot e^{(-\mathrm{iter}/\mathrm{num}_{\mathrm{iter}})c})$, where r represents random factors r_1 and r_2, r_0 is the maximum value for r (set to 1 in this simulation), and c is a constant fixed at 5 in this simulation.

In addition, the suggested learning strategy is compared with another well-known unsupervised learning method. Specifically, the RL strategy, i.e., SARSA, is taken into account [24]. Within this approach, the inverted-pendulum possible conditions are discretized in a set of states s; each one has been characterized by a range of values for the pole position x, angle θ, velocity $\dot{x}$, and angular velocity $\dot{\theta}$. In addition, an action a (i.e., impulsive-driven force F) is associated with every state, and the goal is to find the best-suited set of actions controlling the inverted pendulum. With this aim, a reinforcement process is conducted through a set of episodes, where state-related actions are evaluated and rewarded, as depicted in reinforcement matrix $Q(s, a)$. Moreover, episodes are structured in evaluation subepisodes called steps. Note that there exist no direct association between SARSA evaluation stages, episodes, and Pittsburgh and KASIA iterations. Thus, it is not possible to provide a step-by-step convergence behavior comparison if the same computational effort is borne in mind. However, a fair comparison can be made in terms of final evaluation. To be precise, in this study, 12 000 episodes are considered for learning with SARSA in such a way that total computational effort for

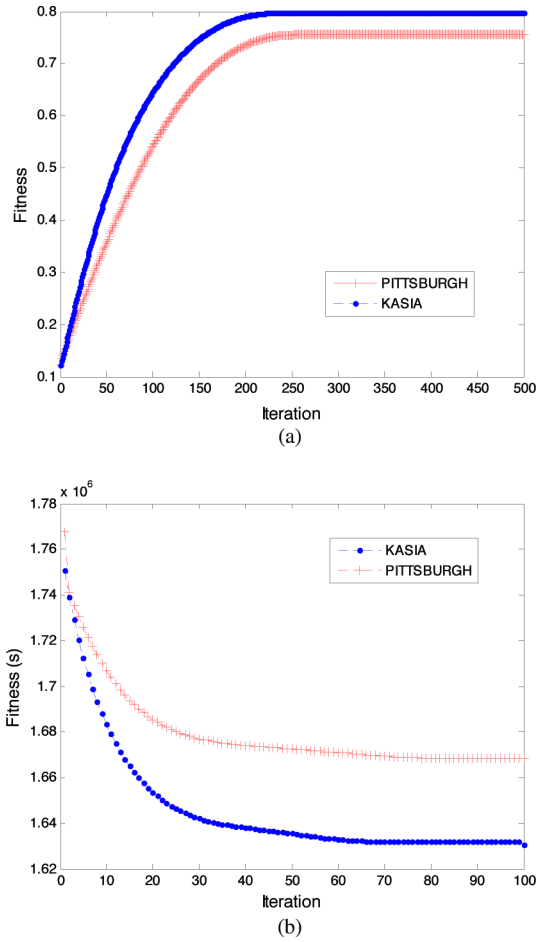

Fig. 3. KASIA and Pittsburgh-learning convergence behavior for inverted-pendulum control and grid-computing scheduling. (a) Fitness (E) convergence in inverted pendulum control. (b) Fitness (makespan) convergence in grid scenario.

KASIA is more reduced than the one for this strategy or equal to it in every simulation.

Table I presents statistical results for the three referred learning strategies. Specifically, results are analyzed in terms of the best (Max), the worst (Min), and average solutions (Average); standard deviation; and 95% confidence interval. The fact that KASIA outperforms GAs-based and SARSA strategies on average by 4.62% and 21.38%, respectively, is illustrated. Furthermore, KASIA best evaluation (Max) improves the other strategies by 2.24% and 3.08%. However, it is also observed that both KASIA standard deviation and Min solution are less accurate than Pittsburgh ones. As far as convergence behavior is concerned, the best solution considering 30 simulations for the KASIA and Pittsburgh approaches is depicted in Fig. 3(a). The fact that the FLC, which considers the KASIA rule-evolution

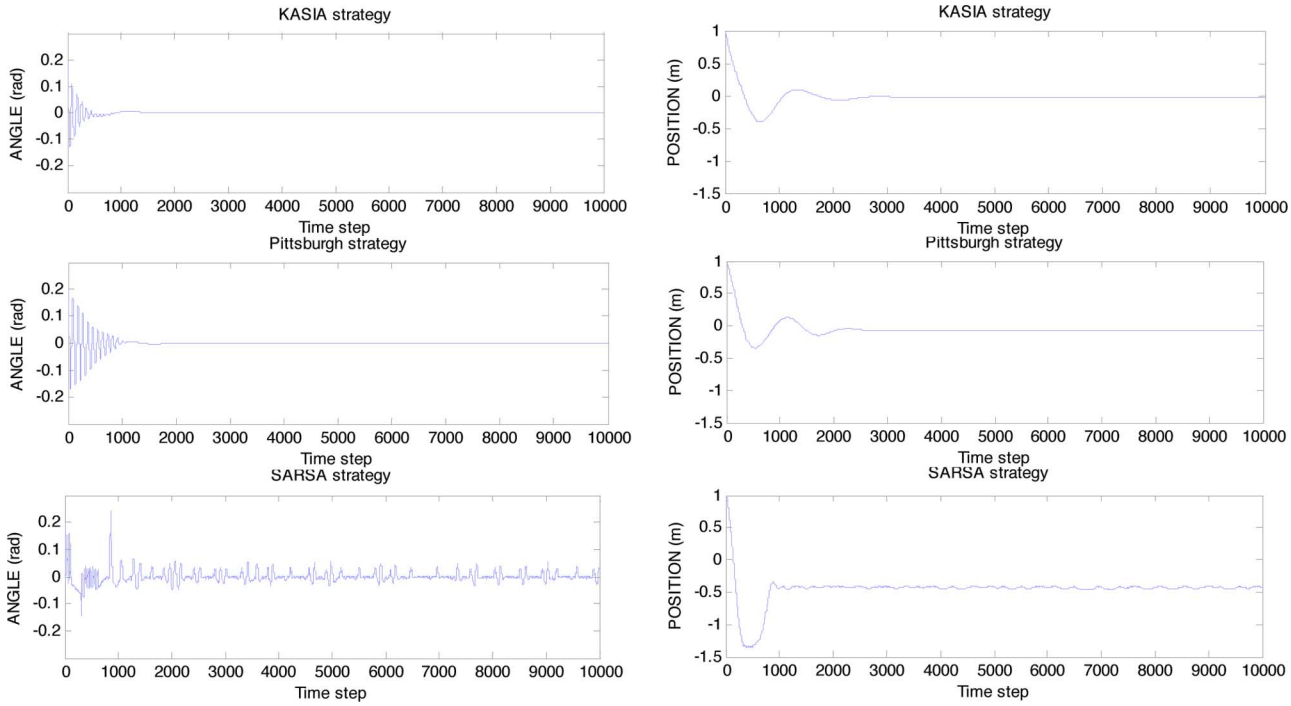

Fig. 4. Inverted-pendulum control with FLC for 100 s. Starting point: ANG = +0.20 rad, WANG = +0.02 rad/s, POS = +1.00 m, and VEL = −0.30 m/s.

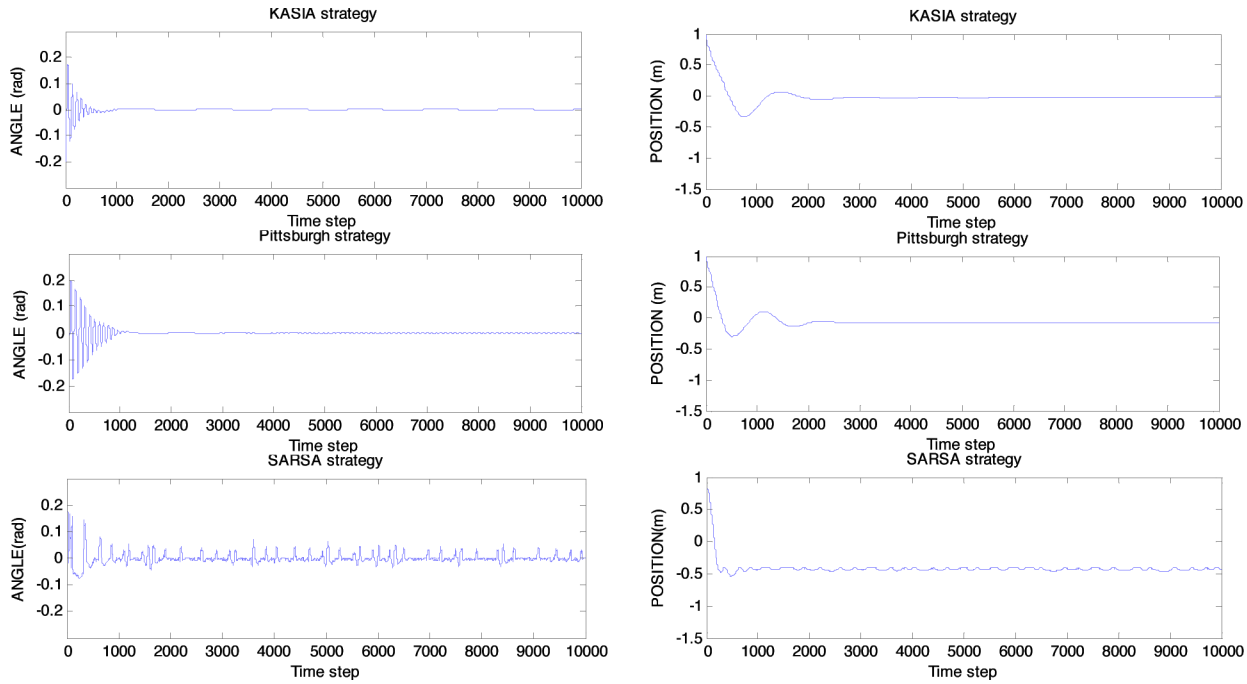

Fig. 5. Inverted-pendulum control with FLC for 100 s. Starting point: ANG = −0.20 rad, WANG = +0.02 rad/s, POS = +1.00 m, and VEL = −0.30 m/s.

strategy, outperforms the one with Pittsburgh approach (i.e., 4.62%) after 500 generations is shown. Moreover, the proposed strategy convergence behavior improves the one presented by the genetic approach through nearly all the simulation in such a way that its associated fitness is higher than the Pittsburgh approach from the tenth generation to the stopping iteration. In addition, as it has been shown that the PSO-based strategy achieves a convergence 98% close to its final result in 20 less generations in comparison with the Pittsburgh-based one, something that means a more reduced computational effort.

To present the acquired knowledge performance within an FLC graphically, Figs. 4 and 5 are introduced, which illustrate the position and angle evolution for a two-initial-condition set for KASIA, Pittsburgh, and SARSA best knowledge. To be precise, the best knowledge found for these strategies are tested for two different starting points. Evolution is presented for a

whole evaluation time period, i.e., $N = 10\,000$ time points. As expected, figures show that the considered strategies are able to control the inverted-pendulum system, i.e., the cart remains inside its track, and the pole angle does not exceed the considered boundaries for simulation time. Moreover, cart position is close to zero for the fuzzy strategies. Nonetheless, it is to be noted that the obtained RB through KASIA generally reaches equilibrium faster than Pittsburgh and SARSA. As shown in Figs. 4 and 5, the Pittsburgh approach through RB presents more variations in the pole angle before the system simulation ends. Furthermore, the cart-position variations present a wider oscillation amplitude. In addition, one can observe that SARSA angle and position show oscillations that remain through the considered 10000 time steps.

B. Scheduling in Grid Computing

Inverted-pendulum simulations provide an analysis to test the proposed learning-strategy feasibility. Despite this, with the aim of testing KASIA in a problem of practical importance nowadays, its application to the construction of FRBS-based metaschedulers for grid computing is presented. Grid computing is an emerging computational framework, providing high-end computational capabilities for the solution to high-performance and massive computation applications in science and engineering, which has previously been limited by local resources computer hardware [28]. Essentially, a grid consists of a large-scale heterogeneous computational resource set, which is geographically distributed and interconnected through high-speed networks. These resources join together to make up a high-performance virtual infrastructure. An efficient resource coordination and cooperation is crucial to harness grid potential, and in this sense, a major challenge is given by scheduling, which is a problem that has proven to be NP-complete in its general form [45], [46].

There exists a variety of scheduling strategies for grids. *Queue-based* scheduling strategies are founded on diverse jobs queue management policies, and they integrate techniques such as *EASY-backfilling (EASY-BF), first come first serve (FCFS),* and *earliest deadline first-BF (EDF-BF)* [29], [47]. These strategies are successful in meeting simple performance objectives, and they are consolidated as de facto standard in grids nowadays. In fact, most present-day production systems, such as Condor [48] or Portable Batch System (PBS) [49] and other grid metascheduling systems, like Grid Service Broker [50], are based on *queue-based* scheduling strategies. However, the increasing demand for scheduling strategies handling complex quality-of-service (QoS) performance criteria, including *flowtime*, *slowdown*, *tardiness,* or *resource usage*, has driven research on alternative strategies, such as adaptive scheduling [45]. As stated in [51], a more or less precise information of the grid state must be taking into account by any scheduling strategy trying to offer a certain level of QoS. An adaptive scheduling suggests preventing performance degradation by considering both current and future grid states at every schedule. In this regard, *schedule-based* strategies such as *ESG Local Search periodical* [29], [47] found their schedules on a "known" grid actual state (i.e., resource capabilities or jobs' computational demand) with the aim of satisfying QoS requirements. Nevertheless, a grid is a*fully dynamic environment with uncertainties* [51], and thus, a precise grid-state characterization is not feasible due to system inherent vagueness and imprecision. This way, new trends are emerging toward the development of reasonable scheduling techniques that tolerate noisy or imprecise dynamic systems information, such as the one in grid networks. In this area, it is relevant to point out the role of FRBSs. They are proving to be an efficient alternative for scheduling [4], [52] in computational distributed environments. However, as it can be inferred, the scheduling performance that use FRBSs strongly depends on the quality of its knowledge and, thus, with the learning strategy. This study suggests the incorporation of KASIA as a learning strategy of a fuzzy-rule-based metascheduler for computational grids.

A grid metascheduler is responsible for allocating L users diverse jobs $J = \{J_1, J_2, \ldots, J_L\}$ to the participating resource domains RD within a grid virtual organization $\mathrm{VO} = \{\mathrm{RD}_1, \mathrm{RD}_2, \ldots, \mathrm{RD}_G\}$. Each RD_j consists of a set of H_j heterogeneous computational machines $\mathrm{RD}_j = \{r_{j,1}, r_{j,2}, \ldots, r_{j,H_j}\}$ and a local scheduler managing access policies and scheduling within its domain. Note that resources are dynamic and therefore may be added or withdrawn in ongoing time. Furthermore, resource performance may be altered due to changes in its RD access policies or local load levels. The FRBS-based metascheduler performance is based on the characterization of every RD state through a limited set of features. This way, the metascheduler obtains the value of these features at every schedule step from the grid environment, and then, it applies a reasoning strategy to decide the most efficient schedule. Specifically, the reasoning strategy considers each feature inherent vagueness for every RD, and it provides a performance index y_o indicating the suitability level for being the selected RD in the actual schedule according to the overall fuzzy RD state characterization. In this study, seven variables for describing an RD state are suggested, as shown in Table II. It should be mentioned here that more features may be required to obtain a more exhaustive grid-state characterization. In spite of this, bearing in mind the fact that RB search space highly increases according to the number of features, this choice is taken to achieve a compromise between RD state characterization and learning processes effort.

Fig. 6 shows a fuzzy metascheduler general structure. It is derived from Mamdani fuzzy logic systems [1] where there exist four main components: *fuzzification system*, *inference system, defuzzification system,* and *KB*. First, *fuzzification system* associates each RD crisp input feature to a fuzzy linguistic variable. To be precise, it must be stated here that fuzzy sets for the proposed fuzzy metascheduler are described by means of Gaussian membership functions, as it happened in the inverted-pendulum problem, with three and five membership functions for input and output features, respectively. Then, in the *inference system,* a fuzzy selector factor is obtained by means of the system knowledge or rule application to finally derive the RD selector factor y_o in the *defuzzification system*. Note y_o is obtained for every resource domain RD_j considering its state $(\mathrm{FPE}_j, \mathrm{PT}_j, \mathrm{RM}_j, \mathrm{RT}_j, \mathrm{PS}_j, \mathrm{RS}_j, \mathrm{RE}_j)$ at every schedule.

TABLE II
INPUT FEATURES FOR THE FUZZY METASCHEDULER

Feature	Description
Number of free processing elements (FPE)	Number of free processing element within RD_i.
Previous Tardiness (PT)	Sum of tardiness of all finished jobs in RD_i.
Resource Makespan (RM)	Current makespan for RD_i.
Resource Tardiness (RT)	Current tardiness of jobs within RD_i.
Previous Score (PS)	Previous deadline score of already finished jobs in RD_i.
Resource Score (RS)	Number of non delayed jobs so far in RD_i.
Resources In Execution (RE)	Number of Resources currently executing jobs within RD_i.

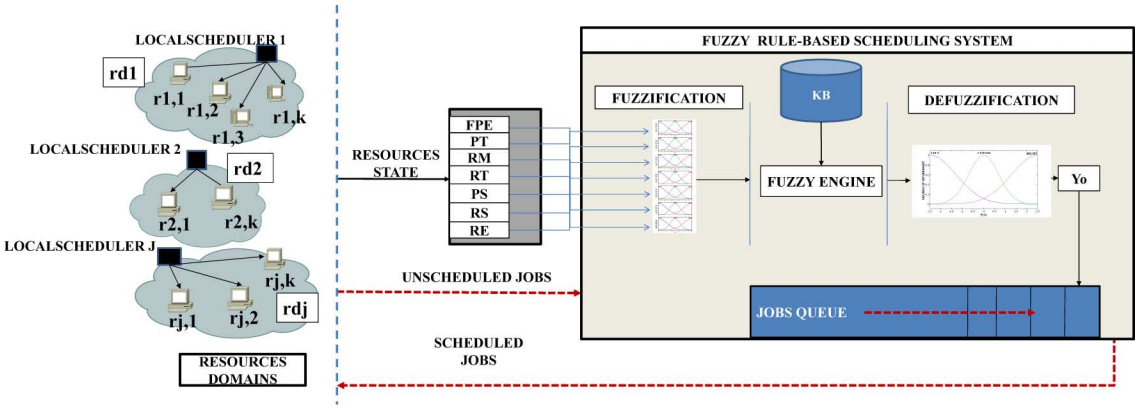

Fig. 6. Fuzzy metascheduling system.

TABLE III
SIMULATION RESULTS FOR 30 SIMULATIONS IN GRID METACENTRUM (TRAINING FITNESS MAKESPAN (*s*))

Results	Max	Min	Average	Standard Deviation	Confidence Interval (95%)
KASIA	1654505.4	1553820.1	1630479.5	36778.7	1617318.41, 1643640.50
GA-Pittsburgh	1684235.5	1625058.2	1667586.2	19757.8	1660515.96, 1674656.43

The fuzzy metascheduler with KASIA learning is evaluated through simulations with Alea software [47], which is a simulation toolkit that allows the usage of grid scenarios and traces from existing installations. In this paper's tests, the grid system is based on a Czech National Grid Infrastructure Metacentrum project [53], whose aim is to support the development of a distributed high-performance computing infrastructure through joint cooperation of resources from a set of academic and research worldwide institutions. In the simulations, the grid system is made up of 14 Metacentrum nodes. The grid includes 806 CPUs of diverse types (i.e., Opteron and Xeon) and speed (i.e., 1500–3200 MHz) located in 210 machines with heterogeneous RAM size (i.e., 1 005 000–27 343 000 kB) running Linux. Moreover, machines queue setting, machines and reservation behavior and jobs traces are obtained from Metacentrum facilities, retrieved from January to May 2009 (available at http://www.fi.muni.cz/∼xklusac/index.php?page=meta2009 [53]).

As far as the learning process is concerned, *makespan* is suggested as performance index or fitness. Hence, the minimization of the latest job finalization time or *makespan* is pursued in this stage [45]. Training is carried out for 100 iterations where the fuzzy scheduler has to allocate 2000 jobs in the Metacentrum scenario. Moreover, the KASIA learning strategy is compared with the Pittsburgh approach for the evolution of fuzzy rules. In other words, FRBS-based scheduler in grid scenario is tested, taking into account two learning strategies emanating from the same grid knowledge and knowledge encoding (fuzzy rules). KASIA is configured with $\omega = 0.9, d_1 = 2, d_2 = 2$, and $\text{RB}_{\text{size}} = 10$, whereas Pittsburgh considers selection rate, $\lambda = 0.9$, elitism, two-point crossover, and initmax$\text{RB}_{\text{size}} = 20$. Furthermore, both sizes of swarm and genetic population are fixed to allow a fair comparison at every iteration in FEs, NP $= 18$, and $PS = 20$. Fig. 3(b) represents learning evolution for KASIA and Pittsburgh strategies. Curves illustrate the best RB associated fitness at every iteration (i.e., 30 simulations on average). It is observed that KASIA initial convergence behavior presents no meaningful difference at early iterations (approximately iteration 10) in comparison with Pittsburgh. However, from this point on, the fact that KASIA achieves a faster convergence is appreciated. Note that Pittsburgh approach requires 78 iterations to reach its final fitness, in contrast with KASIA, which converges in 64 iterations. Thus, KASIA obtains its convergence in 14 iterations fewer than the genetic strategy, which is translated into a computational effort reduction of 252 FEs. Moreover, KASIA convergence fitness improves Pittsburgh in 2.23%, as presented in Table III. In addition, the best RB found in KASIA learning outperforms the best genetic RB fitness (Min) by 4.38%, with something that proves KASIA ability to achieve a deeper search-space exploration. Hence, KASIA learning provides a faster and more accurate learning that Pittsburgh approach.

TABLE IV
SCHEDULING STRATEGIES RESULTS FOR KASIA, PITTSBURGH, QUEUE-BASED STRATEGY EASY-BF, AND SCHEDULE-BASED STRATEGY ESG + LS PERIODICAL

Metric/Strategy	Fuzzy-KASIA	Fuzzy-Pittsburgh	EASY-BF	ESG+LS periodical
Makespan (s)	1633719.8	1659926.3	1749586.0	1973151.4
Flow-time (s)	87765.539	88865.534	87491.471	83379.182
Weighted usage (%)	48.184	46.420	44.471	34.744
Classic usage (%)	59.360	56.528	47.013	40.914
Tardiness (s)	4683.250	4757.286	3235.311	1274.822
Slowdown (s)	197.036	192.049	184.352	17.522

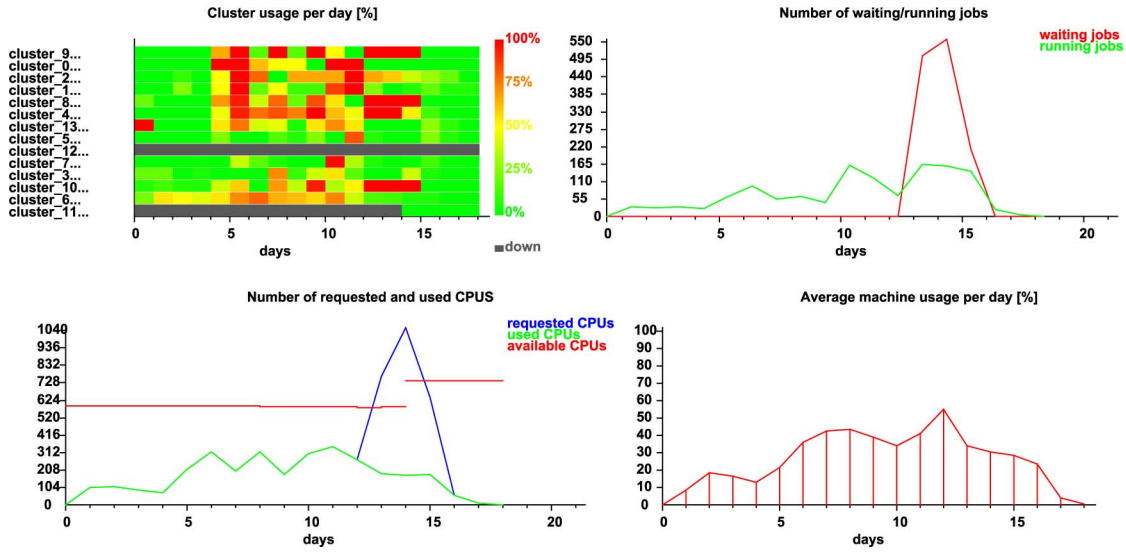

Fig. 7. Fuzzy KASIA results.

With the aim of getting further analyses about KASIA performance and validate results, the fuzzy meta-scheduler is tested in several QoS criteria, namely *flow time, weighted usage, classic usage, tardiness, and slowdown.* Moreover, Metacentrum scenario is modified with alternative machine failure and reservation behavior, and load is increased by 16.67% (i.e., 2400 jobs). Table IV presents average results (i.e., 30 simulations) for the fuzzy scheduler in the new scenario with KASIA and Pittsburgh. Similar to the previous scenario, it is shown that the fuzzy scheduler with KASIA improves *makespan* results of Pittsburgh by 1.58%. This could be expected if one considers the fact that the learning training has been conducted through *makespan* and, as presented earlier, KASIA achieves a more accurate result in this criterion. Furthermore, it is observed that a decrease in *makespan* is accompanied by an increase in *machine usage*. Thus, fuzzy scheduler with KASIA outperforms the genetic scheduler in terms of *classical* and *weighted usage* by 4.77% and 3.66%, respectively. However, no relevant difference is appreciated in metrics such as *flow time*, *slowdown,* or *tardiness.* Note that these metrics are not learning-process objectives, and they present conflicting interests with selected fitness. In addition, results are compared with two widely extended scheduling strategies in current grid systems: *EASY-BF* queue-based strategy and *ESG+LS periodical* schedule-based strategy [29], [47]. Table IV presents these strategy results under the same grid conditions.

It is shown that KASIA metascheduler achieves the best performance in terms of *makespan* in comparison with the rest of strategies. To be precise, it improves *makespan, EASY-BF* by 6.62% and outperforms *ESG+LS periodical* strategy by 17.20%. Hence, it can be stated that KASIA scheduler succeeds in finding the highest quality RB in terms of fitness index, and this quality is kept in modified conditions, as validation results prove. In addition, an improvement in *weighted* and *classical usage* is observed (7.71% and 20.81% in comparison with its best competitor, i.e., *EASY-BF*, respectively). Despite this, as expected, *makespan* optimization causes a deterioration in *flow time*, *slowdown,* or *tardiness*.

Finally, the scheduling-process evolution is graphically analyzed through the whole simulation. Figs. 7–9 illustrate progress in *cluster usage*, *number of waiting and running jobs*, *number of request and used CPU,* and *average machine usage per day* for fuzzy scheduling with KASIA learning, *EASY-BF*, and *ESG+LS periodical*, respectively. Concerning *cluster usage,* it is shown that KASIA scheduler achieves a higher distributed assignment for jobs within RDs. As presented in Fig. 7, all clusters participate in the simulation at any moment in contrast with *EASY-BF* (see Fig. 8), where clusters 13 and 5 do not intervene in any schedule, and load is concentrated on the rest of clusters. In addition, KASIA scheduler proves to have a better performance in more demanding conditions. This can be analyzed in days from 8 to 11, where both *EASY-BF* and *ESG+LS periodical* waiting jobs increase at the advent of higher load (see Figs. 8 and 9), in contrast with KASIA scheduler. Furthermore, as shown in Fig. 7, the number of CPUs used satisfies the requested number of CPUs in the fuzzy strategy until workload

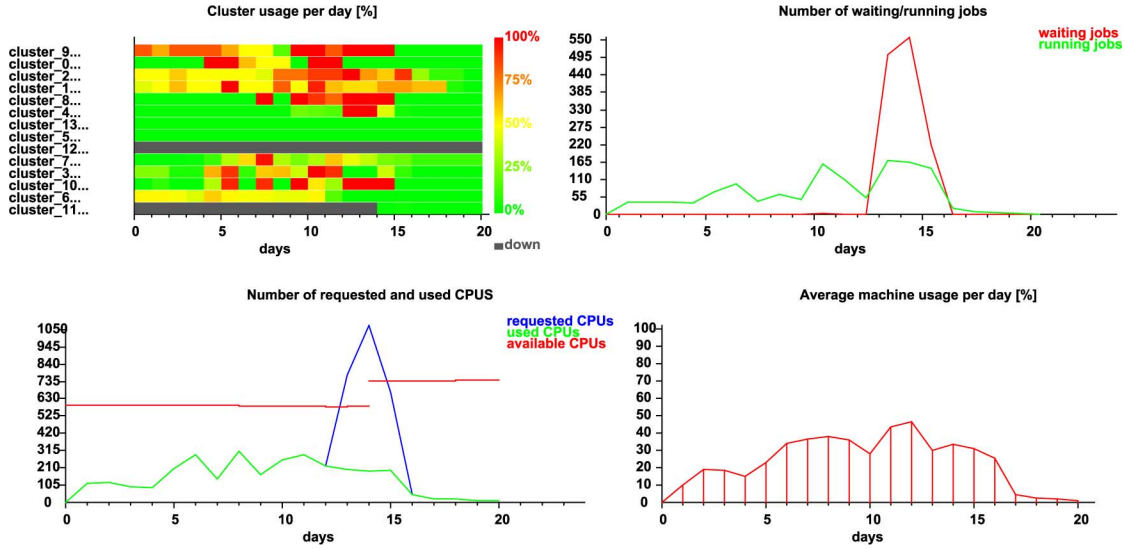

Fig. 8. EASY-BF results.

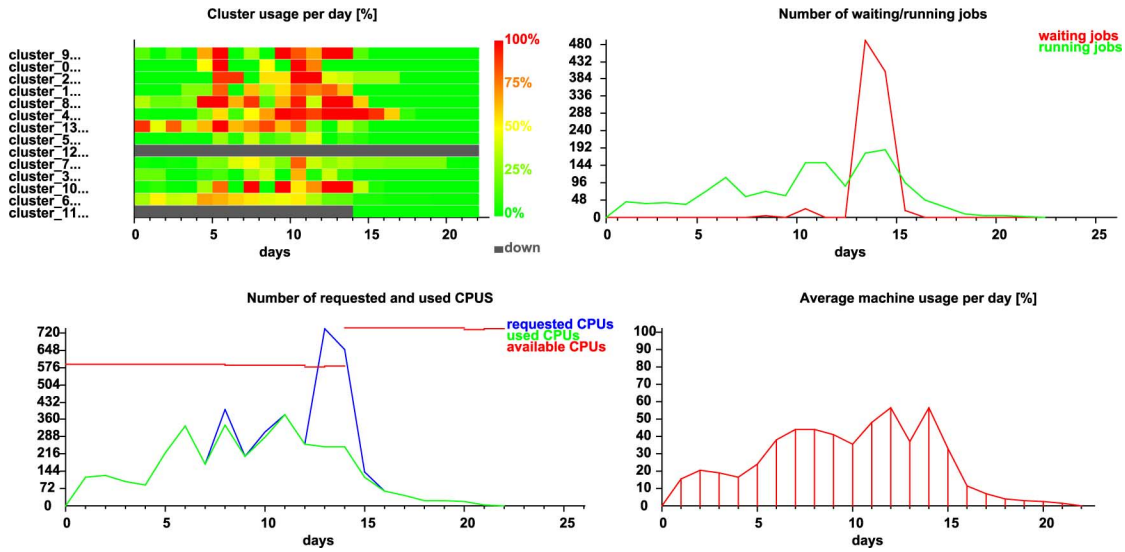

Fig. 9. EGS-LS periodical results.

peak appears (i.e., days 1–13), in contrast to *ESG+LS periodical,* where CPUs request outranges available resources not even in the most demanding conditions, as observed in Fig. 9. Regarding *machine utilization per day*, it is shown that the fuzzy KASIA scheduler converges to its maximum faster than *EASY-BF* and *ESG+LS periodical* when workload peak appears (i.e., days 13–15), which is something that proves KASIA scheduler adaptability to changes in grid conditions.

V. Conclusion

Automatically incorporating knowledge is a relevant task in expert system design. Specifically, this study deals with high-quality RB acquisition in FRBSs. GAs-based strategies for rule evolution have been extensively used for this purpose. Nevertheless, the growing importance of FRBSs drives more efficient learning strategy development. In this study, a new methodology for knowledge acquisition has been suggested. This methodology is based on the swarm intelligence algorithm PSO. PSO has been applied to many areas successfully and has proven to be an efficient global optimization algorithm. However, its application to knowledge acquisition in FRBSs is innovative, and a novel strategy to its adaptation to Mamdani-rule evolution has been presented, i.e., KASIA. Simulation results considering the classical benchmark problem, i.e., the inverted pendulum, show the fact that the proposed rule-evolution strategy is feasible and outperforms, in terms of final average evaluation, other classical RL strategies, the Pittsburgh approach, and SARSA by 4.62% and 21.38%, respectively. On the other hand, KASIA is tested as a learning strategy for fuzzy-rule-based metaschedulers for grid computing, and its performance is compared with some of the most extended scheduling strategies: the *EASY-BF* queue-based strategy and the *ESG+LS periodical* schedule-based strategy. To be precise, it outperforms *makespan* by 6.62% on average

with respect to best competitor, i.e., *EASY-BF*. In addition, it is shown that KASIA learning provides higher quality RBs in terms of training fitness in comparison with genetic-evolution strategies (2.23%). Furthermore, KASIA converges faster to its final solution and reduces the required computational effort (in the number of FEs) in considered scenarios. Therefore, simulation results, together with its simplicity and capability to control convergence make it a promising strategy for knowledge acquisition in expert systems.

ACKNOWLEDGMENT

The Metacentrum workload log was generously provided by the Czech National Grid Infrastructure Metacentrum.

REFERENCES

[1] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: World Scientific, 2001.

[2] J. Exposito, S. Galan, N. Reyes, and P. Candeas, "Audio coding improvement using evolutionary speech/music discrimination," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, 2007, pp. 1–6.

[3] R. Alcalá, J. Casillas, O. Cordón, A. González, and F. Herrera, "A genetic rule weighting and selection process for fuzzy control of heating, ventilating and air conditioning systems," *Eng. Appl. Artif. Intell.*, vol. 18, no. 3, pp. 279–296, 2005.

[4] C. Franke, F. Hoffmann, J. Lepping, and U. Schwiegelshohn, "Development of scheduling strategies with genetic fuzzy systems," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 706–721, 2008.

[5] R. Yager, "Perception-based granular probabilities in risk modeling and decision making," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 2, pp. 329–339, Apr. 2006.

[6] S. Alonso, E. Herrera-Viedma, F. Chiclana, and F. Herrera, "Individual and social strategies to deal with ignorance situations in multi-person decision making," *Int. J. Inf. Technol. Decis. Making*, vol. 8, no. 2, pp. 313–333, 2009.

[7] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Comput.*, vol. 13, no. 10, pp. 959–977, 2009.

[8] G. Leng, T. McGinnity, and G. Prasad, "Design for self-organizing fuzzy neural networks based on genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 755–766, Dec. 2006.

[9] R. Alcalá, P. Ducange, F. Herrera, B. Lazzerini, and F. Marcelloni, "A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1106–1122, Oct. 2009.

[10] S. Chung, F. T. Chan, and H. Chan, "A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling," *Eng. Appl. Artif. Intell.*, vol. 22, no. 7, pp. 1005–1014, 2009.

[11] X. Shi, Y. Liang, H. Lee, C. Lu, and L. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Inf. Process. Lett.*, vol. 93, no. 5, pp. 255–261, 2005.

[12] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, Pittsburgh Univ., Pittsburgh, PA, 1980.

[13] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artif. Intell.*, vol. 40, no. 1–3, pp. 235–282, 1989.

[14] P. R. López, F. Jurado, N. R. Reyes, S. G. Galán, and M. Gómez, "Particle swarm optimization for biomass-fuelled systems with technical constraints," *Eng. Appl. Artif. Intell.*, vol. 21, no. 8, pp. 1389–1396, 2008.

[15] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 52, no. 2, pp. 397–407, Feb. 2004.

[16] Y. Shi, R. Eberhart, E. Center, and I. Carmel, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput.*, vol. 3, 1999, pp. 1945–1950.

[17] C.-J. Huang, Y.-T. Chuang, and K.-W. Hu, "Using particle swam optimization for qos in ad-hoc multicast," *Eng. Appl. Artif. Intell.*, vol. 22, no. 8, pp. 1188–1193, 2009.

[18] M. Meissner, M. Schmuker, and G. Schneider, "Optimized particle swarm optimization (OPSO) and its application to artificial neural network training," *BMC Bioinform.*, vol. 7, no. 1, p. 125, 2006.

[19] A. Chatterjee and F. Matsuno, "A neuro-fuzzy assisted extended Kalman filter-based approach for simultaneous localization and mapping (slam) problems," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 5, pp. 984–997, Oct. 2007.

[20] E. Araujo and L. dos Santos Coelho, "Particle swarm approaches using lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system," *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1354–1364, 2008.

[21] L. dos Santos Coelho and B. Herrera, "Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3234–3245, Dec. 2007.

[22] G. K. Venayagamoorthy, L. L. Grant, and S. Doctor, "Collective robotic search using hybrid techniques: Fuzzy logic and swarm intelligence inspired by nature," *Eng. Appl. Artif. Intell.*, vol. 22, no. 3, pp. 431–441, 2009.

[23] T. Hussein, A. Elshafei, and A. Bahgat, "Comparison between multi-band and self-tuned fuzzy power system stabilizers," in *Proc. 16th Mediterr. Conf. Control Autom.*, 2008, pp. 374–379.

[24] G. A. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[25] G. Faria and R. Romero, "Incorporating fuzzy logic to reinforcement learning [mobile robot navigation]," in *Proc. 9th IEEE Int. Conf. Fuzzy Syst.*, 2000, vol. 2, pp. 847–852.

[26] A. Barto, R. Sutton, and C. Anderson, "Neuron-like adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983.

[27] J. Bago and L. Magdalena, "Knowledge based evaluation of knowledge bases," presented at the EUSFLAT-ESTYLF-Joint Conf., Palma de Mallorca, Spain, 1999.

[28] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann, 2003.

[29] D. Klusacek, H. Rudova, R. Baraglia, M. Pasquali, and G. Capannini, "Comparison of multi-criteria scheduling techniques," in *Grid Computing: Achievements and Prospects*. Berlin, Germany: Springer, pp. 173–184, 2008.

[30] M. Drobics and J. Himmelbauer, "Creating comprehensible regression models: Inductive learning and optimization of fuzzy regression trees using comprehensible fuzzy predicates," *Soft Comput.*, vol. 11, no. 5, pp. 421–438, 2007.

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[32] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, 2007, pp. 120–834.

[33] Z.-H. Zhan, J. Zhang, Y. Li, and H.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[34] H. Zhu, C. Pu, K. Eguchi, and J. Gu, "Euclidean particle swarm optimization," in *Proc. 2nd Int. Conf. Intell. Netw. Intell. Syst.*, 2009, pp. 669–672.

[35] D. P. Tian and N. Q. Li, "Fuzzy particle swarm optimization algorithm," in *Proc. Int. Joint Conf. Artif. Intell.*, 2009, pp. 263–267.

[36] A. Abraham, H. Guo, and H. Liu, "Swarm intelligence: foundations, perspectives and applications," *Comput. Intell. (SCI)*, vol. 26, pp. 3–25, 2006.

[37] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proc. Workshop Particle Swarm Optim.* vol. 1, Indianapolis, IN: Purdue School Eng. Technol., IUPUI, 2001, pp. 1–6.

[38] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[39] H. Liu, A. Abraham, and A. E. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," *Future Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1336–1343, 2010.

[40] K. Belarbi, F. Titel, W. Bourebia, and K. Benmahammed, "Design of Mamdani fuzzy logic controllers with rule base minimisation using genetic algorithm," *Eng. Appl. Artif. Intell.*, vol. 18, no. 7, pp. 875–880, 2005.

[41] F. Hoffmann, D. Schauten, and S. Holemann, "Incremental evolutionary design of TSK fuzzy controllers," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 4, pp. 563–577, Aug. 2007.

[42] R. Shahnazi and M. R. Akbarzadeh-Totonchi, "Pi adaptive fuzzy control with large and fast disturbance rejection for a class of uncertain nonlinear systems.," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 187–197, Feb. 2008.

[43] N. Seifipour and M. B. Menhaj, "A new Ga-based real time controller for the classical cart-pole balancing problem," in *Computational Intelligence: Theory and Applications,* ser. Advances in Soft Computing, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, vol. 33, pp. 771–786.
[44] L. Magdalena and F. Monasterio, "Estudio de la coordinación inteligente en robots bípedos." Ph.D. dissertation, Madrid Technical Univ., Madrid, Spain, 1994, 2010.
[45] F. Xhafa and A. Abraham, "Meta-heuristics for grid scheduling problems," *Metaheuristics for Scheduling: Distributed Computing Environments,* Studies in Computational Intelligence, Berlin, Germany: Springer Verlag, vol. 146, pp. 1–37, 2008.
[46] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*., New York: Freeman, 1979.
[47] D. Klusacek, L. Matyska, and H. Rudova, "Alea—Grid scheduling simulation environment," in *Parallel Processing and Applied Mathematics* (Lecture Notes in Computer Science Series 4967), 2008, Proceedings Paper, pp. 1029–1038.
[48] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The Condor experience," *Concurrency Comput. Pract. Exp.*, vol. 17, no. 2-4, pp. 323–356, 2005.
[49] B. Nitzberg, J. Schopf, and J. Jones, "PBS Pro: Grid computing and scheduling attributes," *Int. Ser. Oper. Res. Manag. Sci.*, pp. 183–192, 2003.
[50] S. Venugopal, R. Buyya, and L. Winton, "A grid service broker for scheduling distributed data-oriented applications on global grids," in *Proc. 2nd Workshop Middleware Grid Comput.*, New York: ACM, 2004, pp. 75–80.
[51] D. Klusacek, "Dealing with uncertainties in grids through the event-based scheduling approach," in *Proc. 4th Doctoral Workshop Math. Eng. Methods Comput. Sci.*, 2008, vol. 1, pp. 978–980.
[52] R. P. Prado, S. G. Galán, A. J. Yuste, J. E. M. Expósito, A. J. S. Santiago, and S. Bruque, "Evolutionary fuzzy scheduler for grid computing," in *Lecture Notes in Computer Science*, vol. 5517, Berlin, Germany: Springer, 2009, pp. 286–293.
[53] C. N. G. Infrastructure. (2009). Metacentrum data sets [Online]. Available: http://www.fi.muni.cz/∼xklusac/index.php?page= meta2009

R. P. Prado received the M.S. degree in telecommunication engineering from Seville University, Seville, Spain, in 2008.

She is currently a Researcher with the Telecommunication Engineering Department, Jaén University, Jaén, Spain. She is currently engaged in artificial intelligence applied to scheduling in computational grids. Her research interests include highly distributed computing-resource management and scheduling.

S. García-Galán received the M.S. degree in telecommunication engineering from Malaga University, Malaga, Spain, in 1995 and the Ph.D. degree in telecommunication engineering from Madrid Technical University, Madrid, Spain, in 2004.

Since 1999, he has been an Associate Professor with the Telecommunication Engineering Department, Jaén University, Jaén, Spain. His current research interests include engineering applications, artificial intelligence, grid computing, and telecommunication systems. He is currently involved in research projects of the Spanish Ministry of Science and Education and private companies.

J. E. Muñoz Expósito received the M.S. degree in telecommunication engineering from Malaga University, Malaga, Spain, in 1995 and the Ph.D. degree in telecommunication engineering from Jaén University, Jaén, Spain, in 2009.

Since 2003, he has been an Associate Professor with the Telecommunication Engineering Department, Jaén University. His current research interests include speech and audio analysis, computer networks, and soft computing. He is currently involved in research projects of the Spanish Ministry of Science and Education and private companies.

A. J. Yuste received the M.S. degree in telecommunication engineering from Malaga University, Malaga, Spain, in 1994.

Since 2003, he has been an Associate Professor with the Telecommunication Engineering Department, Jaén University, Jaén, Spain. His current research interests include routing in ad hoc networks, grid computing, and trace analysis of computer networks. He is also involved in research projects of the Spanish Ministry of Science and Education and private companies.