Applied Soft Computing 29 (2015) 424–435

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Rules discovery in fuzzy classifier systems with PSO for scheduling in grid computational infrastructures

S. García-Galán*, R.P. Prado, J.E. Muñoz Expósito

Telecommunication Engineering Department, University of Jaén, Alfonso X el Sabio, 28 Linares, Jaén, Spain

A R T I C L E I N F O

Article history:
Received 24 June 2013
Received in revised form 29 July 2014
Accepted 15 November 2014
Available online 20 January 2015

Keywords:
Fuzzy computing
Fuzzy classifier systems
Particle swarm optimization
Grid computing

A B S T R A C T

Particle swarm optimization (PSO) is a bio-inspired optimization strategy founded on the movement of particles within swarms. PSO can be encoded in a few lines in most programming languages, it uses only elementary mathematical operations, and it is not costly as regards memory demand and running time. This paper discusses the application of PSO to rules discovery in fuzzy classifier systems (FCSs) instead of the classical genetic approach and it proposes a new strategy, *Knowledge Acquisition with Rules as Particles* (KARP). In KARP approach every rule is encoded as a particle that moves in the space in order to cooperate in obtaining high quality rule bases and in this way, improving the knowledge and performance of the FCS. The proposed swarm-based strategy is evaluated in a well-known problem of practical importance nowadays where the integration of fuzzy systems is increasingly emerging due to the inherent uncertainty and dynamism of the environment: scheduling in grid distributed computational infrastructures. Simulation results are compared to those of classical genetic learning for fuzzy classifier systems and the greater accuracy and convergence speed of classifier discovery systems using KARP is shown.

1. Introduction

Fuzzy classifier systems (FCSs) are fuzzy rule-based systems (FRBSs) that integrate expert knowledge and learn through credit assignment and classifier discovery processes [1]. These expert systems are extensively used in current applications because of their ability to deal with uncertain, imprecise or partial information, e.g., medical diagnosis [2], image analysis [3], adaptive sampling [4] and protein classification [5]. Their main strength is their ability to offer an efficient trade-off between the empirical accuracy of classical engineering strategies and the interpretability obtained by the use of natural language through linguistic labels.

The prototype learning of a FCS is divided into two main processes: the evaluation in the credit assignment system and the evolution in the classifier discovery system [6]. In the former, a collection of classifiers is provided and the efficacy of each classifier is determined considering feedback from the environment. In the latter, new classifiers are generated. In contrast to conventional expert systems where rules are provided by knowledge engineers, FCSs use discovery strategies to obtain new classifiers for their knowledge bases that are generally presented in the form of "IF-THEN" structures. Specifically, the traditional classifier discovery process for FCSs uses genetic algorithms (GAs) for this purpose based on the Michigan approach. In this way, the classifier discovery system generates new classifiers from the evolution of a previous set of classifiers. Basically, this approach selects those classifiers with high usefulness to take part and to generate new individuals in the next generation. In other words, with the aim of evolving the system knowledge, the genetic approach substitutes the set of classifiers of the population showing the lower efficiency with new classifiers that are generated through the application of genetic operators. Hence, the learning of rules is achieved following a similar procedure to natural evolution.

In this work, an alternative bio-inspired method for classifier discovery systems, *Knowledge Acquisition with Rules as Particles* (KARP) is proposed, based on the swarm intelligence optimization strategy, Particle Swarm Optimization (PSO) [7]. PSO is a stochastic evolutionary algorithm that has shown its efficacy for the optimization of complex multi-dimensional problems in diverse areas including articulated human motion tracking [8], interplanetary trajectories [9], reactive power dispatch [10] and mortality prediction of septic patients [11]. It is a very straightforward strategy derived from a simple concept based on the movement of swarms that has become an extended optimization procedure. The strategy

* Corresponding author. Tel.: +34 953 648556.
E-mail addresses: sgalan@ujaen.es (S. García-Galán), rperez@ujaen.es (R.P. Prado), jemunoz@ujaen.es (J.E. Muñoz Expósito).

http://dx.doi.org/10.1016/j.asoc.2014.11.064
1568-4946/© 2015 Elsevier B.V. All rights reserved.

only uses elementary mathematical operations, it is not costly in terms of memory and speed and its ability to achieve a fast convergence has been demonstrated [7,12]. Furthermore, PSO has shown to be more efficient than other optimization strategies such as GAs in many situations [13]. One of the main advantages of this strategy with respect to GAs is given by its simple implementation and the limited number of parameters to be fixed. PSO does not require genetic operators such as crossover or mutation, but particles are updated considering internal velocities. In addition, particles have memory and consider unidirectional information exchanges what can significantly reduce the necessary communications among particles. In this work, the utilization of PSO is suggested to be adapted to classifier discovery systems in FCSs.

In order to test performance, KARP is used in the classifier discovery system of FCSs in a well-known problem of practical importance, the design of metaschedulers for the emerging computational grids. Grid computing is a high-performance computational framework founded on the share of geographically distributed and heterogeneous computing resources with the purpose of solving large-scale problems in science, engineering, and technology [14]. Due to the high dynamism of grids, the coordination and cooperation of resources is a major problem and a lot of research has been done to find effective scheduling strategies. Specifically, the interest in fuzzy rule-based schemas is increasingly emerging to achieve near optimal schedules given their ability to tolerate and work with information subject to a certain degree of uncertainty, as the resources state in grid computing [15,16]. Hence, KARP is used for the self-learning of FCS metaschedulers for grid computing and the strategy is compared to fuzzy schedulers made up with classical classifiers discovery systems based on Michigan approach. There exist previous works that use PSO for knowledge acquisition in fuzzy schedulers where every particle represents a whole rule base (RB) that can be employed by the fuzzy schedulers [17,18] and therefore, they are computationally comparable to fuzzy genetic systems based on Pittsburgh approach. However, this work presents a learning strategy where every particle represents a single rule of the rule base used by the fuzzy scheduler, and in this way, it is computationally equivalent to fuzzy genetic systems based on Michigan approach.

The rest of this paper is organized as follows. In Section 2, the general background to rules discovery in FCSs with GAs and the essentials of PSO are presented. The proposed learning strategy based on PSO, KARP, is introduced in Section 3. Section 4 analyzes the performance of KARP in a problem of practical importance, scheduling in grid computing and results are compared with the classical genetic learning. Finally, Section 5 concludes the paper.

2. Rule discovery in fuzzy classifier systems

As introduced by Holland [6,19], a natural procedure in rule discovery processes of rule-based systems is to represent a whole rule as an individual, to generate a population of candidate rules, and to apply genetic operators to originate new generations of rules. He suggested a cognitive model (classifier system) where the individuals of the population are rules and a rule set represents the whole population. This strategy is known as Michigan approach. In this section, a brief introduction to rule discovery in FCSs is presented. First, the representation of fuzzy rules used in this work and the general procedure of FCSs systems for learning new rules based on the application of GAs with Michigan approach is described. Next, some background to the PSO optimization method that is adapted for rules learning in FCSs as an alternative strategy is provided.

2.1. Rule encoding and genetic classifier discovery

A fuzzy rule or classifier R_i is made up of two differentiated terms: the antecedent and the consequent. The antecedent term corresponds to the rules activation condition whereas the consequent represents the output contribution to the final decision. In this work, a Mamdani codification [6,20] is considered for the rules and thus, no analytical functions are employed in consequent terms as in the case of Takagi, Sugeno and Kang approach [21]. Hence, a rule R_i is denoted as,

$$R_i = \text{IF}\, x_1 \text{ is } a_{i,1},\ c = \text{and/or}, \ldots x_n \text{ is } a_{i,n} \text{ THEN } y \text{ is } b_i \tag{1}$$

where $(x_1, \ldots, x_n)$ indicates the set of n input features, c represents the rule connective, $a_{i,m}$ and b_i are the associated fuzzy sets for feature x_m and output for rule i, respectively, with $m = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, N$, N denoting the number of rules within the RB. As it can be inferred, $a_{i,m}$ and b_i correspond to one of the possible fuzzy sets for input x_m and output, NF_{in}^{m} and NF_{out}, respectively. Fig. 1 depicts the encoding of a fuzzy rule considering two connectives "and/or" encoded as "1/2", respectively.

The rules of a FCS constitute a main part of the knowledge of the expert system and thus, its efficiency is strongly related to their quality and in turn, to the learning mechanisms. The traditional procedure of FCSs for learning new classifiers is based on the application of GAs following Michigan approach. A basic execution cycle can be summarized as follows [6]:

(1) Generate a set of rules as initial population N of the strategy.
(2) Rank rules of N in regard to decreasing fitness or strength representing its usefulness.
(3) Select $\lambda * N$ rules to be replaced among those rules with the lowest usefulness, where λ represents the selection rate.
(4) Select $\lambda * N/2$ pairs of rules to be parents of the new generation among those rules with the highest usefulness.
(5) Apply genetic operators to the $\lambda * N/2$ pairs selected at step 4 and generate $\lambda * N/2$ offspring pairs of rules.
(6) Substitute the $\lambda * N$ rules selected at step 3 with the offspring population generated at step 5.

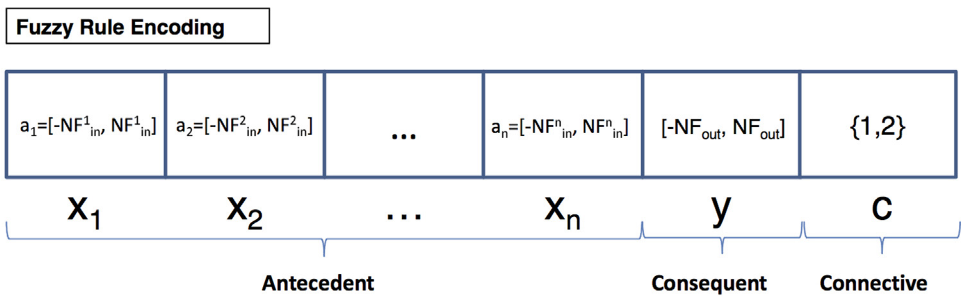

Fig. 1. Mamdani fuzzy rules encoding.

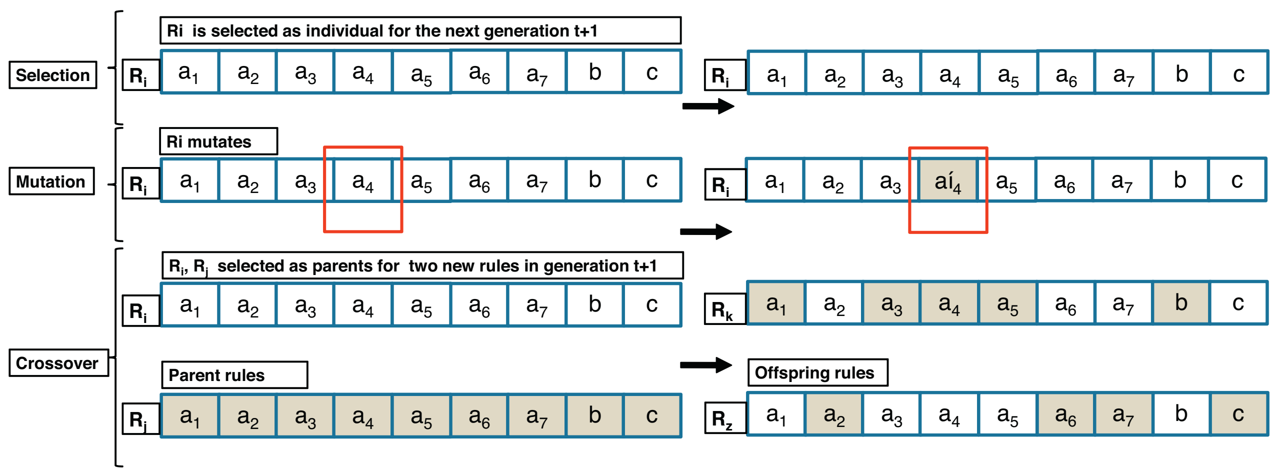
Selection
Ri is selected as individual for the next generation t+1
Mutation
Ri mutates
Crossover
Ri, Rj selected as parents for two new rules in generation t+1
Parent rules
Offspring rules
Ri
Rj
Rk
Rz
a1 a2 a3 a4 a5 a6 a7 b c
aí4

Fig. 2. Example of the genetic operators in classifier discovery systems of FCSs with Mamdani rules.

Next, the different processes of the algorithm are described:

- *Initialization*: the generation of N rules is done randomly in a way that no previous knowledge is required. Every antecedent $a_{i,m}$ component can take a value in the interval $[-NF_{\text{in}}^{m}, NF_{\text{in}}^{m}]$ where the negative elements correspond to the negation of the associated positive fuzzy set (i.e., not operator), the consequent is limited to the interval $[-NF_{\text{out}}, NF_{\text{out}}]$ and the connective of every rule can be "and/or" operator.
- *Selection*: The selection process can be performed randomly with proportional selection probabilities of rules according to associated fitness, in a way that the strongest rules become parents of the new rules and participate in the next generation.
- *Crossover*: The mixing of the genome is done considering pairs of rules as in reproduction or biological recombination. In the uniform crossover every rule component is compared between parents and they are swapped with a given probability.
- *Mutation*: It consists of the random alteration of a component in a rule from its original form.

Fig. 2 illustrates an example of genetic operations (i.e., selection, crossover and mutation) of fuzzy rules with seven antecedents in Michigan approach.

2.2. Optimization based on social behavior

Through cooperation and competition among the population of rules, the genetic optimization approach in classifier discovery systems of FCSs can generally achieve good solutions. As discussed above, this strategy is founded on natural evolution. PSO, on the other hand, is derived by the social behavior of swarms such as flocks or schools to achieve better locations [7]. In contrast to GAs, PSO does not use evolutionary operators to handle individuals so-called particles but they move in the search space considering an internal velocity that is modified in a dynamic way regarding their own flying experience and their neighbors flying experience. Specifically, adjusting particles velocity is done considering three different terms: an inertial term, a self-recognition term or tendency of the particle to return to its best location, and a social term that is related to the individuals tendency to return to the best location reached by their neighbors. Hence, at every step $(t+1)$, position x and velocity v of particle i are adjusted by the expression,

$$v_i^{(t+1)} = \omega v_i^t + d_1 r_1 (Pb_i^t - x_i^t) + d_2 r_2 (Gb^t - x_i^t) \tag{2}$$

$$x_i^{(t+1)} = x_i^t + v_i^{(t+1)} \tag{3}$$

where d_1, d_2 represent weight factors, Pb_i denotes the best location achieved by particle i, Gb_i is the best position found by the neighbors of particle i, r_1, r_2 indicate random factors, and ω is the inertia weight. In contrast to GAs, PSO does not contemplate a selection process where the fittest particles take part in the next generation but all the individuals of the swarm survive during the whole optimization process. The adaptation of the particles is done through the associated velocity. As shown in Eq. (2), each particle modifies its velocity in every step regarding its own current state, the own best solution and the best solution found by the whole swarm in the runs [12,22]. Hence, PSO does not perform a crossover process but it considers a so-called "conscious mutation" [23] through Eq. (3) where individuals evolve according to previous experience of the group. A key parameter in PSO is the inertia weight ω, which allows the balance between global and local searches. High values for ω promote global search whereas reduced values promote local search. In this way, inertia weight is generally associated to a decreasing value through the run of the strategy with the aim that the whole search space can be explored initially and that the search is intensified around final locations. In the next section, the adaptation of PSO to the learning of rules in FCSs is presented.

3. *Knowledge Acquisition with Rules as Particles* (KARP)

In this work, a novel strategy for classifier discovery systems for the generation of high quality fuzzy rules in FCSs is suggested: *Knowledge Acquisition with Rules as Particles (KARP)*. The learning strategy is based on the adaptation of PSO to the evolution of fuzzy rules in FCSs. In KARP, the swarm is made up of NP particles, each particle P_i representing a fuzzy rule with the encoding introduced in Section 2.1 and the goal is to move these particles or rules through the space to obtain good knowledge for the RB. The quality of particles is rated in terms of a selected fitness which is desirable for the FCS and that is obtained through the credit apportion system of the FCS. KARP approach stages are the following:

(1) *Generate a RB made up of NP particles determined by its position to become the individuals of the swarm. Definition of position of particles in the integer space.* As stated before, PSO is a well-known optimization strategy where particles are represented by a position in the search space and whose aim is to find the best particle/position considering a specific cost function that evaluate the quality of the particle/position in that search space. The goal of KARP is obtaining high-quality Mamdani fuzzy rules for a FCS. Thus, the terms particle, position and rule are interchangeable used in this work. With this aim, each

particle/position in KARP represents a rule and a rule is represented by a vector of integers values that indicate their antecedents a, consequent b and connectives c associated to a Mamdani structure (Eq. (1)). The structure for particle/position i is presented as follows,

$$P_i = [\, a_1^i \quad a_2^i \quad \ldots \quad a_n^i \quad b^i \quad c^i \,] \tag{4}$$

where n is the number of input variables. The possible values for this vector are integers values in all of its terms and thus, the possible coordinates (i.e., antecedents, consequents and connector) of the particle/position are restricted to integers values. Therefore, KARP purpose is to obtain a set of vectors (set of particles/positions) that represents a set of fuzzy rules making up a high-quality RB for the FCS. Antecedents, consequents and connector representation are explained as follows in detail:

Antecedent and consequent representation: Every possible fuzzy set for every variable is enumerated consecutively starting from 1 to NF, where NF is the number of possible fuzzy sets for a given variable. In every rule/particle, each variable is associated to an integer value representing its associated fuzzy set. Also, 0 value indicates the variable absence in the rule and negatives values represent the complementary set (i.e., NOT). Also, to keep the coherence of the fuzzy rules, a rule presenting zero 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, consequents are allowed to be set to zero as a way of representing the associated rule non contribution. Thus, every particle i in the swarm is restricted to the integer search space for antecedents and consequents,

$$a_j^i \in [-NF_{\text{in}},\ NF_{\text{in}}],\ j \in \{1, 2, \ldots, n\} \tag{5}$$

$$b^i \in [-NF_{\text{out}},\ NF_{\text{out}}] \tag{6}$$

where NF_{in} and NF_{out} represent the number of fuzzy set for the input and the output, respectively, where $NF_{\text{in}},\ NF_{\text{out}} \in \mathbb{N}$.

Connector representation: If the connector is set to 1, it indicates that the connective for the antecedents of the rule is "and". However, if it is set to 2, it indicates the connector for the antecedents of the rule is "or". Thus, the connector value must satisfy $c^i \in \{1, 2\}$ which made up the integer search space for connectives of rules.

(2) *Associate each particle a velocity that drives the modification of each rule in every step. Definition of velocity of a particle in the integer space.*

On the other hand, the swarm position update process must be considered. With this aim, the velocity vector is introduced for this approach. The velocity at every iteration is represented as a vector of real values with the particle dimensions for each particle i, $n+2$,

$$V_i = [\, v_1^i \quad v_2^i \quad \ldots \quad v_n^i \quad v_{n+1}^i \quad v_{n+2}^i \,] \tag{7}$$

The velocity determines the increment or decrement in the value of the antecedent, consequent and connectives that must be added to a particle/position antecedents, consequent and connectives, respectively, in the following iteration and that lead to the movement or modification of the particle/position.

Also, as in the initialization of rules, velocity must also be generated bearing in mind lower and upper bounds, $V_{\min}$ and $V_{\max}$, respectively,

$$v_k^i \in [V_{\min},\ V_{\max}],\ k \in \{1, 2, \ldots, n+2\} \tag{8}$$

with $k = 1, 2, \ldots, n+2$. This limitation is given by the explosion control. As found in [24], explosion is generally contained through the consideration of a $V_{\max}$ parameter, which limits step size of velocity. That is, with the goal of effectively guiding particles in the search space, the velocity in each iteration must be limited. In this way, the elements of the velocity must meet the following condition [25],

$$v_k^i = \text{sign}(v_k^i,)\min(|v_k^i|, V_{\max}) \tag{9}$$

where the value for $V_{\max}$ is fixed according to the dimension of the search space. As in the case of rules, the velocity vectors must be randomly generated.

(3) *Identify the best particle found by every individual and the whole swarm. Quality of a particle.*

Once all the population of particles/rules and their associated velocities have been defined, they are evaluated to know its quality. With this aim, the RB made up by all the rules/particles is used by the FCS and its performance is measured considering a determined cost function. Considering this performance, a strength value that estimates each rule/particle quality using a credit assignment strategy (e.g., Holland's Bucket Brigade [26]) is obtained.

(4) *Update the velocity of each particle according to the their own and social experience.* In this stage, particles/rules are moved (i.e., modified) to obtain a greater quality. Specifically, the evolution of particles/rules is done modifying their associated velocity and adding this velocity to the particle/rule value. This modification is done considering three terms:

- *Inertial component*: modification of the velocity of the particle/rule given by its value in the current iteration t weighted by a factor ω, so-called inertial weight,

$$\omega \otimes V(t) \tag{10}$$

where $\otimes$ denotes the scalar multiplication of vectors.

- *Self-recognition component or inner tendency to return to its best position*: modification of the velocity of the particle/rule given by the weighted difference of the best rule values for the particle/rule found so far, $P^{\#}(t)$, and its current value $P(t)$. The weight factor is denoted as $(d_1 * r_1)$,

$$(d_1 * r_1) \otimes (P^{\#}(t) \ominus P(t)) \tag{11}$$

where $\ominus$ represents the regular subtraction between vectors.

- *Social component representing the swarm particle leaning to move toward the best position found by its neighbors*: modification of the velocity of the particle/rule given by the weighted difference of the best rule values found by all the swarm, $P^{*}(t)$, and its current value $P(t)$. The weighted factor is denoted as $(d_2 * r_2)$,

$$(d_2 * r_2) \otimes (P^{*}(t) \ominus P(t)) \tag{12}$$

Therefore, the velocity for a particle/rule in the following iteration $t+1$ is given by,

$$\begin{aligned} V(t+1) = {} & \omega \otimes V(t) \oplus (d_1 * r_1) \otimes (P^{\#}(t) \ominus P(t)) \\ & \oplus (d_2 * r_2) \otimes (P^{*}(t) \ominus P(t)) \end{aligned} \tag{13}$$

where $\oplus$ indicates the regular addition between vectors. Note that after initialization, given to the weight factors mentioned above, velocity vectors are not necessarily composed of integers. However, the integer nature of particles must be considered in its update process as explained in the following step.

(5) *Update the location of every particle in the search space.*

Finally, the particle value is updated as follows:

$$P(t+1) = round[P(t) \oplus V(t+1)] \tag{14}$$

where *round* function obtains the closer integer for every vector term. The update of particles must respect the coherence of the formulation of rules. Antecedents, consequents and connectives may exceed the integer search space as a consequence

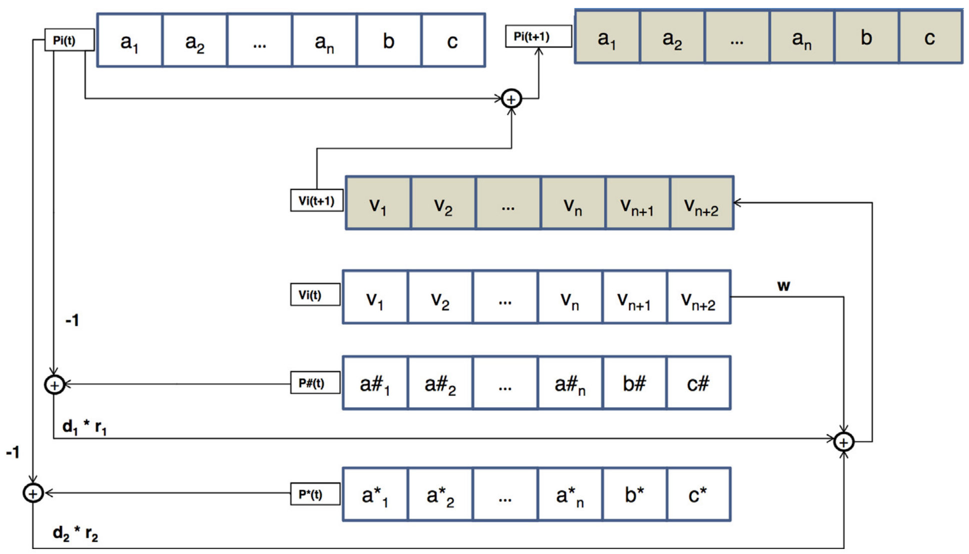

Fig. 3. Updating particle P_i of the swarm in KARP strategy in step $t+1$.

of the movement of particles (i.e., velocities vectors updated by Eq. (13) are not necessarily composed of integers, therefore, when considering Eq. (14), the value for antecedents, consequents and connectives must be rounded to the closer integers values within the search space). Thus, the following restrictions must be satisfied in every update process,

$$a_k^i = \begin{cases} NF_{\text{in}} & \text{if } a_k^i > NF_{\text{in}} \\ -NF_{\text{in}} & \text{if } a_k^i < -NF_{\text{in}} \\ round(a_k^i) & \text{otherwise} \end{cases} \tag{15}$$

$$b^i = \begin{cases} NF_{\text{out}} & \text{if } b^i > NF_{\text{out}} \\ -NF_{\text{out}} & \textit{if } b^i < -NF_{\text{out}} \\ round(b_k^i) & \text{otherwise} \end{cases} \tag{16}$$

$$c^i = \begin{cases} 1 & \text{if } c^i < 1 \\ 2 & \text{if } c^i > 2 \\ round(c^i) & \text{otherwise} \end{cases} \tag{17}$$

Algorithm 1. *Knowledge Acquisition with Rules as Particles*, KARP.

```
Rules initialization
1. Swarm initialization: Num_particles=Num_rules=NP,
      Parameters initialization: Num_iter, Init_Inertial_weight $\omega_o$,$r_1$, $d_1$, $r_2$, $d_2$
2. Rules coherence
   for (every particle/every rule i),
      (a) No zero antecedents (rules coherence)
      (b) if $|a_k^i| > NF_{in}$ then Eq. (15)
      (c) if $|b_k^i| > NF_{out}$ then Eq. (16)
      (d) if $c^i \notin \{1, 2\}$ then Eq . (17)
   endfor
Velocity initialization
3. Random setting of velocity.
4. Velocity constraints. Eq. (8)
Best rules initialization
5. Initialize Gbest($P^*$) rule /Pbest($P^\#$) rule
Best rules identification/Position updating/Velocity updating cycle
for (Num_steps)
   for (Num_particles)
      1. Update position. Eq. (14)
      2. Constraints Rules-Swarm position.
         (a) No zero antecedents (rules coherence)
         (b) if $|a_k^i| > NF_{in}$ then Eq. (15)
         (c) if $|b_k^i| > NF_{out}$ then Eq. (16)
         (d) if $c^i \notin \{1, 2\}$ then Eq. (17)
      3. Evaluate usefulness of the rule.
      Particles ++
   endfor
   Update Gbest ($P^*$) rule
   for (Num_particles)
      1. Update Pbest ($P^\#$) rule.
      2. Update velocity. Eq. (13)
      3. Velocity Constraints. Eq. (8)
      Particles ++
   endfor
   iter++
endfor
Return solution: Gbest ($P^*$) rule
```

Fig. 3 illustrates the updating process of a rule P_i in step $t+1$. As shown, P_i new location is obtained by the combination of the particle's current value and the new adjusted velocity. Specifically, the adjusted velocity is made up of the inertial, the self-recognition and the social components subject to weight factors. The basic cycle of KARP after initialization (i.e., steps 3–5) is reiterated until the stopping condition is met (e.g., number of iterations or determined quality for rules). In this way, particles are guided through a set of position updating steps to efficient locations that are translated into good quality rules. KARP strategy is summarized in Algorithm 1.

Note that this work is focused on the design of the learning of the RB (learning of rules with fixed membership functions) in order to obtain fast and simple learning processes and with high interpretability. This is desirable in many systems like FRBS-based schedulers, where the time of the learning and interpretation of knowledge is relevant for its efficient performance and improvement [6,27]. Therefore, as in the case of genetic fuzzy learning such as Michigan approach (used for comparison), KARP assumes a predefined set of fuzzy membership functions in the database to which the rules refer to by means of linguistic labels and the evolutionary process adapts the RB, working with particles instead of

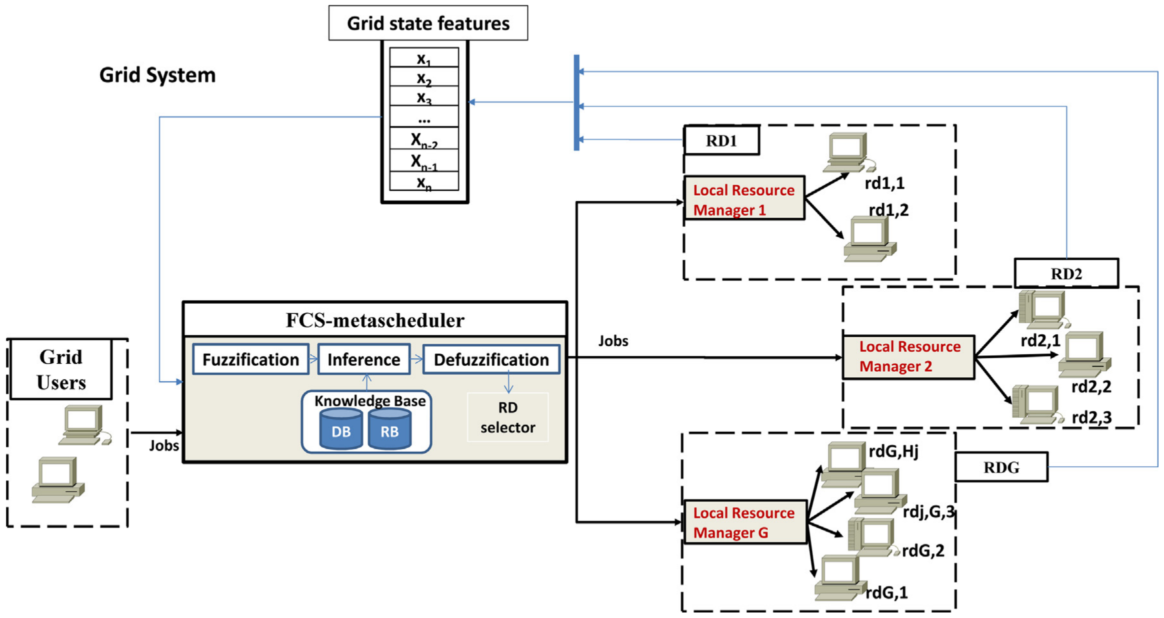

Fig. 4. General structure of the grid systems and FCS metascheduler.

chromosomes that describe a single fuzzy rule. The RB is represented by a list of rules following the Mamdani coding as presented before. The most common rule structure in Mamdani FRBSs involves the use of linguistic variables. Each linguistic variable has associated a fuzzy partition and the fixed number here for every variable represents the value in numeric form of the linguistic set which is not tuned.

4. Simulation results and discussion

The proposed strategy for rules discovery in FCS, KARP, is evaluated in a challenging problem of practical importance nowadays, scheduling in grid computing. First, an introduction to scheduling in grid computing and the fundamentals of the application of FCSs to solve this problem are presented. Next, performance results of KARP are discussed.

4.1. The problem: scheduling in grid computing

Grid computing is an emerging computational framework to solve large-scale problems in diverse fields of science and engineering based on the cooperation of widely distributed resources [14]. A grid is made up of a set of heterogeneous and geographically distributed resources from different institutions that share capabilities to overcome current limitations machines as independent entities. The structure of a hierarchical is grid is made up of two main levels. As shown in Fig. 4, the grid can be regarded as a set G of geographically distributed sites or resources domain (RD), $GS = \{RD_1, RD_2, \ldots, RD_G\}$, where each site is defined as a collection of H_j heterogeneous computational resources $RD_j = \{r_{j,1}, r_{j,2}, \ldots, r_{j,H_j}\}$. Furthermore, grids are known to be *"fully dynamic environments with uncertainties"* [28] where machines actively become available, fall down or reduce its available capabilities in regard to several administrative policies that may change with time. A major challenge of these systems is the efficient coordination of resources and users demand, which is considered critical to harness grid networks potential. In this sense, scheduling in grid computing is a NP-hard problem [29]. In a hierarchical grid, two types of scheduling structure can be differentiated. On the one hand, the Local Resource Managers (LRMs) are responsible for allocating jobs within their own site or RD according to the site own local policies, whereas on the other hand, metaschedulers must decide the LRM where jobs must be sent to. Hence, the goal of a metascheduler is to manage the grid resources efficiently through the local scheduling systems coordination.

Recent tendencies are focused on the adaptation of fuzzy rule-based models to cope with high dynamism and vagueness of such systems state in the scheduling [15,30]. Fuzzy rule-based schedulers base their schedules on the knowledge of the environment in the form of fuzzy rules or RBs and thus, the scheduler performance is subject to RB quality and consequently with the learning of fuzzy rules. Specifically, as shown in Fig. 4, a FCS metascheduler follows the structure of a Mamdani FRBS, i.e., fuzzification, inference and defuzzification systems. The objective of a FCS metascheduler is to provide a performance index for every participating RD or selector, y_o, that represents the suitability of this RD to be selected for the following schedule. With this aim, at every scheduling step, the metascheduler retrieves the state information of all the available RDs given by a set of features in the form of crisp values. Also, these crisp values are transformed into fuzzy values through the fuzzification system in a way that the associated uncertainty is considered in the characterization of resources real conditions. Next, the inference system applies the system knowledge to obtain a fuzzy selector that is translated in a crisp performance index by the defuzzification system. Finally, once a selector y_o has been derived for all the RDs, the site with highest selector is selected as the next target.

As it can be derived, the whole scheduling process with FCS in based on the application of expert knowledge to the fuzzy state of the grid and thus, the quality of this knowledge in the form of rules and associated learning process is critical. In this work, the proposed learning strategy for FCSs, KARP, will be studied to improve the performance of FCS metaschedulers in grid computing.

4.2. Classifier discovery system evaluation

The fuzzy FCS metascheduler with KARP learning is evaluated through simulations with Alea software [31], a simulation toolkit based on GridSim [32] that lets the use of grid scenarios and traces from existing facilities for scheduling strategies evaluation. Specifically, the grid scenario is based on a Czech National Grid Infrastructure Metacentrum project [33], whose aim is to support the development of a distributed high-performance computing

Table 1
Input features for the fuzzy metascheduler.

Feature		Description
x_1	Number of free processing elements (FPE)	Number of free processing element within RD_i
x_2	Previous tardiness (PT)	Sum of tardiness of all finished jobs in RD_i
x_3	Resource makespan (RM)	Current makespan for RD_i
x_4	Resource tardiness (RT)	Current tardiness of jobs within RD_i
x_5	Previous score (PS)	Previous deadline score of already finished jobs in RD_i
x_6	Resource score (RS)	Number of non delayed jobs so far in RD_i.
x_7	Resources in execution (RE)	Number of resources currently executing jobs within RD_i

infrastructure through the joint cooperation of resources from academic and research institutions worldwide. In our simulations, the grid system consists of 14 sites integrating 806 central process units of diverse kinds (i.e., Opteron and Xeon) and speed (i.e., 1500–3200 MHz) allocated in 210 resources with heterogeneous memory size (i.e., 1005,000–27343,000 KB) running Linux. Furthermore, machines setting and machines maintenance, reservation and dedication traces are included in a way that information about the changes in performance of every grid machine through time due to failures, reservation, dedication or unavailability caused by changing condition or sharing policies are considered. In addition, jobs traces specify a set of parameters of every considered job: job identifier, associated job user or owner priority, list of properties to be met in the target resource and arrival time to the scheduler. These traces are collected from Metacentrum facilities in MWF format (available at [33]).

The positive or negative performance of the acquired knowledge is evaluated using *makespan* [34] as performance index or fitness of the learning process. In a high-performance grid, the general goal of the scheduler is to maximize the productivity or throughput. Relevant previous works in the optimization of scheduling strategies [35–38] suggest *makespan* as performance criteria with this aim and it is also considered in this work. Thereby, the goal of the scheduler is the minimization of the latest job finalization time or *makespan* [34],

$$\min\{\max_{j\in J} T_j\} \tag{18}$$

with T_j the finalization time of the latest job j of the set of jobs J.

4.2.1. Grid state variables and associated fuzzy sets

Regarding the configuration of the FCS metascheduler, the state of the grid system is described through 7 variables as shown in Table 1.

Other features may be also used to obtain a more complete grid state characterization. Nevertheless, since the search space for the expert system RB highly expands with the number of features, this selection is considered to reach a balance between the RD state characterization and the effort of learning strategy for rule discovery [15,39]. Also, Fig. 5 illustrates the fuzzy sets for the antecedent and consequent of the FCS metascheduler (corresponding to the grid state features and suitability index), the associated linguistic and numerical representation. Specifically, three fuzzy sets are considered for every antecedent, $NF_{\text{in}} = 3$, with labels *low*, *middle* and *high* encoded as 1, 2, and 3, respectively. Moreover, the negation sets, i.e., *not low*, *not middle* and *not high*, are associated to integers −1, −2 and −3, respectively. Additionally, the absence of the antecedent is encoded as 0,

$$a = \{not\ high,\ not\ middle,\ not\ low,\ abscence,\ low, \tag{19}$$

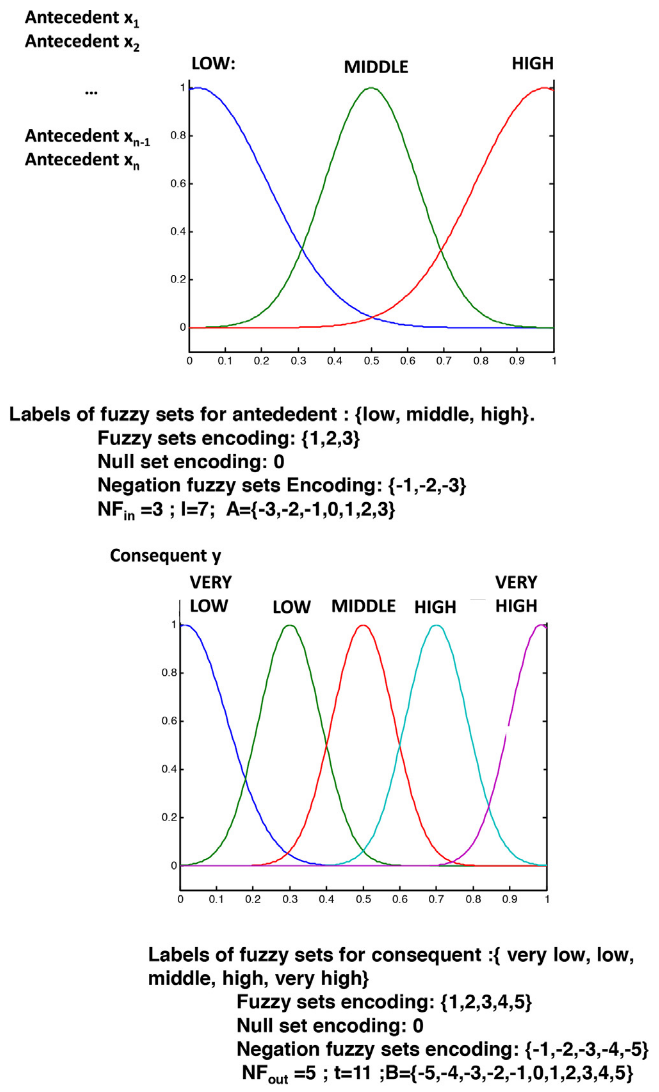

Fig. 5. Fuzzy sets for grid features and resource domain selector. Associated linguistic and numerical representation.

$$middle,\ high\} = \{-3,\ -2,\ -1,\ 0,\ 1,\ 2,\ 3\};\ l = 7 \tag{20}$$

The consequent contemplates five fuzzy sets with labels *very low, low, middle, high and very high* are encoded as 1, 2, 3, 4, 5, respectively. Also, negation sets corresponding to *not very low, not low, middle, high and very high* are encoded as −1, − 2, − 3, − 4, − 5, respectively, with the absence of consequent denoted by 0,

$$b = \{not\ very\ high,\ not\ high,\ not\ middle,\ not\ low,\ not\ very\ low, \tag{21}$$

$$abscence,\ very\ low,\ low,\ middle,\ high,\ very\ high\} \tag{22}$$

$$= \{-5,\ -4,\ -3,\ -2,\ -1,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5\};\ t = 11 \tag{23}$$

4.2.2. Individual's representation

The structure for each particle/position/rule i in grid computing problem in the simulations considering 7 input variables (i.e., antecedents of rules) describing the RD state, 1 output variable (i.e., consequent of rules), and 1 possible connector (i.e., "and/or" connectives of antecedents) relating inputs variables is,

$$P_i = [\,a_1^i \quad a_2^i \quad \dots \quad a_7^i \quad b^i \quad c^i\,] \tag{24}$$

with $i = 1, 2, \dots, NP$ and

$$a_j^i \in [-3,\ 3],\ j \in \{1, 2, \dots, 7\} \tag{25}$$

Table 2
Example of obtained RB with KARP for the FCS metascheduler in linguistic and associated vectorial forms.

```
P1. If (FPE is LOW) or (PT is not HIGH) or (RM is not LOW) or (PS is not HIGH) or (RE is LOW) then (yo is not HIGH)
P2. If (PT is not LOW) or (RM is not MIDDLE) or (RT is not LOW) or (PS is not LOW) or (RS is LOW) or (RE is LOW) then (yo is not HIGH)
P3. If (FPE is not HIGH) and (PS is not LOW) and (RS is LOW) and (RE is not LOW) then (yo is not VERYHIGH)
P4. If (FPE is MIDDLE) or (PT is not HIGH) or (RT is LOW) or (RS is LOW) or (RE is MIDDLE) then (yo is not MIDDLE)
P5. If (FPE is LOW) or (PT is not LOW) or (RM is not LOW) or (RT is not LOW) or (PS is not HIGH) or (RS is not LOW) or (RE is not MIDDLE) then (yo is not VERYLOW)
P6. If (FPE is HIGH) or (PT is not LOW) or (RM is LOW) or (PS is not MIDDLE) or (RS is LOW) or (RE is LOW) then (yo is not VERYHIGH)
P7. If (FPE is LOW) or (PT is not LOW) or (RM is not MIDDLE) or (RT is not LOW) or (PS is not MIDDLE) or (RS is not MIDDLE) or (RE is LOW) then (yo is not HIGH)
P8. If (FPE is MIDDLE) or (PT is not MIDDLE) or (RM is not LOW) or (RT is not MIDDLE) or (PS is not LOW) or (RE is LOW) then (yo is not LOW)
P9. If (FPE is not LOW) or (PT is not MIDDLE) or (RT is not LOW) or (PS is not LOW) or (RE is not MIDDLE) then (yo is not MIDDLE)
P10. If (FPE is MIDDLE) and (PT is not LOW) and (RM is not LOW) and (RT is not LOW) and (RS is LOW) and (RE is not MIDDLE) then (yo is not VERYLOW)

P1=[1 −3 −1 0 −3 0 1 −4  2 ]
P2=[0 −1 −2 −1 −1 1 1 −4  2 ]
P3=[−3 0 0 0 −1 1 −1 −5  1 ]
P4=[2 −3 0 1 0 1 2 −3  2 ]
P5=[1 −1 −1 −1 −3 −1 −2 −1  2 ]
P6=[3 −1 1 0 −2 1 1 −5  2 ]
P7=[1 −1 −2 −1 −2 −2 1 −4  2 ]
P8=[2 −2 −1 −2 −1 0 1 −2  2 ]
P9=[−1 −2 0 −1 −1 0 −2 −3  2 ]
P10=[2 −1 −1 −1 0 1 −2 −1  1 ]
```

$$b^i \in [-5,\ 5] \tag{26}$$

$$c^i \in \{1, 2\} \tag{27}$$

The number of rules, NP, in FCS for this work has been set considering previous experimental studies in the optimal number of rules in FCS metaschedulers in grid computing. Specifically, the results of these works were published in [15,39]. In these preliminary analysis, the optimal number of rules for seven-input-one-output fuzzy system, with cardinalities for the inputs and output $T(x)$ and $T(y)$ using Gaussian sets, 3 and 5, respectively (as in the present work) was studied for FCS metaschedulers in grid computing and 10 rules were indicated as the optimal number of rules for the problem in hand. Table 2 presents an example of a swarm in its associated linguistic and vectorial forms.

4.2.3. Individual's evaluation

The swarm of rules made up a RB which is used by the FCS metascheduler to allocate workload (i.e., set of jobs) in the grid to the different RD. Once all the jobs have finished their execution in generation n of the learning strategy, the resulted *makespan* is taken as the evaluation of the RB for generation n, f,

$$f_n = \min\{\max_{j \in J} T_j\} \tag{28}$$

Therefore, the shortest the achieved *makespan* f_n, the better the quality of the RB. Nevertheless, in order to evolve rules an evaluation of each rule within the RB is necessary. The evaluation of a rule is given by its strength. The total strength of a rule, $TS_{i,n}$, is obtained considering two different aspects: its influence in the final performance of the RB and its cooperation in the activation of other rules during the performance of the scheduler.

- Influence in the final performance of the RB

  The strength of rule i, $S_{i,n}$ is updated at each generation n considering its influence in the quality of the final performance of the RB as follows [6],

$$S_{i,n} = \begin{cases} S_{i,n-1} + K \cdot T_R \cdot E_{i,n} \cdot (1 - S_{R,n-1}) & \text{if } E_{i,n} \geqslant 0 \\ S_{i,n-1} + K \cdot T_R \cdot E_{i,n} \cdot S_{R,n-1} & \text{if } E_{i,n} < 0 \end{cases} \tag{29}$$

  where K and T_R represent a constant concerning the setting of the system memory and the rule truth value, respectively, and the evaluation for rule i in generation n is calculated as,

$$E_{i,n} = SP_n \cdot RI_{i,n} \tag{30}$$

  with SP_n the system performance and $RI_{i,n}$ the rule influence at generation n. SP_n is a rate value showing improvement or deterioration of the system performance in the current generation compared to the previous generation performance (f_{n-1}), and the overall performance until the current generation n on the basis of best and worst values for fitness f achieved by the strategy, $f_{\min}$ and $f_{\max}$, respectively, so far,

$$SP_n = \begin{cases} 1 & \text{if } f_n \leq f_{\min} \\ \dfrac{f_n - f_{n-1}}{f_{\min} - f_{n-1}} & \text{if } f_n > f_{\min} \wedge f_n \leqslant f_{n-1} \\ \dfrac{f_{n-1} - f_n}{f_{\max} - f_{n-1}} & \text{if } f_n < f_{\max} \wedge f_n > f_{n-1} \\ -1 & \text{if } f_n > f_{\max} \end{cases} \tag{31}$$

  $RI_{i,n}$ weights every rule contribution to the system output y (i.e., scheduler selector) in generation n in the defuzzification. It considers three different possibilities: the fuzzy set A in the consequent of the rule has been activated $\mu_A(y_n) > 0$ and so it contributes to the system output and the fuzzy set of the consequent of the rule has not been activated $\mu_A(y_n) = 0$ but the Euclidean distance of the limits of the fuzzy set of the consequent of the rule A and the system output has been reduced or increased from the previous generation $n-1$,

$$RI_{i,n} = \begin{cases} \mu_A(y_n) & \text{if } \mu_A(y_n) > 0 \\ 0 & \text{if } \mu_A(y_n) = 0 \wedge |A - y_n| \leqslant |A - y_{n-1}| \\ \dfrac{y_n - y_{n-1}}{y_{n-1} - y_l} & \text{if } \mu_A(y_n) = 0 \wedge |A - y_n| > |A - y_{n-1}| \end{cases} \tag{32}$$

  where y_1 denotes the closer bound of the associated fuzzy set to y [6].

- Cooperation in the activation of other rules

  The final quality of the RB (i.e., *makespan*) is achieved through the application of rules in the schedule of every job until the whole workload is allocated. Hence, the final success is given to a set of decisions. Note that the decision in every schedule has an influence in the state of the designed RD and thus, it has an influence on the activation of rules in the following schedule. Therefore, the contribution of activated rules in a schedule to the activation of rules in the following schedule is to be also valued to determine the strength of a rule. This is done through a credit assignment system in every generation n.

  One of the most successful strategies is Holland's Bucket Brigade (BB) [26] and it is used in this work. Essentially, BB algorithm considers each particle/rule as an intermediary that works with its suppliers (rules that make its conditions to be met) and consumers (rules that satisfy their conditions with its actions) and it is associated to a bid. If a rule wins a bid (i.e., it is activated), it has to pay part of this bid to its suppliers and it can receive payment from its customers in following stages. Thus, its strength increases if it receives more than it gives. Specifically, in BB, every rule, R_i, can be associated a bid value at schedule t in a given generation (in a generation the performance of a set of schedules is evaluated) that is expressed on the basis of the bid coefficient or specificity $Cbid_i$, the strength S_i and the firing strength F_i,

$$Bid_i(t) = Cbid_i \cdot S_i(t)_n \cdot F_i(t) \tag{33}$$

where the firing strength is defined as the minimum degree of belonging to the membership function of those messages satisfying the rule [40]. Considering this bid value, the credit assignment system update particles/rules associated strength for those particle/rules cooperating in the achievement of the current system state. Thereby, the relevance of the participating rules in a given schedule t is recurrently subject to the success of active rules in the following schedule $t+1$. Thus, a given rule strength is calculated as follows [40],

$$S_i(t+1)_n = S_i(t)_n - Bid_i + \frac{F_i}{\sum_{k \in M(t)} F_k} Sbid(t+1) \tag{34}$$

where $M(t)$ denotes the collection of indexers of activated rules at the schedule t and $Sbid(t)$ the sum of bids values,

$$Sbid(t) = \sum_{k \in M(t)} Bid_k \tag{35}$$

In this way, at every generation each particle/rule is associated a strength value that shows its contribution in the activations of other rules, $S_i(t_E)_n$ with t_E the number of total schedules to allocate all the jobs of the workload in generation n.

Finally, the total strength of rules is calculated as,

$$TS_{i,n} = S_{i,n} + S_i(t_E)_n \tag{36}$$

4.2.4. Individual's update operations

At every generation, the velocity of every rule i must be updated considering its previous velocity, $V_i(n)$, the best position found by the particle, $P_i^{\#}(n)$, and the best position found by the whole swarm, $P^*(n)$, up to generation n,

$$V_i(n+1) = \omega \otimes V_i(n) \oplus (d_1 * r_1) \tag{37}$$

$$\otimes (P_i^{\#}(n) \ominus P_i(n)) \oplus (d_2 * r_2) \tag{38}$$

$$\otimes (P^*(n) \ominus P_i(n)); \tag{39}$$

Once the velocity for every rule i has been updated, rules are modified,

$$P_i(n+1) = round[P_i(n) + V_i(n+1)] \tag{40}$$

Table 3 shows an example of particles, velocity and associated strengths and evaluation of particles at generation n for example in Table 2. Also, Table 4 presents an example of the update of velocities and particles for generation n.

4.2.5. Learning results

The learning of the FCS metascheduler with KARP is compared to the classical genetic strategy for classifiers discovery, Michigan approach. For both strategies several configurations are considered, as presented in Table 5.

Also, in KARP approach, the inertia weight follows a decreasing exponential expression in a way that global search is favored at first iterations and a deepest exploration of the reached locations is allowed at the end ones,

$$\omega(iter) = \omega_0 \cdot e^{(-iter/Num_{iter})c} \tag{41}$$

where *iter* represents the current step of the learning process, Num_{iter} is the total number of steps in the learning process and c is constant fixed to 5 in our simulations.

As stated before, the quality of the metascheduler is measured in terms of *makespan*. Fig. 6 represents the learning evolution for the FCS metascheduler with the proposed swarm-based learning

Table 3
Example of particles, velocity and associated strengths and evaluation of particles at generation *n*.

```
%%%%%% Particles, Velocities and Stregths at iteration n %%%%%%

%Particles at iteration n

P1=[1 −3 −1 0 −3 0 1 −4  2 ]
P2=[0 −1 −2 −1 −1 1 1 −4  2 ]
P3=[−3 0 0 0 −1 1 −1 −5  1 ]
P4=[2 −3 0 1 0 1 2 −3  2 ]
P5=[1 −1 −1 −1 −3 −1 −2 −1  2 ]
P6=[3 −1 1 0 −2 1 1 −5  2 ]
P7=[1 −1 −2 −1 −2 −2 1 −4  2 ]
P8=[2 −2 −1 −2 −1 0 1 −2  2 ]
P9=[−1 −2 0 −1 −1 0 −2 −3  2 ]
P10=[2 −1 −1 −1 0 1 −2 −1  1 ]


%Velocities at iteration n

V1=[1.00 1.00 −1.00 −1.00 0.35 −1.00 −1.00 −1.00 −1.00 ]
V2=[0.71 −0.54 1.00 −1.00 0.00 1.00 0.38 0.13 0.20 ]
V3=[1.00 0.30 0.00 −1.00 0.95 1.00 1.00 0.00 0.00 ]
V4=[−1.00 −1.00 0.00 −1.00 0.00 0.00 0.00 1.00 1.00 ]
V5=[0.33 1.00 −1.00 −1.00 −1.00 −1.00 0.00 −1.00 1.00 ]
V6=[−1.00 0.00 0.36 0.00 1.00 −0.72 0.00 1.00 0.06 ]
V7=[1.00 1.00 0.00 −1.00 0.25 0.00 −1.00 0.98 0.00 ]
V8=[−0.98 −0.87 0.00 1.00 −1.00 1.00 1.00  0.00 1.00 ]
V9=[1.00 −1.00 0.00 1.00 0.00 −1.00 −1.00 0.00 0.00 ]
V10=[1.00 −1.00 0.22 0.00 −1.00 −0.45 1.00 0.00 0.02 ]


%Best strength of particles up to iteration n.

S1= 6.6  %Associated to best particle/rule P#1
S2= 9.9  %Associated to best particle/rule P#2. Best Swarm Particle so far P*=P2
S3= 3.3  %Associated to best particle/rule P#3
S4= 5.0  %Associated to best particle/rule P#4
S5= 0.7  %Associated to best particle/rule P#5
S6= 1.7  %Associated to best particle/rule P#6
S7= 8.9  %Associated to best particle/rule P#7
S8= 5.0  %Associated to best particle/rule P#8
S9= 0.5  %Associated to best particle/rule P#9
S10= 2.1 %Associated to best particle/rule P#10

%Best particles found up to iteration n

P#1=[2 −3 −1 2 −3 0 1 4  1 ]
P#2=[0 −1 −2 −1 −1 1 1 −4  2 ]
P#3=[−2 0 0 0 −1 1 −1 −5  1 ]
P#4=[1 −3 0 1 0 −1 2 −3  1 ]
P#5=[−1 −1 −1 −1 −3 −1 −2 −1  1 ]
P#6=[−3 −1 1 0 −2 1 1 −5  2 ]
P#7=[−1 −1 −2 −1 −2 −2 1 −4  1 ]
P#8=[−1 −1 −1 0 −1 0 1 −2  1 ]
P#9=[1 −2 0 −1 −1 0 −2 −3  2 ]
P#10=[−2 −1 −1 −3 0 1 −2 −5  2 ]

%Best Particle of the Swarm up to iteration n
P*=[0 −1 −2 −1 −1 1 1 −4  2 ]



%%%%%% Rules Evaluation at iteration n %%%%%%

%Strength of particles at iteration n

S1= 2.4
S2= 5.1
S3= 10.3 %Best quality found for particle3, P3#=P3. Best Swarm Particle so far P*=P3
S4= 2.0
S5= 0.2
S6= 7.1  %Best quality found for particle6, P6#=P6.
S7= 6.5
S8= 9.1  %Best quality found for particle8, P8#=P8.
S9= 0.4
S10= 0.9


%Best strength of particles at iteration n.

S1= 6.6  %Associated to best particle/rule P#1
S2= 9.9  %Associated to best particle/rule P#2
S3= 10.3  %Associated to best particle/rule P#3
S4= 5.0  %Associated to best particle/rule P#4
S5= 0.7  %Associated to best particle/rule P#5
S6= 7.1  %Associated to best particle/rule P#6
S7= 8.9  %Associated to best particle/rule P#7
S8= 9.1  %Associated to best particle/rule P#8
S9= 0.5  %Associated to best particle/rule P#9
S10= 2.1 %Associated to best particle/rule P#10

%Best particles found at iteration n

P#1=[2 −3 −1 2 −3 0 1 4  1 ]
P#2=[0 −1 −2 −1 −1 1 1 −4  2 ]
P#3=[−3 0 0 0 −1 1 −1 −5  1 ]
P#4=[1 −3 0 1 0 −1 2 −3  1 ]
P#5=[−1 −1 −1 −1 −3 −1 −2 −1  1 ]
P#6=[3 −1 1 0 −2 1 1 −5  2 ]
P#7=[−1 −1 −2 −1 −2 −2 1 −4  1 ]
P#8=[−2 −2 −1 −2 −1 0 1 −2  2 ]
P#9=[1 −2 0 −1 −1 0 −2 −3  2 ]
P#10=[−2 −1 −1 −3 0 1 −2 −5  2 ]

%Best Particle of the Swarm at iteration n
P*=[−3 0 0 0 −1 1 −1 −5  1 ]
```

Table 4
Example of the update of velocities and particles for generation *n*.

```
%%%%%% Rules Update in iteration n %%%%%%

%Velocities in generation n+1

V1=w*V1+d1*r1*(P#1−P1)+d2*r2*(P*−P1)
V2=w*V2+d1*r1*(P#2−P2)+d2*r2*(P*−P2)
V3=w*V3+d1*r1*(P#3−P3)+d2*r2*(P*−P3)
V4=w*V4+d1*r1*(P#4−P4)+d2*r2*(P*−P4)
V5=w*V5+d1*r1*(P#5−P5)+d2*r2*(P*−P5)
V6=w*V6+d1*r1*(P#6−P6)+d2*r2*(P*−P6)
V7=w*V7+d1*r1*(P#7−P7)+d2*r2*(P*−P7)
V8=w*V8+d1*r1*(P#8−P8)+d2*r2*(P*−P8)
V9=w*V9+d1*r1*(P#9−P9)+d2*r2*(P*−P9)
V10=w*V10+d1*r1*(P#10−P10)+d2*r2*(P*−P10)


%Particles in generation n+1

P1=round(P1+V1)
P2=round(P2+V2)
P3=round(P3+V3)
P4=round(P4+V4)
P5=round(P5+V5)
P6=round(P6+V6)
P7=round(P7+V7)
P8=round(P8+V8)
P9=round(P9+V9)
P10=round(P10+V10)
```

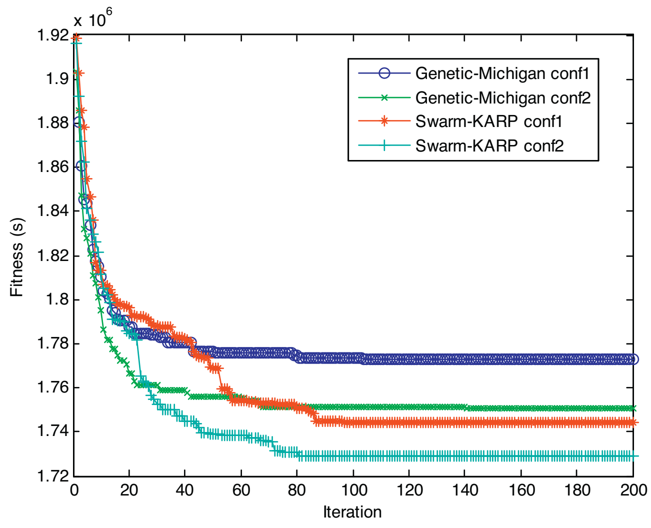

Fig. 6. Classifier system learning with genetic and swarm-based learning.

strategy and the classical genetic approach. As shown, the learning process for both types of classifier discovery systems is done for 200 iterations. Curves illustrate the average performance of the FCS metascheduler with the acquired knowledge at every step (40 runs) in the different configurations. It is observed that the initial convergence behavior of the FCS metascheduler with KARP in both configurations does not present any meaningful difference at early steps (approximately step 30) in comparison to the genetic strategy.

Nevertheless, from this step, it can be appreciated that KARP achieves a faster convergence. Specifically, the genetic-based approach requires 103 and 141 iterations (configuration 1 and 2, respectively) to reach its final fitness, in contrast to KARP that converges in 97 and 81 iterations (configuration 1 and 2, respectively). In this way, the best configuration of KARP obtains its convergence in 22 iterations less than the best configuration of best genetic strategy, which is translated into a computational effort reduction of 22 function evaluations (FEs) [41].

Regarding computational effort, it must be pointed out that optimizations algorithms are computationally rated generally according to the number of FE required [41,42]. Hence, since a FE in KARP is associated to a RB evaluation (as in Michigan approach), computational effort is measured interchangeably in RB evaluations or FE in this work. Specifically, in KARP the obtained RB is evaluated at every iteration and thus, computational effort for an experiment can be expressed as $CE_{\text{KARP}} = num_{\text{iter}}$, where num_{iter} is the number of iterations or stopping condition. In the same way, computational effort for Michigan approach is $CE_{\text{MICHIGAN}} = num_{\text{iter}}$ and therefore, both strategies can be fairly compared at each iteration in terms of computational effort. Also, regarding communication cost for the metascheduler to retrieve the state information of all the available RDs, it must be noted that communication cost of the FCS metascheduler is the same as any other state-based scheduling strategy for grid computing in the same scenario and using the same variables to describe the grid state [30,34,35]. The proposed FCS metascheduler using KARP is compared to the same FCS metascheduler using Michigan in the same conditions without considering communication cost. Communication cost is the same for both approaches and thus, the compared results are not affected by this network parameter.

Table 6 shows the performance of the classifier discovery strategies along several iterations, i.e., 50, 100 and 200 in 40 runs. Specifically, it presents the mean result achieved by the swarms/populations (average) with the associated standard deviation (standard deviation) and 95% confidence interval (confidence interval -95%) and the best result (Min). As shown, the performance of the classifier discovery system with KARP obtains better average convergence results (iteration 200) than the system with genetic learning. Furthermore, as expected from the discussion above, the better average results are significant from iteration 100 whereas there is not significant improvement in early iterations (50) of KARP over the genetic strategy in every configuration. Table 7 summarizes the relative improvement of KARP over the genetic strategy in diverse iterations. It is shown that the best KARP final results (iteration 200) improve the genetic strategy by 1.23% and 2.47% in its best and worst configuration (configuration 2 and 1, respectively). Moreover, it can be appreciated that KARP in its worst setting (configuration 1) is able to improve the best result of the genetic strategy (configuration 2) by 0.34% and reach an improvement of 1.60% (configuration 1). Additionally, as shown in Table 6, the best RB found with KARP learning outperforms the best genetic RB fitness (Min) by 0.66%, something that proves KARP ability to achieve a deeper search space exploration.

Also, statistical tests are conducted to further analyze the validity of results. First, the normality of data with Shapiro-Wilks test is studied for Michigan and KARP best final results. As shown in Table 8, *p*-values < = 0.05, then the null hypothesis that the samples came from a Normal distribution must be rejected.

Therefore, the data is analyzed using nonparametric tests. Also, Table 8 presents the nonparametric Wilcoxon or Mann-Whitney test [43]. An observation is made to compare the best results for Michigan and KARP. It relates the average of the fitness function in

Table 5
Parameters configuration for KARP and genetic approach.

Strategy	Configuration	Parameters configuration	
Swarm-KARP	(1)	ω_o =0.9	$d_1 = 2$, $d_2 = 2$
	(2)	ω_o =0.7	Swarm size (NP)=10
Genetic-Michigan	(1)	Selection rate λ=0.7	Mutation rate=$0.1e^{(-iter/Num_{iter})}$
	(2)	Selection rate λ=0.9	Population size (N)=10

Table 6
Learning strategies results for 40 simulations. Training fitness: Makespan.

Strategy	Configuration	Iteration	Average	Standard deviation	Confidence interval (95%)	Min
Swarm-KARP	(1)	200	1.7444037e + 06	4.5721233e + 04	1.7297814e + 06, 1.7590261e + 06	1.6481630e + 06
		100	1.7444037e + 06	4.5721233e + 04	1.7297814e + 06, 1.7590261e + 06	1.6481630e + 06
		50	1.7688881e + 06	4.1179042e + 04	1.7557184e + 06, 1.7820578e + 06	1.6842350e + 06
	(2)	200	1.7289457e + 06	5.0358125e + 04	1.7128403e + 06, 1.7450510e + 06	1.6372561e + 06
		100	1.7289457e + 06	5.0358125e + 04	1.7128403e + 06, 1.7450510e + 06	1.6372561e + 06
		50	1.7390166e + 06	5.1733520e + 04	1.7224714e + 06, 1.7555618e + 06	1.6372561e + 06
Genetic-Michigan	(1)	200	1.7727782e + 06	5.3704463e + 04	1.7556027e + 06, 1.7899537e + 06	1.6545054e + 06
		100	1.7732668e + 06	5.4210024e + 04	1.7559296e + 06, 1.7906040e + 06	1.6545054e + 06
		50	1.7763504e + 06	5.4268442e + 04	1.7589945e + 06, 1.7937063e + 06	1.6545054e + 06
	(2)	200	1.7504240e + 06	5.0117158e + 04	1.7343957e + 06, 1.7664522e + 06	1.6481630e + 06
		100	1.7510691e + 06	4.9404318e + 04	1.7352688e + 06, 1.7668693e + 06	1.6481630e + 06
		50	1.7561682e + 06	4.9401943e + 04	1.7403687e + 06, 1.7719677e + 06	1.6481630e + 06

Table 7
Relative performance of the learning strategies.

Comparison/iteration	50	100	200
Swarm-KARP 1 vs Genetic-Michigan 1 (%)	0.42	1.63	1.60
Swarm-KARP 1 vs Genetic-Michigan 2 (%)	−0.72	0.38	0.34
Swarm-KARP 2 vs Genetic-Michigan 1 (%)	2.10	2.50	2.47
Swarm-KARP 2 vs Genetic-Michigan 2 (%)	0.98	1.26	1.23

two different populations (Distribution 1 vs Distribution 2). For Distribution 1, the evidence against the null hypothesis is that the sum of the ranges in Distribution 1 (Michigan) is higher than that of Distribution 2 (KARP). Wilcoxon W parameter results in W value with a *p*-value, i.e., the evidence against the null hypothesis is significant with the obtained *p*-value. As it can be observed, the *p*-value < = 0.05. Therefore, it can be concluded that the improvement of KARP is statistically significant in comparison to Michigan. Hence, simulations show that KARP learning provides a faster and more accurate learning that the genetic approach in classifier discovery systems.

Finally, some observations are made about the comparison of KARP to genetic strategies. In many diverse previous works in literature [44,45] a wide range of configurations have been considered to test the performance of Michigan and Pittsburgh approach. It has been shown that Pittsburgh approach is able to achieve a greater accuracy than Michigan approach in most configurations. However, the associated computational effort is significantly more reduced for Michigan approach to achieve good results and thus, they can be convenient in many applications. In this work, the goal has been to provide an alternative learning strategy to Michigan approach for FCS requiring the same computational effort and improving its accuracy. An strategy, KARP, has been proposed that evolves rules as individuals as Michigan approach does, and that requires the evaluation of a RB at every iteration as Michigan approach requires too. However, the evolution of rules in not done through genetic operations as Michigan approach does but using PSO. Both strategies Michigan and KARP belong to the same type of fuzzy systems, FCS, and they can be fairly compared in terms of computational effort at every iteration. Results show that KARP has a greater accuracy than Michigan for FCS. Therefore, the main advantage of FCS using KARP or Michigan approach in comparison to Pittsburgh approach is its reduced computational effort. Since the computational effort of Michigan and KARP is it exactly the same at every generation and iteration, respectively, our previous works [18,39] comparing Michigan approach and Pittsburgh approach in grid computing fuzzy metaschedulers can be also considered to analyze the performance of KARP with regard to Pittsburgh in detail.

Table 8
Normality Shapiro-Will tests and Wilcoxon statistical tests for best results found for Michigan and KARP approach.

Data	Shapiro-Wilk W	*p*-Value
Swarm-KARP	0.8961	0.002299
Genetic-Michigan	0.928	0.02446

Observation (Distribution 1 vs Distribution 2)	Wilcoxon W	*p*-Value
Genetic-Michigan vs Swarm-KARP	808.5	0.03501

5. Conclusions

In this work, the application of swarm intelligence to rule discovery in FCSs has been proposed and a new strategy, KARP, has been presented. Specifically, KARP is based on PSO, a simple algorithm proving its efficiency for optimizing a wide range of functions based on the movement of swarms in the space towards best suited locations. In KARP approach each rule acts as particle that moves in the space with the final aim of finding the best location and thus, the best quality to cooperate in obtaining RBs for the experts classifiers systems. KARP is intended to be an alternative to the classic genetic approach for rule discovery in FCS, Michigan approach. To test the performance of KARP, a problem of practical importance nowadays, scheduling in grid computing has been selected. Specifically, the performance of FCS metaschedulers with both genetic and swarm-based strategies has been discussed. As simulation results show, KARP achieves an average greater accuracy (0.34–2.47%) and a faster convergence speed with the consequent reduction of computational effort. Moreover, one of the most relevant advantages of KARP over the genetic approach is related to its simplicity. KARP requires the use of a major operator, i.e., velocity, in contrast to the mutation, selection and crossover processes in the genetic approach. Hence, KARP is suggested as an efficient alternative for rule discovery in FCSs.

Acknowledgments

This work was financially supported by Research Projects CEATIC-2013-001, P2010-TIC-6762 and TEC2012-38142-C04-03. The Metacentrum workload log was generously provided by the Czech National Grid Infrastructure Metacentrum.

References

[1] H. Ishibuchi, T. Nakashima, M. Nii, Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining (Advanced Information Processing), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.
[2] M.-R. Akbarzadeh-T, M. Moshtagh-Khorasani, A hierarchical fuzzy rule-based approach to aphasia diagnosis, J. Biomed. Inform. 40 (5) (2007) 465–475.
[3] G. Schaefer, M. Závisek, T. Nakashima, Thermography based breast cancer analysis using statistical features and fuzzy classification, Pattern Recognit. 42 (6)

(2009) 1133–1137 (digital Image Processing and Pattern Recognition Techniques for the Detection of Cancer).
[4] M.F. Zarandi, A. Alaeddini, I. Turksen, A hybrid fuzzy adaptive sampling – run rules for Shewhart control charts, Inf. Sci. 178 (4) (2008) 1152–1170.
[5] E. Mansoori, M. Zolghadri, S. Katebi, Protein superfamily classification using fuzzy rule-based classifier, IEEE Trans. NanoBiosci. 8 (1) (2009) 92–99.
[6] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena, Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientific Pub. Co. Inc., 2001.
[7] J. Kennedy, R. Eberhart, Particle swarm optimization Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995.
[8] V. John, E. Trucco, S. Ivekovic, Markerless human articulated tracking using hierarchical particle swarm optimisation, Image Vis. Comput. 28 (11) (2010) 1530–1547.
[9] F. Alonso-Zotes, M. Santos-Peñas, Particle swarm optimisation of interplanetary trajectories from earth to jupiter and saturn, Eng. Appl. Artif. Intell. 25 (1) (2012) 189–199.
[10] K. Mahadevan, P. Kannan, Comprehensive learning particle swarm optimization for reactive power dispatch, Appl. Soft Comput. 10 (2) (2010) 641–652.
[11] S.M. Vieira, L.F. Mendonca, G.J. Farinha, J.M. Sousa, Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients, Appl. Soft Comput. 13 (8) (2013) 3494–3504.
[12] Y. Shi, R. Eberhart, E. Center, I. Carmel, Empirical study of particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Vol. 3, 1999.
[13] J. Robinson, Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, IEEE Trans. Antennas Propag. 52 (2) (2004) 397–407.
[14] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
[15] R.P. Prado, S. García-Galán, A.J. Yuste, J.E. Muñoz-Expósito, A fuzzy rule-based meta-scheduler with evolutionary learning for grid computing, Eng. Appl. Artif. Intell. 23 (7) (2010) 1072–1082.
[16] R.P. Prado, S.G. Galán, A.J. Yuste, J.E.M. Expósito, A.J.S. Santiago, S. Bruque, Evolutionary fuzzy scheduler for grid computing Lecture Notes in Computer Science, vol. 5517, Springer, 2009, pp. 286–293.
[17] R.P. Prado, S. García-Galán, J.E. Muñoz-Expósito, A.J. Yuste, Knowledge acquisition in fuzzy rule based systems with particle swarm optimization, IEEE Trans. Fuzzy Syst. 18 (6) (2010) 1083–1097, http://dx.doi.org/10.1109/TFUZZ.2010.2062525.
[18] S. García-Galán, R.P. Prado, J.E. Muñoz-Expósito, Fuzzy scheduling with swarm intelligence-based knowledge acquisition for grid computing, Eng. Appl. Artif. Intell. 25 (2) (2012) 359–375.
[19] L.B. Booker, D.E. Goldberg, J.H. Holland, Classifier systems and genetic algorithms, Artif. Intell. 40 (1–3) (1989) 235–282.
[20] E.H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, Proc. IEEE 121 (12) (1974) 1585–1588.
[21] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 15 (1) (1985) 116–132.
[22] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: Proceedings of the 7th International Conference on Evolutionary Programming VII, EP98, Springer-Verlag, London, UK, 1998, pp. 611–616.
[23] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of the 7th International Conference on Evolutionary Programming VII, EP '98, Springer-Verlag, London, UK, 1998, pp. 591–600.
[24] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[25] H. Liu, A. Abraham, A.E. Hassanien, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, Future Gener. Comput. Syst. 26 (2010) 1336–1343.
[26] J.H. Holland, Properties of the bucket brigade, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985, pp. 1–7.
[27] M.J. Gacto, R. Alcalá, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures, Inf. Sci. 181 (20) (2011) 4340–4360, http://dx.doi.org/10.1016/j.ins.2011.02.021.
[28] D. Klusacek, Dealing with uncertainties in grids through the event-based scheduling approach Fourth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2008), vol. 1, 2008, pp. 978–980.
[29] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1979.
[30] C. Franke, F. Hoffmann, J. Lepping, U. Schwiegelshohn, Development of scheduling strategies with genetic fuzzy systems, Appl. Soft Comput. 8 (1) (2008) 706–721.
[31] D. Klusacek, L. Matyska, H. Rudova, Alea – grid scheduling simulation environment, in: R. Wyrzykowski, J. Dongarra, K. Karczewski, Wasniewski, J (Eds.), Parallel Processing and Applied Mathematics, Vol. 4967 of Lecture Notes in Computer Science, Intel; Microsoft; IBM; Action SA; SIAM, Springer-Verlag Berlin, 2008, pp. 1029–1038.
[32] A. Sulistio, G. Poduval, R. Buyya, C.-K. Tham, On incorporating differentiated levels of network service into gridsim, Future Gener. Comput. Syst. 23 (4) (2007) 606–615.
[33] C.N.G. Infrastructure, Metacentrum data sets, 2009 http://www.fi.muni.cz/ xklusac/index.php?page=meta2009
[34] F. Xhafa, A. Abraham, Meta-heuristics for Grid Scheduling Problems, Meta-heuristics for Scheduling: Distributed Computing Environments, Studies in Computational Intelligence, Springer Verlag, Germany, 2008, pp. 978–983, ISBN: 978-3-540-69260-7.
[35] F. Xhafa, A. Abraham, Computational models and heuristic methods for grid scheduling problems, Future Gener. Comput. Syst. 26 (4) (2010) 608–621.
[36] A.A. Grosan, C.B. Helvik, Multiobjective evolutionary algorithms for scheduling jobs on computational grids, in: Proc. IADIS International Conference, Applied Computing 2007, Salamanca, Spain, 2007, pp. 459–463.
[37] S. Farzi, Efficient job scheduling in grid computing with modified artificial fish swarm algorithm, Int. J. Comput. Theory Eng. 1 (2009) 1793–8201.
[38] H. Izakian, B. Tork Ladani, K. Zamanifar, A. Abraham, A novel particle swarm optimization approach for grid job scheduling, in: S.K. Prasad, S. Routray, R. Khurana, S. Sahni (Eds.), Information Systems, Technology and Management, Vol. 31 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2009, pp. 100–109.
[39] R. Prado, S. García-Galán, A. Yuste, J. Expósito, Genetic fuzzy rule-based scheduling system for grid computing in virtual organizations, Soft Comput. Fusion Found. Methodol. Appl. 15 (2011) 1255–1271.
[40] C.-S. Joung, D.-W. Lee, K.-B. Sim, The fuzzy classifier system using the implicit bucket brigade algorithm, vol. 1, 1999, pp. 83–87, http://dx.doi.org/10.1109/IROS.1999.812985.
[41] Z.-H. Zhan, J. Zhang, Y. Li, H.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. B: Cybern. 39 (6) (2009) 1362–1381.
[42] G. Ciuprina, D. Ioan, I. Munteanu, Use of intelligent-particle swarm optimization in electromagnetics, IEEE Trans. Magnet. 38 (2) (2002) 1037–1040, http://dx.doi.org/10.1109/20.996266.
[43] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977, http://dx.doi.org/10.1007/s00500-008-0392-y.
[44] H. Ishibuchi, Book Review: "Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases" by Oscar Cordon, Francisco Herrera, Frank Hoffmann and Luis Magdalena; World Scientific, Singapore, New Jersey, London, Hong Kong, 2001, 462pp., ISBN 981-02-4016-3, Fuzzy Sets Syst. 141 (1) (2004) 161–162.
[45] O. Cordon, F. Herrera, P. Villar, Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base, IEEE Trans. Fuzzy Syst. 9 (4) (2001) 667–674.

S. García-Galán received the M.S. and Ph.D. degrees in Telecommunication Engineering from the Málaga University and the Technical University of Madrid (UPM), in 1995 and 2004, respectively. Since 1999, he is an associate professor at the Telecommunication Engineering Department of the Jaén University. He is a member of the research group "Signal Processing for Telecommunication Systems" (TIC-188 of the PAI) and the European Society for Fuzzy Logic And Technology (EUSFLAT). His areas of research interest are artificial intelligence, grid/cloud computing, speech and audio analysis. Also, he is reviewer of several journals indexed in JCR. He has been involved in research projects of the Spanish Ministry of Science and Education and private companies.

R. P. Prado received the M.S. degree in telecommunication engineering from Seville University, Seville, Spain, in 2008 and the Ph.D. degree in telecommunication engineering with European Mention from Jaén University, Jaén, Spain, in 2011. At present, she is an assistant professor with the Telecommunication Engineering Department of the Jaén University. Her current research interests include artificial intelligence, machine learning, grid/cloud computing and scheduling. She is an active member of the research group "Signal Processing for Telecommunication Systems" (TIC-188 of the PAI) and she forms part of the editorial board of Applied Soft Computing and reviewer board of diverse international JCR indexed journals such as IEEE Transactions on Fuzzy Systems, Soft Computing and Artificial Intelligence in Medicine.

J.E. Muñoz Expósito received M.S. degree in telecommunication engineering from Malaga University, Spain and Ph.D. in telecommunication engineering from Jáen University, Spain. Since 2003, he has been an associate professor at the Telecommunication Engineering Department of the Jaén University. His research interest include speech an audio analysis, artificial intelligence, grid and cloud computing. He is currently a senior researcher with the chair for "Signal Processing and Telecommunication Systems" (TIC-188 of the PAI). He is involved in research projects and forms part of editorial reviewer board of the Annual Telematics Engineering Conferences (JITEL).