Computing and Informatics, Vol. 43, 2024, 1285–1319, doi: 10.31577/cai_2024_6_1285

KNOWLEDGE ACQUISITION APPROACHES FOR VIRTUAL MACHINE MIGRATION IN CLOUD COMPUTING

Francisco Javier Maldonado Carrascosa*, Antonio Jiménez Sánchez, Doraid Seddiki, Sebastián García Galán, Manuel Valverde Ibáñez, Tomasz Marciniak

University of Jaén, Telecommunication Engineering Department
Av. de la Universidad S/N, 23700, Linares
Jaén, Spain
e-mail: fjmaldon@ujaen.es

Abstract. In recent years, the utilization of Cloud computing services has significantly increased, placing a heightened demand on the workload, computational infrastructure, storage, and communication networks managed by datacenters. This surge has prompted researchers to enhance Cloud performance, focusing on minimizing execution time and computational costs. To address this challenge, efficient scheduling strategies involve the allocation of Cloud workload across different datacenters through the migration of virtual machines. This article examines and compares three methodologies to tackle this issue: one based on an Adaptive Neuro-Fuzzy Inference System, another utilizing a Swarm Fuzzy System, and a third one employing a Genetic Algorithm. The study evaluates their effectiveness for workflow scheduling using a CloudSim-based simulator in terms of makespan and computational costs. Results reveal that the neuro-fuzzy system outperforms the fuzzy and genetic systems regarding makespan in Montage and CyberShake environments. It demonstrates a computational cost advantage, achieving reductions of 7.01 % and 6.33 % for KASIA and 10.74 % and 8.86 % for Pittsburgh in Montage and CyberShake, respectively. Furthermore, it surpasses the KASIA system by 50 % and Pittsburgh by 37.5 % in terms of the number of rule base evaluations.

Keywords: Fuzzy rule-based systems, swarm fuzzy systems, ANFIS, cloud computing, virtual machine migration, follow the renewable

* Corresponding author

Mathematics Subject Classification 2010: 03B52, 68M10, 68M20, 68T05, 68T30, 68T37

1 INTRODUCTION

Fuzzy Rule-Based Systems (FRBS) are gaining attention in Cloud computing for large-scale scheduling due to their capability to manage uncertain information in highly dynamic systems. This paper examines the benefits of using fuzzy systems as schedulers to enhance allocation processes, with a focus on improving makespan and efficiently distributing workloads across multiple datacenters [1]. By emulating human-like reasoning, FRBS offer promising approaches to address uncertainty challenges in Cloud computing environments.

In this context, fuzzy systems serve as expert systems by integrating expert knowledge into their Knowledge Bases (KBs) [2]. Their effectiveness relies heavily on the quality of knowledge representation within their KBs. However, these systems face limitations in generating new knowledge or improving performance for complex and challenging problems. To address this, automated methods for knowledge generation in FRBSs have emerged, focusing on self-learning strategies like genetic approaches. The Michigan approach [3] treats rule encoding as individuals, while the Pittsburgh approach [4] sees the Rule Base (RB) as individuals. Recently, particle swarm optimization (PSO)-based strategies have been used [5, 6]. Notably, Knowledge Acquisition with a Swarm Intelligence Approach (KASIA) [7] and Knowledge Acquisition with Rules as Particles (KARP) [8] outperform genetic strategies in certain scenarios. This study introduces a new comparison of these techniques in a novel scenario.

Nowadays, cloud computing services are widely adopted across diverse domains due to their multidisciplinary nature. However, the proliferation of digital services has significantly increased datacenter workloads, making efficient scheduling crucial for minimizing computational costs. In this sense, integrating Artificial Intelligence (AI) techniques has emerged as a promising approach to improve cloud performance. As discussed earlier, various AI techniques, as demonstrated in [9], have been extensively studied in different fields. Therefore, given that cloud computing schedulers rely on the quality of KBs and the learning process, evaluating the effectiveness of these strategies in terms of total execution time and computational effort is essential.

This study compares three methodologies, highlighting the paper's contribution to migrate virtual machines (VM) in cloud data centers (CDCs) through scheduling. The techniques include KASIA, Adaptive Neuro-Fuzzy Inference System (ANFIS), and Pittsburgh approaches. To enhance the learning acquisition process, the paper introduces an ANFIS-based methodology, employing a Fuzzy Inference System (FIS) implemented through adaptive networks [10]. ANFIS, an Artificial Neural Network (ANN) that combines neural networks and fuzzy logic principles, has found applications in various fields such as audio/speech discrimination and cloud computing [11].

To overcome the physical and economical challenges of implementing a real scenario, the study uses a simulator based on WorkflowSimDFVS to evaluate the three methodologies [12]. Testing involves diverse scenarios with varying complexities using a simulator that combines CloudSim [13] and WorkflowSim [14] features for efficient workload allocation across multiple datacenters. The cloud simulator incorporates Pegasus workflow structures, like Montage or CyberShake, using real workflow traces to replicate a cloud system [15]. Moreover, VM migration is included to facilitate task distribution among CDCs.

1.1 Motivation and Contributions

In today's ever-changing technological landscape, continuous innovation is vital for achieving efficiency and resource conservation in various domains. In computing systems and resource management, the critical goal is to optimize computational cost, makespan, and energy usage [16] while minimizing environmental impact.

Reducing computational costs is essential in diverse fields such as cloud computing, data analysis, and scientific simulations, offering financial savings and progress in resource-constrained environments like cloud, fog, edge, and IoT computing [17]. Efficient resource use, through improved algorithms, allocation, and parallel computing, is crucial. Makespan, reflecting task completion time, is key, and intelligent scheduling, using machine learning and real-time analysis, minimizes makespan, enhancing overall system efficiency with consideration for task dependencies and dynamic workloads [18]. Addressing energy consumption is imperative in computing [19], and employing energy-efficient techniques with renewable sources sustains computing while reducing energy use. Achieving these goals involves exploring emerging technologies such as cloud, fog, edge computing, AI, and quantum computing [20]. Integrating these technologies produces innovative solutions for resource optimization, driving advancements, unlocking potential in resource-constrained environments, and promoting sustainability. This document specifically aims to optimize energy sustainability in CDCs using renewable energy.

In this paper, a significant contribution to the field of VM migration in CDCs is made by introducing an innovative approach that leverages ANFIS within the CloudSim simulator. The primary motivation of this research stems from the imperative need to enhance VM migration efficiency in the context of CDCs, where resource optimization and computational cost reduction are paramount. The novelty of this work lies in the application of ANFIS to VM migration, demonstrating superior performance when compared to existing strategies. Importantly, the findings reveal that ANFIS achieves notable advantages, not only in terms of migration efficiency but also in configuration simplicity, requiring fewer parameters. This streamlined configuration results in reduced computational costs and resource consumption, making ANFIS an optimal strategy for VM migration in CDCs in the proposed scenarios. Our comprehensive comparison establishes ANFIS as a robust and resource-efficient solution, thereby contributing to the ongoing discourse on optimizing cloud infrastructure through intelligent VM migration strategies.

The structure of the paper is as follows: the background reviews previous relevant work, providing a foundation. Section 3 discusses meta-scheduling systems and the use of fuzzy reasoning in KASIA, Pittsburgh, and ANFIS. Section 4 evaluates these systems in cloud computing meta-schedulers in terms of execution time and computational cost. Finally, Section 5 draws conclusions.

2 BACKGROUND

In recent years, the surge in cloud computing services has been substantial, providing numerous computing resources over the Internet [21]. Recognized for their crucial role in critical situations, these services have propelled job scheduling to a prominent area of research. However, effective job scheduling demands evaluation across various components, presenting challenges and potential failures within the cloud computing system. Contributing factors include the use of VMs for managing large-scale workloads and virtualizing cloud resources [22], VM migration addressing resource inadequacies and optimizing host utilization [23], and inherent uncertainty in the cloud environment. This uncertainty stems from the unpredictable nature of incoming tasks and the limited information offered by service providers. Nevertheless, existing solutions heavily rely on models incorporating prior job and resource information [24]. Recent focus on energy-efficient allocation and parallel scheduling adds complexity to job scheduling, making it a critical issue for ensuring optimal cloud performance.

Various strategies, including classic techniques and AI methods, have successfully addressed the scheduling problem [21]. Classic approaches involve dynamic methods like DHRA, Min-Min, FCFS, Round Robin, and Fussy Clustering, as well as predictive techniques such as energy-aware resource provisioning frameworks and AI-based algorithms like genetic fuzzy systems (GFS) [2]. Notably, the Michigan and Pittsburgh approaches, utilizing FRBSs, excel in discovering near-optimal solutions. Fuzzy systems, recognized for their adaptability, have become prominent in scheduling research, proving effective in dynamic and uncertain scenarios [25]. FRBS systems find applications in various domains, including serving as core algorithms for meta-scheduling in cloud computing simulations [1]. Fuzzy sets, introduced by Zadeh [26], are a natural choice for modeling human reasoning in these systems, as evidenced in studies related to COVID-19 vaccine selection [27, 28].

In this context, rules are defined using if-then structures with fuzzy antecedents and a consequent. Systems that use multiple inputs and a single output are called Mamdani [29] or Sugeno [30] systems and have their own specific formulas. In these formulations, an input variable, fuzzy sets associated with the input parameters for each rule, fuzzy sets corresponding to the output variable and AND/OR used as the connectives for antecedents are employed. The quality of RBs is crucial for successful performance in FRBS and numerous studies have focused on evaluating the quality of RBs to improve their overall performance.

The quality of expert knowledge in FRBSs is crucial, leading to the exploration of new machine learning techniques. Reinforcement learning techniques optimize or generate KBs in FRBS, combining classical engineering techniques' accuracy and interpretability with AI capabilities. This paper makes use of a swarm fuzzy system, a concept from [8]. Swarm Intelligence (SI), a popular form of AI, is widely used in various engineering applications, with PSO being an SI-based optimization technique leveraging social behavior of particles in a multidimensional search space [31]. This study considers the KASIA technique, adapting PSO to RB evolution, due to its strong performance. Despite achievements, there is a need for new approaches offering better performance and faster convergence in the learning process. Thus, this study proposes comparing ANFIS, which uses neural networks to obtain RB, with KASIA and Pittsburgh approaches in cloud computing task scheduling scenarios to evaluate effectiveness.

2.1 Relevant and Related Works

KASIA, applying PSO to RB evolution, treats each RB as a particle in a swarm, striving for optimal positioning. Initially introduced and extensively studied in [7], KASIA outperformed conventional learning methods such as the Pittsburgh approach and Q-learning-based strategies, as well as other scheduling approaches like EASY-backfilling or ESG+local periodical search, excelling in both computational efficiency and outcomes, particularly in convergence handling and simplicity. Its success transcended to grid computing job scheduling, surpassing six queue-based and scheduling traditional methodologies, specifically in the field of meta-scheduling for job assignment in computer networks. In the realm of fuzzy classifier systems, KASIA exhibited superior convergence rates and higher quality by reducing the number of control parameters in comparison to GFS [8]. Ongoing research delves into KASIA's application in cloud computing, with a focus on sustainable practices in task scheduling and VM migration, highlighting its versatility and adaptability and seeking more sustainable cloud computing practices.

The Pittsburgh genetic approach, rooted in natural evolution, is a notable evolutionary computational technique gaining attention in computational intelligence. Its efficacy in solving complex optimization problems is well-documented, notably accelerating classification rule discovery using GPUs with extensive datasets like hepatitis C, poker hands, and COVID-19 [32]. Researchers employ this approach in various contexts, including multi-objective genetic algorithms for fuzzy classification rule mining, enhancing accuracy and interpretability through fitness inheritance [33]. It also contributes to the development of genetic-based fuzzy classifiers [34, 35]. Recent applications involve using the Pittsburgh genetic algorithm for designing a genetic fuzzy tree-based node moving strategy in multi-modal wireless sensor networks for target tracking [36]. Additionally, a novel algorithm combines the Pittsburgh approach and the PSO algorithm for rule mining purposes [37].

ANFIS proves effective in diverse fields, estimating unknown characteristics in applications such as fluid flow, where it predicts missing velocity vectors using particle image velocimetry [38]. In healthcare, ANFIS detects breast cancer through mammography image classification and early detection of cardiovascular diseases via heartbeat sound classification [39, 40]. Its versatility extends to modern areas like smart homes and cloud computing, stabilizing clusters in cognitive radio networks for enhanced stability [41]. ANFIS optimizes the response time of a PID controller in an Eddy Current Dynamometer [42] and aids in hybrid multi-objective optimization for tapping center machines [43]. In smart homes, an ANFIS-based model significantly reduces processing time for risk estimation in the IoT risk-based control framework [44]. ANFIS excels in computer state evaluation, analyzing CPU usage and computer part behavior [11]. It further contributes to an urban environmental noise monitoring system in smart cities [45]. ANFIS, with its fusion of neural networks and fuzzy logic, emerges as a vital tool for addressing complex problems in technologies like cloud computing [46, 47, 48, 49].

3 PROPOSED META-SCHEDULERS

This section presents the fuzzy reasoning-based systems employed in the proposed meta-schedulers. These systems are used to classify jobs and assign them to different CDCs, allowing two stages of scheduling: the central scheduler assigns jobs to the datacenters, and the datacenters further distribute the jobs among their computing machines. Initially, the meta-scheduler is presented as a FRBS system, which is suitable for classification based on previous literature. Additionally, a neuro-fuzzy system is considered, benefiting of an ANN for classification purposes.

The goal of rule-based meta-schedulers is to assign a set of tasks to various CDCs within a virtual organization. Each datacenter comprises multiple computing machines and local schedulers responsible for the efficient distribution of tasks among these machines. Specifically, this study evaluates schedule performance from a global system perspective, focusing on makespan.

In this context, the proposed meta-schedulers use fuzzy reasoning to process the available information within the system, with the aim of obtaining a more complete understanding of the environment. Considering that the scheduling process can benefit from certain cloud features, in this work several variables are taken into account in this work to describe the actual cloud conditions. The characteristics of each data center are influenced by the current state of the resources and how their performance evolves over time. The scheduler goal is to improve the efficiency of cloud systems in terms of makespan, which serves as the performance metric during the learning process.

The next step is to define fuzzy sets for the system variables and learn their relationships within the fuzzy rules. Each rule, denoted as R_i, consists of antecedents and a consequent. The antecedents represent the rule activation, while the con-

sequent indicates the output contribution towards the selection of the datacenter DC_j. The rules use Mamdani or Sugeno encoding, which can be expressed by the following formulation:

$$R_i = \text{IF } x_i \text{ is } A_{i,1} \text{ AND/OR } \ldots \ x_n \text{ is } A_{i,n} \text{ THEN } y \text{ is } B_i, \tag{1}$$

where $(x_i, \ldots, x_n)$ represent the input features, while $A_{i,n}$ and B_i denote the respective fuzzy sets for feature x_m and output of rule i. It is noteworthy that the knowledge is encoded in a RB, which consists of a set that comprises all the fuzzy rules that follow the aforementioned structure.

3.1 Knowledge Acquisition with a Swarm Intelligence Approach

In this subsection, Figure 1 illustrates the organization of the KASIA meta-scheduler, where the reasoning stage is indicated by the KB and its composition plays a crucial role in ensuring the successful performance of the scheduler. The entire structure is organized as follows:

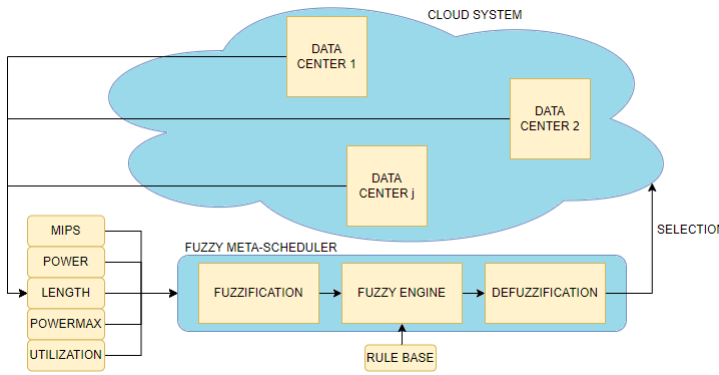


Figure 1. FRBS-based meta-scheduling system

Cloud system: This system is composed of multiple datacenters that contain local schedulers and computing machines. It generates the information to be processed by the scheduler.

Fuzzy Rule-Based Scheduling System: This stage involves the processing of inputs by the scheduler using fuzzy rules. Within this stage, several steps are performed to obtain the output and select the optimal datacenter for task execution. These steps include:

Fuzzification: This process considers the inherent uncertainty of crisp input values and transforms them into fuzzy sets that incorporate the environmental conditions.

Fuzzy engine: For each fuzzy rule, the input fuzzy sets of the antecedent are transformed into an output fuzzy set.

Defuzzification: The output set of each rule is aggregated into an overall output fuzzy set, and a crisp value y is determined for the entire RB.

Input Features: These variables represent the characteristics of the meta-scheduler that influence the selection of the optimal datacenter. These features will be discussed in Section 4, along with the output variable.

Rule Base: An RB consists of if-then rules that express fuzzy control statements related to the linguistic values of system variables. The inference engine uses the RB to make decisions during the reasoning process. An example of a RB will be presented in the results section.

The fuzzification and defuzzification stages use membership functions to facilitate decision-making in the scheduling process. In this study, Gaussian functions are employed to define fuzzy sets due to their efficiency and simplicity. The formulation of the Gaussian function is as follows:

$$\mu_i^{X_m} = \frac{1}{\sqrt{2\pi\sigma_i^{2X_m}}} \exp\left(\frac{-(z-\tau_i^{X_m})^2}{2\sigma_i^{2X_m}}\right), \quad \{z \in \mathbb{R}, z \leq 1\}, \tag{2}$$

where $\tau_i^{X_m}$ and $\sigma_i^{X_m}$ denote the mean and the standard deviation, respectively, the variable z represents the independent variable that describes the feature and m corresponds to the current feature. The selection of Gaussian functions is motivated by the advantageous properties they possess. First, their inherent differentiability and smoothness make Gaussian functions advantageous in optimization algorithms, promoting stable learning and convergence. Second, their mathematical simplicity, characterized by only a few parameters, facilitates implementation and analysis. Third, the universal approximation property enables Gaussian functions to adeptly capture intricate data relationships. Fourth, their symmetry and peak localization contribute to accurate modeling of fuzzy sets. Fifth, the non-zero membership across the entire range ensures comprehensive coverage of the universe of discourse. Sixth, the statistical interpretation and connection to normal distributions provide a valuable probabilistic perspective. Seventh, computational efficiency, including separability, enhances performance in large-scale or real-time applications. Lastly, the intuitive interpretability of Gaussian function parameters facilitates practical tuning and understanding. Together, these factors underscore the suitability of Gaussian membership functions for this research, aligning with the work's goal of effective and versatile fuzzy inference. In a fuzzy system, each value of a system variable is associated with a membership degree related to a fuzzy set. Typically, normalized values ranging from 0 (complete exclusion of the variable) to 1 (full membership of the variable) are assigned, with intermediate values indicating partial membership in the set.

In KASIA, fuzzy RBs are treated as knowledge individuals that undergo evaluation and evolution using SI. Specifically, KASIA adopts the PSO algorithm as the basis for acquiring fuzzy RBs. Then, the system operates based on the social behavior exhibited by interacting individuals within a swarm. The standard procedure begins with the generation of an RB swarm comprising several particles. Each particle, represented by a matrix P, consist of multiple rows corresponding to individual fuzzy rules within their respective RB particle. A particle P is composed of antecedents, consequents, and connectives. Every rule follows the Mamdani coding and is made up of the previous antecedents, consequents, and connectives. Particle initialization involves assigning a velocity (V) to each particle, resulting in rule modifications at each step of the algorithm. Next, the process proceeds to determine the best position for each particle as an individual and for the entire swarm. In this sense, KASIA evaluation is equivalent to the evaluation of an RB. As stated in [7], there are some restrictions for the algorithm variables that are considered.

Once the variables are defined and evaluated, the algorithm velocity and position of the particles must be updated based on both individual and social experiences. The velocity update considers the inertia of the particle ω, the best RB obtained by the particle (P^B), and the best position found by the swarm (G^B). The velocity update is formulated as:

$$V(t+1) = \omega V(t) + (c_1 r_1)(P^B(t) - P(t)) + (c_2 r_2)(G^B(t) - P(t)), \tag{3}$$

where c_1 and c_2 are weight factors that remain constant and represents the competitive and cooperative social factors, respectively, and r_1 and r_2 are random numbers within the interval $[0, 1]$. The update of the particle's position is formulated as follows:

$$P(t+1) = P(t) + V(t+1). \tag{4}$$

The KASIA algorithm presented in Algorithm 1 is a metaheuristic optimization algorithm that uses a swarm of particles to solve a given problem. It starts by initializing the swarm with random positions and velocities. The algorithm then iteratively updates the particle positions and velocities, evaluating their fitness based on a predefined objective function, in this case, the makespan. The best positions found by the particles are stored as the global best and particle best. After a certain number of iterations, the algorithm returns the global best position, which represents the best solution found. Overall, KASIA combines the exploration of the search space by updating particle positions and exploitation of promising regions to find an optimal or near-optimal solution.

3.2 Pittsburgh Approach

The Pittsburgh genetic approach integrates evolutionary principles, fuzzy reasoning, and a population-based framework to effectively optimize complex problems such as task scheduling in cloud computing. In this approach, the genetic system employs a set of entire RBs as chromosomes. Consequently, each genetic individual represents

Algorithm 1 KASIA algorithm

```
Swarm Inizialization: NParticles, NRules, NIter, InertialWeight ω, c1 and c2 factors.
Random setting of RB-Swarm Position P.
Random setting of velocity V.
Position and velocity constraints.
Initialize G^B and P^B.
while NIter do
    while NParticles do
        Update P.
        Apply constraints to P.
        Evaluate fitness (makespan).
        Particles++.
    end while
    Update G^B.
    while NParticles do
        Update P^B.
        Update V.
        Apply constraints to V.
        Particles++.
    end while
    iter++.
end while
Return: G^B
```

a complete RB and a population of RBs evolves by applying genetic operators. At the end of the optimization process, the best RB is selected to be used by the Fuzzy System. The general schema of the Pittsburgh approach is illustrated in Figure 2. In each generation, all RBs within the population undergo evaluation within the operating environment of the Fuzzy System. This evaluation is facilitated by the Evaluation System, which determines the quality of each RB based on the fitness value obtained within the environment being optimized. Following the ranking of RBs based on their quality, those with higher rankings are crossed, resulting in the derivation of new RBs within the RB discovery system. Additionally, these higher-ranked RBs are subject to mutations. Simultaneously, the least favorable RBs within the generation are replaced by the newly derived RBs and the replacement ratio is determined by the replacement rate. Following this approach, RBs with improved fitness in terms of the selected criteria are obtained across generations, which ultimately leads to the selection of the best RB to serve as the RB of the Fuzzy System in the final generation.

As previously mentioned, the rule-discovery and RB-discovery components in these machine learning techniques rely on genetic algorithms. However, in this study, these components are based on SI. In this approach, the RB serves as the

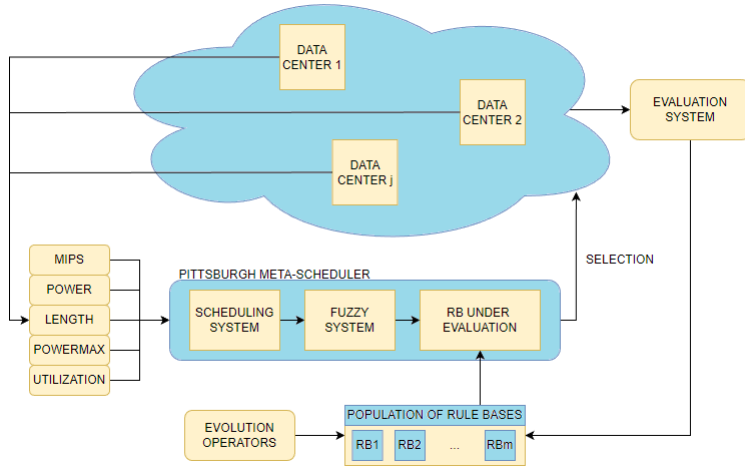


Figure 2. Pittsburgh Fuzzy Meta-Scheduler

individual within the genetic system, encoding a complete RB and operating within a population of Fuzzy RBs. The organization of the prototype for this approach can be summarized as follows:

Performance System: When employing this approach, the GFS evolves populations of RBs, and the fitness function, specifically the makespan, is computed for each individual. Consequently, the cooperation between the fuzzy rules comprising the RB is evaluated, allowing for efficient evolution of the population to identify the RB with the most effective fuzzy rule cooperation.

Rule-Base Discovery System: Following the usual genetic approach, four genetic operators are considered for the generation of new rules:

Selection: To ensure that the best RBs are not eliminated or replaced, elitism is chosen as the selection mechanism. Consequently, within the selection process, a number of parents, fewer than the initial number of RBs, are selected. After undergoing crossover and mutation, the new population replaces the least suitable RBs from the original set.

Crossover and Random Mutation: In this approach, standard two-point crossover is applied, allowing the number of rules in each child individual to deviate from their parents' original values. This introduces further variability in the cooperative behavior of the rules. Random mutation is employed to maintain genetic diversity, explore the search space and increase the chances of finding optimal or near-optimal solutions to the given problem.

Replacement: This work uses full replacement, in which the offspring completely replaces the parent population, ensuring a fresh start for each generation. This approach allows for a more radical exploration of the search

space since all individuals are replaced simultaneously, potentially leading to faster convergence towards better solutions. However, it also eliminates the possibility of preserving good solutions from the previous generation. Full replacement is commonly used in simple genetic algorithms and can be effective when the population size is small or when the search space is highly dynamic, requiring frequent exploration of new areas.

Algorithm 2 Pittsburgh algorithm

Inizialization: N_Population, N_Rules, N_Iter, Crossover_Rate,
Mutation_Rate_Init, Selection_Rate, Replacement_Rate.
Random setting of RB Population P.
Initialize G^B.
while N_Iter **do**
    Update Mutation_Prob with Mutation_Prob = Mutation_Rate_Init $*$ $\exp\left(\frac{-5t}{N\_iter}\right)$
    while N_Particles **do**
        Evaluate fitness (makespan).
        Particles++.
    end while
    Update G^B.
    while N_Particles·Replacement_Rate **do**
        Generate Q offspring:
        Apply crossover to P.
        Apply mutation to Q.
        Apply constraints to Q.
    end while
    Update part of P with Q.
    iter++.
end while
Return: G^B

The Pittsburgh algorithm iteratively evolves a population of individuals by applying crossover and mutation operations to generate offspring. It evaluates the fitness of individuals, updates the global best solution and replaces a portion of the population with the offspring. The algorithm aims to find an optimal or near-optimal solution for the given optimization problem by iteratively improving the population over multiple iterations.

3.3 Adaptive Neuro-Fuzzy Inference System

ANFIS is based on the aforementioned fuzzy-based structures but incorporates adaptive networks. An adaptive network is composed of interconnected nodes and directional links, where the outputs of the nodes depend on their respective parameters.

The network consists of multiple adaptive nodes and learning rules control the adjustment of these parameters to minimize an error measure. In the case of ANFIS, a hybrid learning rule is employed, combining the gradient descent-backpropagation method with the least squares estimate (LSE), thereby enhancing the learning process and determining the optimal parameters. This decision is motivated by the fact that relying solely on the gradient descent method could lead to slow performance and potential convergence to local minimum, which may negatively impact the overall performance of ANFIS.

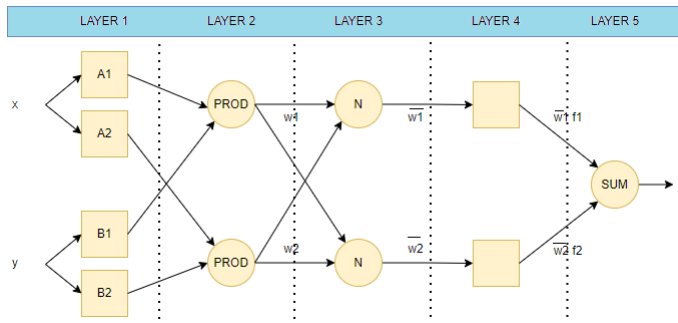


Figure 3. ANFIS architecture

To simplify the description, a simple ANFIS architecture can be represented with two inputs, x and y, and one output, z. The RB is designed to incorporate Takagi and Sugeno-type fuzzy if-then rules. In this particular scenario, the fuzzy reasoning category depicted in Figure 4 corresponds to the ANFIS architecture illustrated in Figure 3.

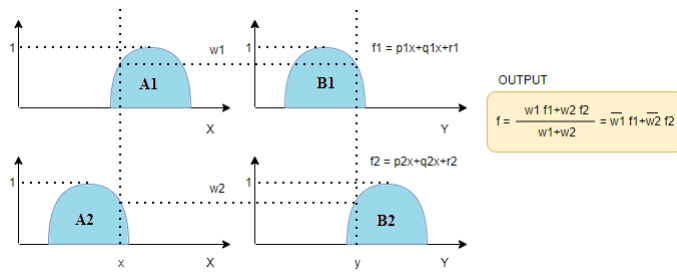


Figure 4. Equivalent fuzzy reasoning

As described below, nodes within the ANFIS architecture are organized into distinct layers as in [50, 51], each serving a specific function:

Layer 1: In the fuzzification layer of the ANFIS, neurons represent fuzzy sets, and the generation of fuzzy clusters from input data is facilitated through the uti-

lization of membership functions. These functions, including triangular, trapezoidal, bell, Gaussian, sigmoidal, among others, play a crucial role in determining the membership values in this layer. The shape of these membership functions is defined during ANFIS training, guided by a set of antecedent parameters (a_i, b_i, c_i). The output of a node in this layer corresponds to the membership degrees associated with each employed membership function. Notably, this layer is characterized by input membership functions, each distinguished by its set of premise parameters, predominantly triangular or bell-shaped. In the specific context of our work, Gaussian functions are applied, underlining their relevance in capturing the inherent complexities of the data for enhanced fuzzification in the ANFIS model. Each node i is a square node with the following node function:

$$O_i^1 = \mu_{A_i}(x). \tag{5}$$

Layer 2: In the second layer, referred to as the product or rule layer, the representation of the firing strength of rules takes center stage. Each node within this layer serves as an indicator of the intensity of a specific rule. The layer plays a pivotal role in determining the firing strengths associated with each rule, leveraging the membership values acquired in the preceding fuzzy layer. Here, every node remains fixed, and its output is a direct reflection of the incoming signals. The operation predominantly employed in this layer is the AND fuzzy operation, contributing to the integration and evaluation of the combined rule strengths within the ANFIS framework. The output of this layer is the below function:

$$O_i^2 = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y). \tag{6}$$

Layer 3: In the subsequent layer, aptly named the normalization layer, the output of each node takes the form of normalized firing strengths. This nomenclature is attributed to the computation of the proportion of the firing strength related to a specific rule in relation to the total firing strengths. The layer consists of non-adaptable nodes, each playing a crucial role in the normalization process. At the j^{th} node within the network, the firing strengths undergo normalization by determining the proportion of the firing strengths at that particular node concerning the collective firing strengths of all rules. This normalization layer is pivotal in refining the outputs from the preceding layer, ensuring that the ANFIS model accounts for the relative contributions of individual rules in the decision-making process. The output of this layer is:

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2}. \tag{7}$$

Layer 4: In this designated layer, recognized as the defuzzification layer, the focus shifts to the evaluation of output membership functions and their associated parameters, termed as consequent parameters. This layer plays a pivotal role

in assessing the output of each rule, achieved through the multiplication of normalized firing strengths by a linear polynomial. The set of coefficients within this polynomial, denoted as p_i, q_i, r_i, encapsulates the consequential parameters essential for training the ANFIS approach. The defuzzification layer acts as a critical bridge between the normalized rule strengths and the ultimate output of the ANFIS model, providing a mechanism for translating fuzzy logic evaluations into crisp, actionable results through the manipulation of consequent parameters. The output fruit of the defuzzification is:

$$O_i^4 = \overline{w}_i \cdot f_i = \overline{w}_i \cdot (p_i x + q_i y + r_i). \tag{8}$$

Layer 5: In the final stage, denoted as layer 5 or the output layer, the ANFIS model culminates in the computation of the ultimate output. This layer features a singular non-adaptable node tasked with the responsibility of receiving outputs from the preceding defuzzification layer and aggregating them. The calculation of the final output involves a straightforward addition of all incoming signals, synthesizing the refined information processed throughout the ANFIS network. This output layer serves as the conclusive phase of the model, consolidating the refined rule evaluations into a cohesive and comprehensive result that encapsulates the informed decision or prediction generated by the ANFIS framework. The output of the ANFIS is the following:

$$O_i^5 = \sum_i \overline{w}_i \cdot f_i = \frac{\sum_i w_i \cdot f_i}{\sum_i w_i}. \tag{9}$$

The output of layer 5 can be viewed as a Sugeno FIS model which performs the KB computation and is subsequently defuzzified to produce a crisp output representing the selection of the proper datacenter. It is noteworthy that Sugeno output membership functions are linear. The ANFIS Sugeno model, depicted in Figure 5, illustrates how ANFIS processes input features and generates an output, similar to a FIS. Therefore, the model depicted in the figure replaces the fuzzy meta-scheduler in the global system, effectively functioning as a FIS. As a result, the adaptive network in ANFIS exhibits functional equivalence to FISs. In light of this, this paper aims to compare KASIA, Pittsburgh and ANFIS in terms of makespan for task classification in a meta-scheduling environment within cloud computing.

The ANFIS algorithm iteratively optimizes a FIS by generating an initial FIS, performing forward and backward propagation, updating the FIS parameters, and minimizing the error through the LSE estimation. It evaluates the fitness of the solution (makespan), updates the global best solution, and continues the iterations to improve the FIS. The algorithm aims to find an optimal or near-optimal solution for the given problem by iteratively refining the FIS over multiple iterations.

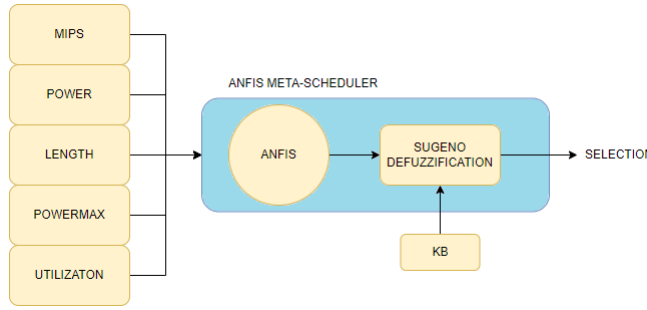


Figure 5. ANFIS meta-scheduler

Algorithm 3 ANFIS algorithm

Inizialization: N_Rules, N_Iter, N_epochs.
Random setting of RB Population P.
Initialize G^B.
while N_Iter **do**
  Generate Initial FIS.
  Perform forward propagation.
  Perform backward propagation.
  Update P parameters using back-propagation method.
  Apply LSE to minimize error and update the membership function parameters.
  Evaluate fitness (makespan).
  Obtain Output FIS.
  Update G^B.
end while
Return: G^B

4 EXPERIMENTAL RESULTS

Given the need to evaluate the state of cloud computing and the challenges associated with conducting large-scale deployments in the real world, the use of simulators that can compute data and address social problems has been proposed. To evaluate the effectiveness of the proposed methodologies, a series of experiments were carried out using various meta-schedulers based on fuzzy reasoning for cloud computing. For this, a simulator that incorporates CloudSim and WorkflowSim features was used, allowing the use of real traces of existing installations. The simulator used in this study, as described in [12], incorporates Dynamic Voltage Frequency Scaling (DVFS), a prominent power-saving strategy for hosts. Simulator validation was performed by evaluating utilization, frequency and voltage scaling, power consumption, energy usage, and time-saving. This open source simulator facilitates energy-aware optimization and analysis of actual workflow processing. To recreate real traces of the workflow within the simulator, complex scenarios based on projects such as

Montage, Sipht, NASA's Inspiral, and the Southern California Earthquake Center's CyberShake were considered. Specifically, this paper focused on the Montage and CyberShake projects [52, 53].

In the context of scientific workflows, a workflow is defined as a collection of processing jobs interconnected in a specific way to perform a given task. Characterizing a workflow involves identifying individual processing jobs, which entails dividing the entire process into subprocesses called jobs or chunks. Additionally, each job must be associated with other jobs, determining the job dependencies, the jobs that must be executed before each job, and the input and output data required for each job. For this study, 100 and 1 000 jobs were considered for each simulation in the Montage and CyberShake scenarios.

Input Feature	**Description**
MIPS	Millions of instructions per second
POW(Power)	IDLE power consumption
LEN(Length)	Task size
POWMAX	Dynamic power consumption
UTILIZATION	Utilization factor of the process entity

Output Feature	**Description**
SEL(Selection)	Suitability of every single host VM

Table 1. Input and Output variables for meta-scheduler

1:	IF (MIPS is low) or (pow is middle) or (len is middle) or (powmax is low) THEN (sel is very low)
2:	IF (MIPS is low) and (len is high) and (powmax is low) and (utilization is middle) THEN (sel is high)
3:	IF (pow is high) and (len is high) and (utilization is middle) THEN (sel is very high)
4:	IF (len is high) and (powmax is low) and (utilization is middle) THEN (sel is middle)
5:	IF (pow is middle) or (len is low) or (powmax is middle) or (utilization is low) THEN (sel is low)

Table 2. Example of meta-scheduler RB

The evaluation of the proposed systems focuses on multiple scenarios related to meta-scheduling in cloud computing. This work considers the application of KASIA, Pittsburgh and ANFIS for the learning process of the meta-schedulers. A crucial aspect of the selection process is obtainig a reliable RB that can effectively support the performance of the methods. Hence, the fuzzy learning system aims to generate a high-quality RB that can be used by the KASIA and Pittsburgh meta-schedulers. Table 2 provides an example of an RB used by the KASIA meta-scheduler, taking into account the features presented in Table 1. These features serve as antecedents

and consequents in the meta-scheduling systems. Additionally, the ANFIS system needs to acquire a suitable RB to ensure its optimal performance. This objective is achieved through the hybrid learning algorithm, which considers the identification of the optimal RB while minimizing the potential error. This approach is essential to prevent the system from getting trapped in local minima and ensure efficient performance.

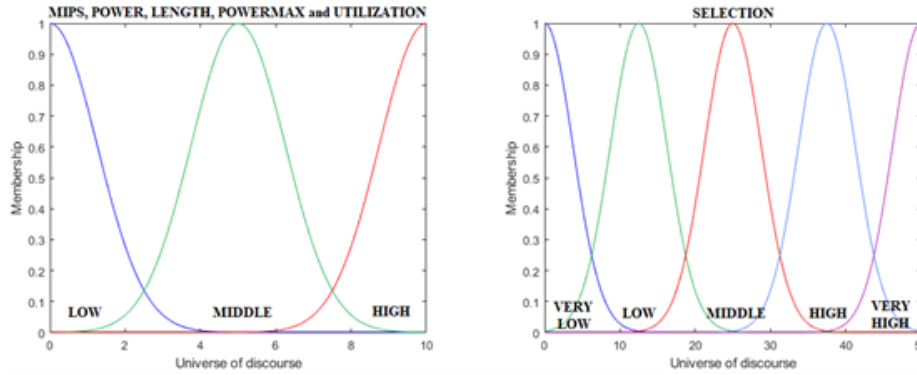


Figure 6. Gaussian fuzzy sets for classification

Figure 6 illustrates an example of the normalized input and output features, represented by fuzzy sets. The input variables are characterized by three membership functions, corresponding to low, middle, and high values within the input domain. Conversely, the output is described by five fuzzy membership functions: very low, low, middle, high, and very high. As explained in Section 3, Gaussian membership functions are used due to their smoothness and concise notation. Additionally, these curves possess the advantageous property of being non-zero at all points within the classification, covering the entire universe of discourse.

4.1 Simulation Scenarios

The simulated scenario in this study consists of a cloud system that includes 4 datacenters. The network topology is composed of 20 hosts, each equipped with a single processing element which has a capacity of 1 500 MIPS. Within these hosts, 20 VMs are deployed, each capable of delivering a performance of 1 000 MIPS. Additionally, the DVFS simulator employs inter-host scheduling as overlapping mechanism.

WorkflowSim, an extension of CloudSim, is used to handle DAX (Directed Acyclic Graph in XML) workflows or workloads characterized by high task independence complexity. Two graphical examples representing real Montage and CyberShake traces are depicted in the subsequent figures, illustrating the job types and dependencies. In the DVFS simulator's workload management, five primary entities are considered: the Planner, the Merger, the Engine, the Scheduler, and the

Datacenter. The Planner initializes the system and parses the DAX file to obtain individual tasks known as Cloudlets. The Merger, or Clustering Engine, groups these tasks into jobs, which are then processed by the Engine according to the specific workflow order. Finally, the Scheduler determines the most suitable VM within the Datacenter for executing each task.

Montage, developed by NASA, is an open-source toolkit designed to facilitate the creation of customized astronomical sky mosaics using FITS-formatted (Flexible Image Transport System) input images. The number of jobs in a Montage workflow depends on the quantity of input images required to generate the final mosaic. Figure 7 a) illustrates a Montage workflow with a limited number of jobs. As mentioned earlier, work traces obtained from the Montage project, including 100 and 1 000 jobs, are utilized to evaluate the performance of the proposed meta-schedulers in the simulator under medium load conditions.

The Southern California Earthquake Center (SCEC) has developed CyberShake, a high-performance computational platform aimed at generating seismic hazard models through extensive earthquake simulations. The number of jobs in a CyberShake workflow depends on the quantity of synthetic seismograms required for the earthquake simulation. Figure 7 b) illustrates an example of a CyberShake workflow. Traces including 100 and 1 000 jobs are utilized to evaluate the performance of the proposed system using the simulator.

4.2 Cloud Simulation Configuration

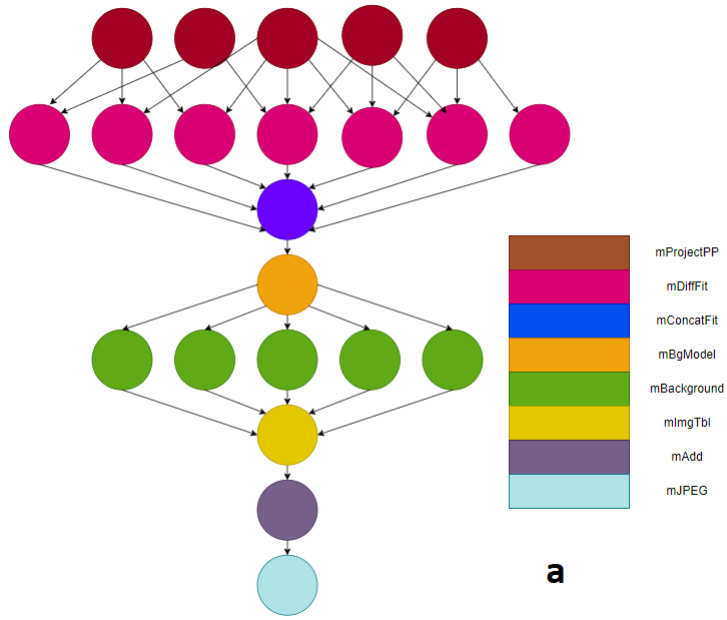
The cloud simulation configuration includes various components and characteristics that imitate a multi-region cloud environment. Below are the details of this configuration:

Number of Datacenters: The simulation involves four datacenters, representing different geographic regions. This setup enables researchers to study the behavior of a distributed cloud infrastructure and analyze factors like data locality and network latency.

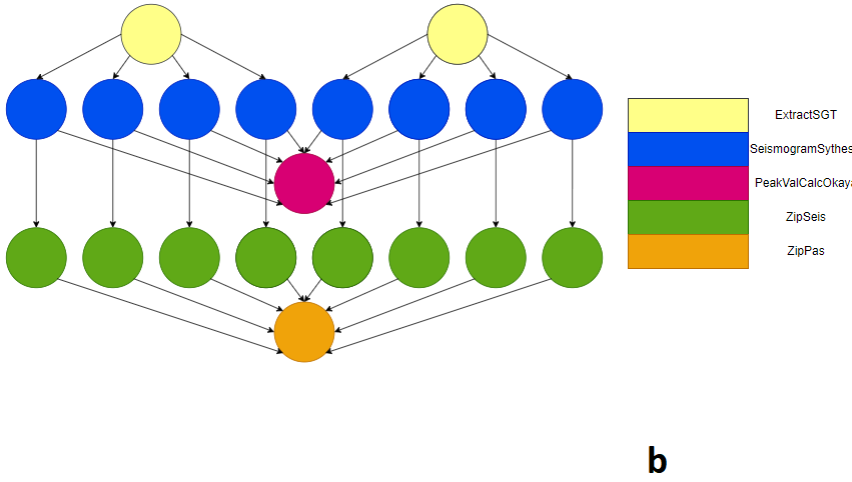
Number of Hosts: Each datacenter consists of 20 physical machines or hosts. These hosts serve as the basis for running VMs and executing cloudlets. The hosts have characteristics shown in Table 3.

Variable	Value
PEs	2
RAM	4 096 MB
BW	1 000 000 MB
MIPS	1 860/2 660
Storage	1 000 000 MB

Table 3. Feature configuration of hosts



a)



b)

Figure 7. Montage and CyberShake workflow DAX examples

Number of VMs: The cloud simulation configuration includes 20 VMs distributed across the four datacenters. Table 4 presents the features for each VM.

Variable	Value
PEs	1
RAM	613/870/1 740 MB
BW	100 000 MB
MIPS	500/1 000/2 000/2 500
Size	2 500 MB

Table 4. Feature configuration of VMs

Number of Cloudlets: The exact number of cloudlets depends on the number of tasks within scientific workflows. The simulation can include either 100 or 1 000 cloudlets to represent the workload that needs to be executed within the cloud environment. Each cloudlet has the characteristics presented in Table 5.

Variable	Value
PEs	1
Length	216 000 000

Table 5. Feature configuration of cloudlets

By utilizing this specific cloud simulation configuration, various aspects of the cloud environment can be analyzed and optimized. Resource allocation strategies can be explored, workload management techniques can be evaluated, and system performance based on real-world values and characteristics can be assessed. This configuration provides a realistic simulation environment for studying and improving cloud computing systems.

4.3 Learning Approaches Configuration

The configuration of the proposed systems plays a crucial role during the simulation stage. In the initial configuration, it is evident that KASIA and Pittsburgh requires the specification of multiple variables. Conversely, ANFIS operates effectively with a smaller number of variables. This ANFIS advantage is noteworthy, as it demonstrates that the system can optimally function with fewer variables, without compromising its performance.

To conduct KASIA simulations, certain variables must be configured, as outlined in Table 6. As indicated in Table 1, there are five antecedents and one consequent, corresponding to the five input variables and one output variable, respectively. The simulations are performed using a swarm consisting of 20 particles, and the learning process is conducted over 100 iterations. The updating mechanism incorporates a weight factor, which is initialized to a value close to unity and its formulation is dependent on the current iteration of the learning process. As shown in Table 6,

Variable	Value
Number of rules	5
Number of antecedents	5
Number of consequents	1
Number of particles	20
Maximum iteration	100
Initial weight (ω_0)	0.9
Competitive factor (c_2)	2
Cooperative factor (c_2)	2

Table 6. Parameter configuration for rule discovery in KASIA

the number of rules representing a RB is set to 5 for the three systems. This selection is based on the constraint imposed by Sugeno-type FIS, where each rule must correspond to a unique membership function. Since five output membership functions are considered, the number of rules is also set to five.

Variable	Value
Number of rules	5
Number of antecedents	5
Number of consequents	1
Number of individuals	20
Maximum iteration	100
Crossover rate	0.8
Initial mutation rate	0.1
Selection rate	0.8
Replacement rate	0.8

Table 7. Parameter configuration for rule discovery in Pittsburgh

Table 7 presents the parameter configuration for rule discovery in Pittsburgh. It includes variables such as the number of rules, antecedents, consequents, and individuals, along with values assigned to each parameter. The table specifies 5 rules with 5 antecedents and 1 consequent, using a population size of 20 individuals. The maximum iteration is set to 100, and a crossover rate of 0.8 is applied for genetic exchange. The initial mutation rate is 0.1, and selection and replacement rates are both set to 0.8. These parameter settings collectively define the rule discovery process, determining the structure and characteristics of the generated rules within a specified population and iteration limit for evolution.

During the ANFIS training process, an additional variable called "epoch" is taken into account, in addition to the number of rules and their content as specified in the KASIA and Pittsburgh configurations. The epoch variable represents the number of times the ANFIS is trained using the adaptive neural network. In the learning process, the ANFIS is trained 10 times for each RB evaluation, with the objective of minimizing the LSE. The configuration details of this technique can be

Variable	Value
Number of rules	5
Number of antecedents	5
Number of consequents	1
Number of particles	20
Maximum iteration	100
Number of epochs	10

Table 8. Parameter configuration for rule discovery in ANFIS

found in Table 8. It is noteworthy that ANFIS is configured with fewer variables compared to KASIA, which is a valuable advantage of the system.

The image depicted in Figure 8 illustrates the neural network architecture used in the ANFIS-based meta-scheduler. The neural network includes five layers. The first layer consists of three Gaussian membership functions for each input, which are connected to the second layer. In the second layer, the membership functions are combined using multiplication operations to calculate the firing strength of the five rules. The third layer normalizes the firing strength values, while the fourth layer determines the consequent parameters for the output membership functions. Finally, in the fifth layer, all incoming signals are summed to generate the output, which is a crucial characteristic of this system.

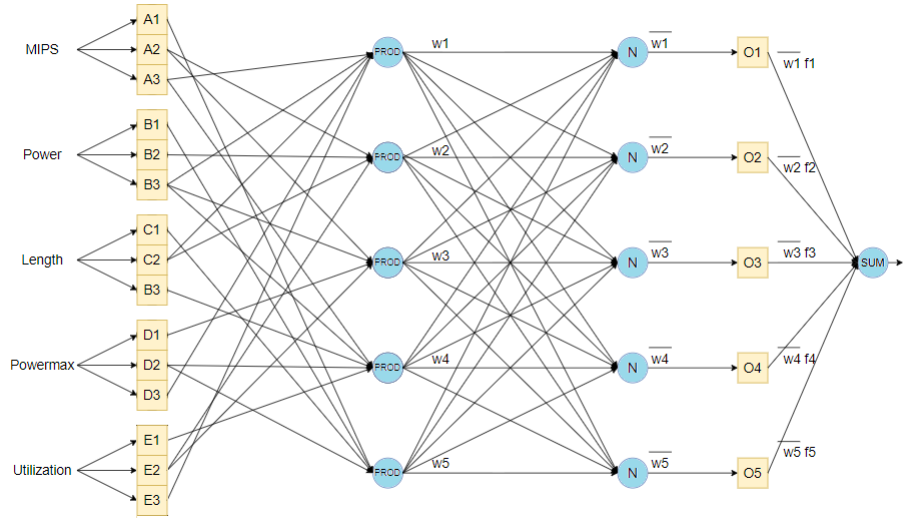
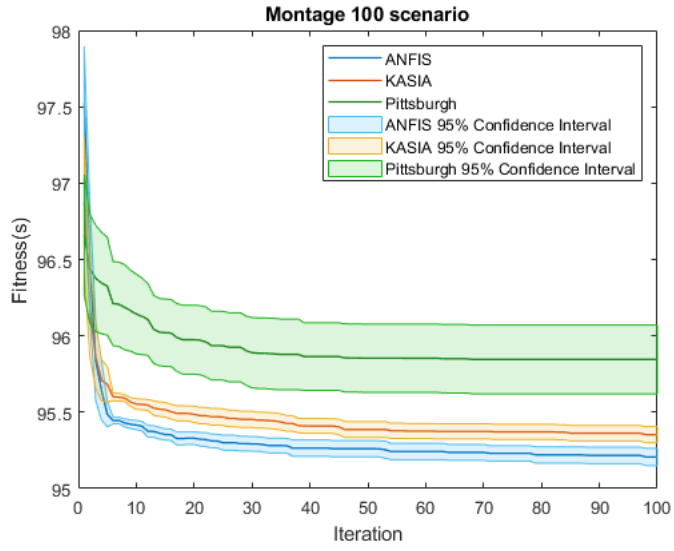
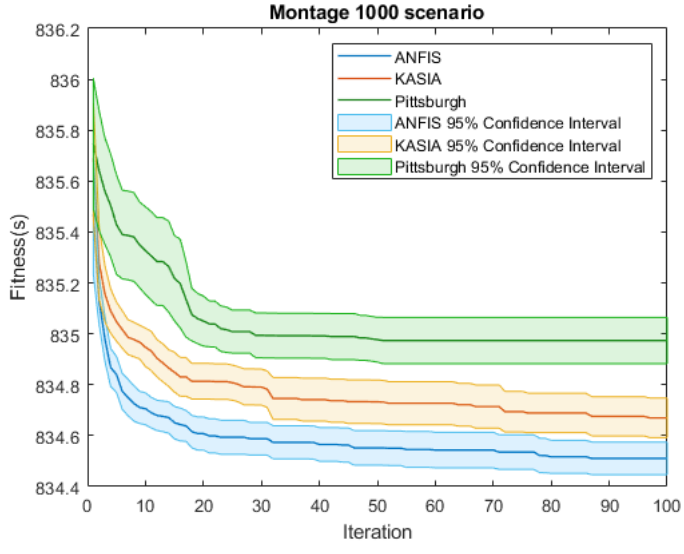


Figure 8. ANFIS neural network used in the experiments



a)



b)

Figure 9. Convergence behavior of rule-based systems with makespan as fitness using Montage structure. The 95 % confidence regions of KASIA, Pittsburgh and ANFIS in orange, green and blue, respectively, can be seen.

4.4 Simulation Results

In Figure 9, a comparison between the KASIA, Pittsburgh and ANFIS systems is presented in the context of a Montage scenario taking into account 100 and 1 000 tasks to be performed. The learning process is depicted by considering makespan as the fitness measure. The displayed values represent the average of the best solution obtained from 30 experiments at each iteration. It can be observed that both KASIA and ANFIS strategies exhibit a similar initial behavior, but ANFIS quickly converges to its final value. The ANFIS better performance is due to the use of the gradient descent/backpropagation mechanism, which allows ANFIS to find optimal or near-optimal outcomes. Conversely, the Pittsburgh approach provides a worse performance than the previous methods because this methodology is thought for finding near-optimal solutions which may lead to local minima, although the method has escape mechanisms. This can be seen in both Montage graphics, where the Pittsburgh approach reaches rapidly its final value and remain still. Regions of confidence are included to facilitate the comparison. Furthermore, Tables 9 and 10 present the mean results achieved by the swarms, indicated as the Average, along with the corresponding standard deviation (Standard Deviation) and 95 % confidence interval (95 % Confidence Interval). On average, the final result of ANFIS slightly outperforms the best result of KASIA and Pittsburgh, as indicated by the non-overlapping confidence regions in terms of makespan.

Strategy	**ANFIS**	**KASIA**	**Pittsburgh**
Average (s)	95.2067	95.3523	95.8453
Standard Deviation (s)	0.1576	0.1475	0.6133
Confidence Interval (s)	[95.1503, 95.2630]	[95.2996, 95.4051]	[95.6202, 96.0705]

Table 9. Simulations result for final makespan(s) using Montage 100

Strategy	**ANFIS**	**KASIA**	**Pittsburgh**
Average (s)	834.5107	834.6697	834.9737
Standard Deviation (s)	0.1664	0.1847	0.2544
Confidence Interval (s)	[834.4466, 834.5747]	[834.5918, 834.7475]	[834.8826, 835.0648]

Table 10. Simulations result for final makespan(s) using Montage 1 000

In Figure 10, the performance of KASIA, Pittsburgh and ANFIS in the CyberShake environment is presented. This figure illustrates the learning evolution of the fuzzy reasoning-based meta-scheduler, considering makespan as the fitness measure for both approaches. As depicted, the convergence patterns of KASIA, Pittsburgh and ANFIS strategies differ as the number of iterations increases. As in the Montage algorithm, in Cybershake the algorithms have a similar behavior when it comes to performance. The Pittsburgh approach has again that remaining still behavior, while KASIA and ANFIS can continue to be improved over iterations. Additionally,

Tables 11 and 12 provide statistical information about the meta-scheduler strategies. The final result of ANFIS demonstrates a slight superiority over the best result of KASIA and Pittsburgh on average, as indicated by the non-superposition of confidence regions in the final iteration in terms of makespan.

Strategy	**ANFIS**	**KASIA**	**Pittsburgh**
Average (s)	2 998.6015	2 999.5316	3 000.7733
Standard Deviation (s)	0.7846	1.0118	1.3604
Confidence Interval (s)	[2 998.3903, 2 998.8127]	[2 999.2099, 2 999.8533]	[3 000.2865, 3 001.2602]

Table 11. Simulations result for final makespan(s) using CyberShake 100

Strategy	**ANFIS**	**KASIA**	**Pittsburgh**
Average (s)	4 426.0363	4 428.6380	4 431.3483
Standard Deviation (s)	1.7492	2.2286	3.6501
Confidence Interval (s)	[4 425.4104, 4 426.6623]	[4 427.8405, 4 429.4355]	[4 430.0422, 4 432.6555]

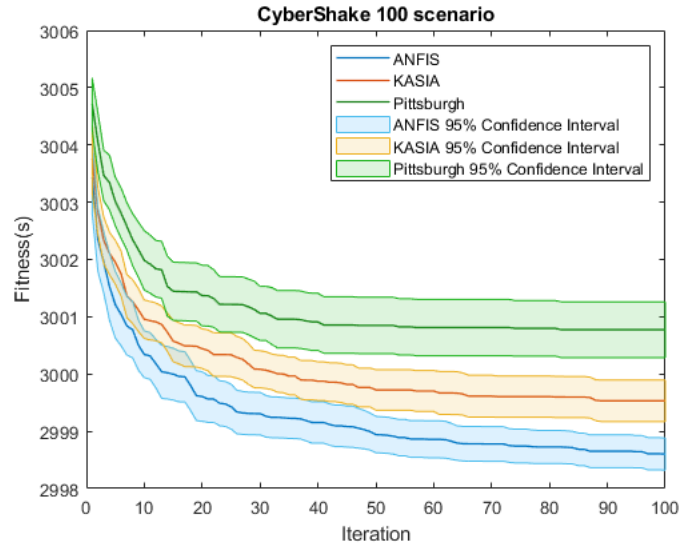
Table 12. Simulations result for final makespan(s) using CyberShake 1 000

In Figure 11, the culmination of the optimization process is graphically depicted, showcasing the final values reached by the Montage and CyberShake scenarios across different task configurations, specifically, 100 and 1 000 tasks for each scenario. The figure meticulously delineates the outcomes achieved through the application of the three distinct approaches: Pittburgh, KASIA, and ANFIS. The juxtaposition of these methodologies allows for a comprehensive analysis of their respective performances in handling the complexities associated with the specified scenarios and task quantities. This visual representation serves as a valuable resource for discerning the efficacy of the optimization techniques employed, providing insights into the comparison of the Pittburgh, KASIA, and ANFIS approaches under varying conditions.

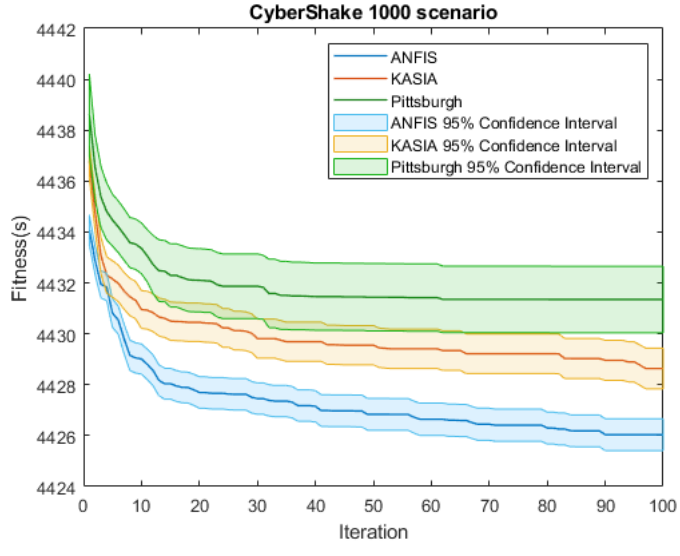
Workflow\Strategy	**ANFIS**	**KASIA**	**Pittsburgh**
Montage (s)	6.7560e+4	7.2654e+4	7.4751e+4
CyberShake (s)	7.0177e+5	7.4920e+5	7.6395e+5

Table 13. Overall execution time of simulations

When evaluating techniques in cloud computing, the computational cost plays a critical role. To assess the efficiency of ANFIS, it is important to reduce its computational cost. Therefore, the computational cost of KASIA, Pittsburgh and ANFIS systems was evaluated in terms of time. The elapsed time for the algorithm's computing delay in Montage and CyberShake scenarios was tested on a computer



a)



b)

Figure 10. Convergence behavior of rule-based systems with makespan as fitness using CyberShake structure. The 95 % confidence regions of KASIA, Pittsburgh and ANFIS can be seen in orange, green and blue, respectively.

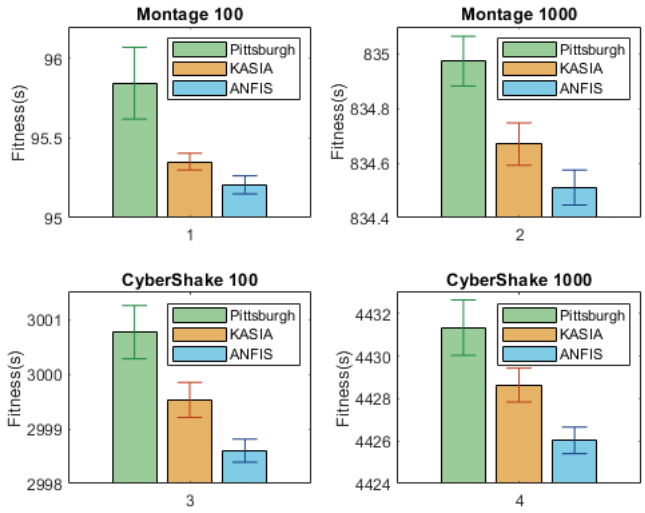


Figure 11. Ultimate values of the optimization process across the four scenarios for Pittsburgh, KASIA and ANFIS

with an Intel i7 processor and 8 cores, and the results are presented in Table 13. The ANFIS algorithm demonstrates superior computational cost performance compared to KASIA and Pittsburgh in the simulated scenarios, with improvements of 7.01 % and 6.33 % for KASIA and 10.74 % and 8.86 % for Pittsburgh in the Montage and CyberShake scenarios, respectively. Furthermore, the number of RB evaluations serves as a reliable measure for evaluating the computational cost of the proposed systems. ANFIS requires 120 000 evaluations, while KASIA and Pittsburgh require 240 000 evaluations and 192 000, respectively. The computational cost formulation in terms of the number of evaluations for the three systems is as follows:

$$N_{ANFIS} = 10 \text{ epochs} \cdot 100 \text{ iterations} \cdot 120 \text{ simulations} = 120\,000 \text{ RB evaluations},$$

$$N_{KASIA} = 20 \text{ particles} \cdot 100 \text{ iterations} \cdot 120 \text{ simulations} = 240\,000 \text{ RB evaluations},$$

$$\begin{aligned} N_{Pittsburgh} &= 0.8\,(80\,\%\text{ Selection}) \cdot 20 \text{ individuals} \cdot 100 \text{ iterations} \cdot 120 \text{ simulations} \\ &= 192\,000 \text{ RB evaluations}. \end{aligned}$$

The ANFIS system's lower computational cost, coupled with its superior performance regarding makespan, suggests that it is the preferred system over KASIA and Pittsburgh. After presenting the results, it is noteworthy to showcase the best RB achieved by the ANFIS-based meta-scheduler, given its outperformance compared to the KASIA-based and Pittsburgh-based systems. The rules for this RB are presented in Table 14.

1:	IF (MIPS is middle) or (pow is high) or (len is middle) THEN (sel is very high)
2:	IF (MIPS is high) and (pow is middle) and (len is middle) and (utilization is middle) THEN (sel is very low)
3:	IF (MIPS is middle) or (pow is high) or (powmax is low) or (utilization is middle) THEN (sel is middle)
4:	IF (MIPS is high) and (pow is high) and (len is high) and (powmax is middle) and (utilization is low) THEN (sel is low)
5:	IF (MIPS is low) or (pow is low) or (len is low) or (powmax is middle) THEN (sel is high)

Table 14. Good RB for the CyberShake scenario

5 CONCLUSIONS AND FUTURE WORK

While there is a significant body of research on scheduling algorithms in cloud computing, only a limited number of studies address the inherent uncertainty and dynamism of resources. To tackle this, three machine learning techniques based on FRBS have been introduced. The first technique leverages PSO to achieve higher quality RBs in a shorter time and with a simple setup. The second technique employs a genetic algorithm that enables the finding of near-optimal solutions taking advantage of the genetics features. And the third technique exploits the properties of neural networks to acquire knowledge more efficiently and achieve better convergence speed in terms of makespan.

In this work, the proposed neuro-fuzzy scheduler is applied to job scheduling in cloud computing and compared with the KASIA and Pittsburgh fuzzy and genetic systems, respectively. Makespan is used as the training metric for the expert system. The results demonstrate that the neuro-fuzzy system slightly outperforms the fuzzy systems in two different environments, Montage and CyberShake. Additionally, the proposed system exhibits a computational cost advantage, with reductions of 7.01 % and 6.33 % for KASIA and 10.74 % and 8.86 % for Pittsburgh in Montage and CyberShake, respectively, in terms of elapsed time. Furthermore, the proposed system outperforms the KASIA system by 50 % and Pittsburgh by 37.5 % in terms of the number of RB evaluations.

These experiments motivate further comparisons and investigations in the field of neural networks applied to cloud computing, particularly in the context of task scheduling in terms of makespan and computational effort. Additionally, future research may explore energy reduction and renewable energy usage through VM migration. Two works are presented for future investigations: investigate game-theoretic approaches to meta-scheduling in cloud computing, developing models capturing interactions among stakeholders and analyzing equilibrium outcomes, and developing a multi-objective swarm fuzzy algorithm for cloud task scheduling, optimizing makespan, renewable energy usage, and interpretability through a Pareto-based optimization framework.

Acknowledgements

This work has been supported by the research project No. P18-RT-4046, funded by the Andalusian Government, and by the NextGenerationEU recovery plan, funded by the European Union.

REFERENCES

[1] Seddiki, D.—García Galán, S.—Muñoz Expósito, J. E.—Valverde Ibáñez, M.—Marciniak, T.—Pérez de Prado, R. J.: Sustainable Expert Virtual Machine Migration in Dynamic Clouds. Computers and Electrical Engineering, Vol. 102, 2022, Art. No. 108257, doi: 10.1016/j.compeleceng.2022.108257.

[2] Cordón, O.—Herrera, F.—Hoffmann, F.—Magdalena, L.: Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. World Scientific, 2001, doi: 10.1142/4177.

[3] Booker, L. B.—Goldberg, D. E.—Holland, J. H.: Classifier Systems and Genetic Algorithms. Artificial Intelligence, Vol. 40, 1989, No. 1-3, pp. 235–282, doi: 10.1016/0004-3702(89)90050-7.

[4] Smith, S. F.: A Learning System Based on a Genetic Adaptive Algorithm. Ph.D. Thesis. University of Pittsburgh, 1980.

[5] Adil, M.—Nabi, S.—Raza, S.: PSO-CALBA: Particle Swarm Optimization Based Content-Aware Load Balancing Algorithm in Cloud Computing Environment. Computing and Informatics, Vol. 41, 2022, No. 5, pp. 1157–1185, doi: 10.31577/cai_2022_5_1157.

[6] Benabbes, S.—Hemam, S. M.: An Approach Based on Genetic and Grasshopper Optimization Algorithms for Dynamic Load Balancing in CloudIoT. Computing and Informatics, Vol. 42, 2023, No. 2, pp. 364–391, doi: 10.31577/cai_2023_2_364.

[7] Pérez de Prado, R. J.—García Galán, S.—Muñoz Expósito, J. E.—Yuste Delgado, A. J.: Knowledge Acquisition in Fuzzy Rule-Based Systems with Particle-Swarm Optimization. IEEE Transactions on Fuzzy Systems, Vol. 18, 2010, No. 6, pp. 1083–1097, doi: 10.1109/TFUZZ.2010.2062525.

[8] García Galán, S.—Pérez de Prado, R. J.—Muñoz Expósito, J. E.: Swarm Fuzzy Systems: Knowledge Acquisition in Fuzzy Systems and Its Applications in Grid Computing. IEEE Transactions on Knowledge and Data Engineering, Vol. 26, 2014, No. 7, pp. 1791–1804, doi: 10.1109/TKDE.2013.118.

[9] Sarhadi, A.—Torkestani, J. A.: Cost-Effective Scheduling and Load Balancing Algorithms in Cloud Computing Using Learning Automata. Computing and Informatics, Vol. 42, 2023, No. 1, pp. 37–74, doi: 10.31577/cai_2023_1_37.

[10] Jang, J. R.: Fuzzy Nodeling Using Generalized Neural Networks and Kalman Filter Algorithm. Proceedings of the 9th National Conference on Artificial Intelligence (AAAI'91), Vol. 2, 1991, pp. 762–767.

[11] Buriboev, A.—Muminov, A.: Computer State Evaluation Using Adaptive Neuro-Fuzzy Inference Systems. Sensors, Vol. 22, 2022, No. 23, Art. No. 9502, doi: 10.3390/s22239502.

[12] Cotes Ruiz, I. T.—Pérez de Prado, R. J.—García Galán, S.—Muñoz-Expósito, J. E.—Ruiz Reyes, N.: Dynamic Voltage Frequency Scaling Simulator for Real Workflows Energy-Aware Management in Green Cloud Computing. PLoS ONE, Vol. 12, 2017, No. 1, Art. No. e0169803, doi: 10.1371/journal.pone.0169803.

[13] Calheiros, R. N.—Ranjan, R.—Beloglazov, A.—De Rose, C. A. F.—Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Practice and Experience, Vol. 41, 2011, No. 1, pp. 23–50, doi: 10.1002/spe.995.

[14] Chen, W.—Deelman, E.: WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments. 2012 IEEE 8th International Conference of E-Science, 2012, pp. 1–8, doi: 10.1109/eScience.2012.6404430.

[15] Deelman, E.—Vahi, K.—Rynge, M.—Mayani, R.—Ferreira da Silva, R.—Papadimitrou, G.—Livny, M.: The Evolution of the Pegasus Workflow Management Software. Computing in Science & Engineering, Vol. 21, 2019, No. 4, pp. 22–36, doi: 10.1109/MCSE.2019.2919690.

[16] Liang, B.—Dong, X.—Wang, Y.—Zhang, X.: A High-Applicability Heterogeneous Cloud Data Centers Resource Management Algorithm Based on Trusted Virtual Machine Migration. Expert Systems with Applications, Vol. 197, 2022, Art. No. 116762, doi: 10.1016/j.eswa.2022.116762.

[17] Rozas, H.—Jaramillo, F.—Perez, A.—Jimenez, D.—Orchard, M. E.—Medjaher, K.: A Method for the Reduction of the Computational Cost Associated with the Implementation of Particle-Filter-Based Failure Prognostic Algorithms. Mechanical Systems and Signal Processing, Vol. 135, 2020, Art. No. 106421, doi: 10.1016/j.ymssp.2019.106421.

[18] Moghadam, M. H.—Babamir, S. M.: Makespan Reduction for Dynamic Workloads in Cluster-Based Data Grids Using Reinforcement-Learning Based Scheduling. Journal of Computational Science, Vol. 24, 2018, pp. 402–412, doi: 10.1016/j.jocs.2017.09.016.

[19] Güğül, G. N.—Gökçül, F.—Eicker, U.: Sustainability Analysis of Zero Energy Consumption Data Centers with Free Cooling, Waste Heat Reuse and Renewable Energy Systems: A Feasibility Study. Energy, Vol. 262, Part B, 2023, Art. No. 125495, doi: 10.1016/j.energy.2022.125495.

[20] Gill, S. S.—Xu, M.—Ottaviani, C.—Patros, P.—Bahsoon, R. et al.: AI for Next Generation Computing: Emerging Trends and Future Directions. Internet of Things, Vol. 19, 2022, Art. No. 100514, doi: 10.1016/j.iot.2022.100514.

[21] Murad, S. A.—Muzahid, A. J. M.—Azmi, Z. R. M.—Hoque, M. I.—Kowsher, M.: A Review on Job Scheduling Technique in Cloud Computing and Priority Rule-Based Intelligent Framework. Journal of King Saud University – Computer, and Information Sciences, Vol. 34, 2022, No. 6, Part A, pp. 2309–2331, doi: 10.1016/j.jksuci.2022.03.027.

[22] Wu, H.—Chen, X.—Song, X.—Zhang, C.—Guo, H.: Scheduling Large-Scale Scientific Workflow on Virtual Machines with Different Numbers of vCPUs. The Journal of Supercomputing, Vol. 77, 2021, No. 1, pp. 679–710, doi: 10.1007/s11227-020-03273-3.

[23] Rukmini, S.—Shridevi, S.: An Optimal Solution to Reduce Virtual Machine Migration SLA Using Host Power. Measurement: Sensors, Vol. 25, 2023, Art. No. 100628, doi: 10.1016/j.measen.2022.100628.

[24] Aladwani, T.: Types of Task Scheduling Algorithms in Cloud Computing Environment. In: Righi, R. (Ed.): Scheduling Problems – New Applications and Trends. IntechOpen, 2019, pp. 131–142, doi: 10.5772/intechopen.86873.

[25] Skabar, A.—Abdalgader, K.: Clustering Sentence-Level Text Using a Novel Fuzzy Relational Clustering Algorithm. IEEE Transactions on Knowledge and Data Engineering, Vol. 25, 2013, No. 1, pp. 62–75, doi: 10.1109/TKDE.2011.205.

[26] Zadeh, L. A.: Fuzzy Sets. Information and Control, Vol. 8, 1965, No. 3, pp. 338–353, doi: 10.1016/S0019-9958(65)90241-X.

[27] Meniz, B.—Özkan, E. M.: Vaccine Selection for COVID-19 by AHP and Novel VIKOR Hybrid Approach with Interval Type-2 Fuzzy Sets. Engineering Applications of Artificial Intelligence, Vol. 119, 2023, Art. No. 105812, doi: 10.1016/j.engappai.2022.105812.

[28] Jithendra, T.—Sharief Basha, S.: A Hybridized Machine Learning Approach for Predicting COVID-19 Using Adaptive Neuro-Fuzzy Inference System and Reptile Search Algorithm. Diagnostics, Vol. 13, 2023, No. 9, Art. No. 1641, doi: 10.3390/diagnostics13091641.

[29] Mamdani, E. H.: Application of Fuzzy Algorithms for Control of the Simple Dynamic Plant. Proceedings of the Institution of Electrical Engineers, Vol. 121, 1974, No. 12, pp. 1585–1588, doi: 10.1049/piee.1974.0328.

[30] Takagi, T.—Sugeno, M.: Fuzzy Identification of Systems and Their Applications to Modeling and Control. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-15, 1985, No. 1, pp. 116–132, doi: 10.1109/TSMC.1985.6313399.

[31] Kennedy, J.—Eberhart, R.: Particle Swarm Optimization. Proceedings of ICNN'95 – International Conference of Neural Networks, Vol. 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[32] Roui, M. B.—Zomorodi, M.—Sarvelayati, M.—Abdar, M.—Noori, H.—Pławiak, P.—Tadeusiewicz, R.—Zhou, X.—Khosravi, A.—Nahavandi, S.—Acharya, U. R.: A Novel Approach Based on Genetic Algorithm to Speed Up the Discovery of Classification Rules on GPUs. Knowledge-Based Systems, Vol. 231, 2021, Art. No. 107419, doi: 10.1016/j.knosys.2021.107419.

[33] Kalia, H.—Dehuri, S.—Ghosh, A.: Fitness Inheritance in Multi-Objective Genetic Algorithms: A Case Study on Fuzzy Classification Rule Mining. International Journal of Advanced Intelligence Paradigms, Vol. 23, 2022, No. 1-2, pp. 89–112, doi: 10.1504/IJAIP.2022.125235.

[34] Konishi, T.—Masuyama, N.—Nojima, Y.: Effects of Accuracy-Based Single-Objective Optimization in Multiobjective Fuzzy Genetics-Based Machine Learning. 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS & ISIS), IEEE, 2022, pp. 1–6, doi: 10.1109/SCISISIS55246.2022.10002139.

[35] Dinh, T. P.: Adaptive Evolutionary Multitasking to Solve Inter-Domain Path Computation Under Node-Defined Domain Uniqueness Constraint: New Solution En-

coding Scheme. Computing and Informatics, Vol. 42, 2023, No. 1, pp. 98–125, doi: 10.31577/cai_2023_1_98.

[36] Yu, X.—Liang, J.: Genetic Fuzzy Tree Based Node Moving Strategy of Target Tracking in Multimodal Wireless Sensor Network. IEEE Access, Vol. 6, 2018, pp. 25764–25772, doi: 10.1109/ACCESS.2018.2835162.

[37] Tan, Q.—Sun, L.: A Novel Association Rules Mining Based on Improved Fusion Particle Swarm Optimization Algorithm. In: Atiquzzaman, M., Yen, N., Xu, Z. (Eds.): Proceedings of the 4th International Conference on Big Data Analytics for Cyber-Physical System in Smart City – Volume 1 (BDCPS 2022). Springer, Singapore, Lecture Notes on Data Engineering and Communications Technologies, Vol. 167, 2023, pp. 103–111, doi: 10.1007/978-981-99-0880-6_12.

[38] Kazemi, M. A.—Pa, M.—Uddin, M. N.—Rezakazemi, M.: Adaptive Neuro-Fuzzy Inference System-Based Data Interpolation for Particle Image Velocimetry in Fluid Flow Applications. Engineering Applications of Artificial Intelligence, Vol. 119, 2023, Art. No. 105723, doi: 10.1016/j.engappai.2022.105723.

[39] Nagalakhsmi, K.—Suriya, S.: Performance Analysis of Breast Cancer Detection Method Using ANFIS Classification Approach. Computer Systems, Science and Engineering, Vol. 44, 2023, No. 1, pp. 501–517, doi: 10.32604/csse.2023.022687.

[40] Keikhisrokani, P.—Naidu A/P Anathan, A. B.—Iryanti Fadilah, S.—Manickam, S.—Li, Z.: Heartbeat Sound Classification Using Hybrid Adaptive Neuro-Fuzzy Inferences System (ANFIS) and Artificial Bee Colony. Digital Health, Vol. 9, 2023, doi: 10.1177/20552076221150741.

[41] Ambhika, C.—Murukesh, C.: Optimized ANFIS Model for Stable Clustering in Cognitive Radio Network. Intelligent Automation and Soft Computing, Vol. 35, 2023, No. 1, pp. 827–838, doi: 10.32604/iasc.2023.026832.

[42] Uluocak, İ.—Yavuz, H.: Artifical Intelligence Based PID Controller for an Eddy Current Dynamometer. Intelligent Automation and Soft Computing, Vol. 33, 2022, No. 2, pp. 1229–1243, doi: 10.32604/iasc.2022.023835.

[43] Chang, P. Y.—Chou, F. I.—Yang, P. Y.—Chen, S. H.: Hybrid Multi-Object Optimization Method for Tapping Center Machines. Intelligent Automation and Soft Computing, Vol. 36, 2023, No. 1, pp. 23–38, doi: 10.32604/iasc.2023.031609.

[44] Atlam, H. F.—Wills, G. B.: ANFIS for Risk Estimation in Risk-Based Access Control Model for Smart Homes. Multimedia Tools and Applications, Vol. 82, 2023, No. 12, pp. 18269–18298, doi: 10.1007/s11042-022-14010-8.

[45] Renaud, J.—Karam, R.—Salomon, M.—Couturier, R.: Deep Learning and Gradient Boosting for Urban Environmental Noise Monitoring in Smart Cities. Expert Systems with Applications, Vol. 218, 2023, Art. No. 119568, doi: 10.1016/j.eswa.2023.119568.

[46] Thandra, J.—Sharief Basha, S.: Artificial Intelligence (AI) Model: Adaptive Neuro-Fuzzy Inference System (ANFIS) for Diagnosis of COVID-19 Influenza. Computing and Informatics, Vol. 41, 2022, No. 4, pp. 1114–1135, doi: 10.31577/cai_2022_4_1114.

[47] Wu, X.—Wang, H.—Wei, D.—Shi, M.: ANFIS with Natural Language Processing and Gray Relational Analysis Based Cloud Computing Framework for Real Time

Energy Efficient Resource Allocation. Computer Communications, Vol. 150, 2020, pp. 122–130, doi: 10.1016/j.comcom.2019.11.015.

[48] Amekraz, Z.—Hadi, M. Y.: CANFIS: A Chaos Adaptive Neural Fuzzy Inference System for Workload Prediction in the Cloud. IEEE Access, Vol. 10, 2022, pp. 49808–49828, doi: 10.1109/ACCESS.2022.3174061.

[49] Kumar, D.—Mandal, N.—Kumar, Y.: Fog-Based Framework for Diabetes Prediction Using Hybrid ANFIS Model in Cloud Environment. Personal and Ubiquitous Computing, Vol. 27, 2023, No. 3, pp. 909–916, doi: 10.1007/s00779-022-01678-w.

[50] Jithendra, T.—Basha, S. S.: Analyzing Groundwater Level with Hybrid ANN and ANFIS Using Metaheuristic Optimization. Earth Science Informatics, Vol. 16, 2023, No. 4, pp. 3323–3353, doi: 10.1007/s12145-023-01097-2.

[51] Jithendra, T.—Sharief Basha, S.—Das, R.—Gajjela, R.: Modeling and Optimization of WEDM of Monel 400 Alloy Using ANFIS and Snake Optimizer: A Comparative Study. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, Vol. 238, 2023, No. 5, pp. 1573–1589, doi: 10.1177/09544062231187207.

[52] Montage: An Astronomical Image Mosaic Engine. `http://montage.ipac.caltech.edu`.

[53] Statewide California Earthquake Center. `https://www.scec.org`.

Francisco Javier Maldonado Carrascosa received his M.Sc. degree in physics and mathematics from the Granada University in 2021. Currently, he is a junior researcher and Ph.D. student at the Telecommunication Engineering Department of the Jaén University. His current research interests include engineering applications of artificial intelligence (neural networks, scheduling in edge/cloud/fog computing).

Antonio Jiménez Sánchez received his M.Sc. degree in telecommunication engineering from the Jaén University in 2023. Currently, he is a junior researcher at the Telecommunication Engineering Department of the Jaén University. His current research interests include engineering applications of artificial intelligence (multi-objective optimization, scheduling in edge/cloud/fog computing).

Doraid Seddiki received his M.Sc. degree in telecommunication engineering from Cartagena University. Currently he is a Ph.D. student at the University of Jaén. His current research interests include artificial intelligence, cloud computing and scheduling. In addition, he also has a wide experience working in the private sector for more than 20 years.

Sebastián García Galán received his M.Sc. and his Ph.D. degrees in telecommunication engineering from the Málaga University and the Technical University of Madrid (UPM), in 1995 and 2004, respectively. Currently, he is Full Professor in telematics engineering at the Telecommunication Engineering Department of the Jaén University. His current research interests include engineering applications of artificial intelligence (cardiopulmonary illnesses, scheduling in edge/cloud/fog computing).

Manuel Valverde Ibáñez received his M.Sc. degree in industrial management engineering from the University of Jaén, Spain, in 2002, and his Ph.D. degree in industrial engineering from the UNED in 2006. He is Associate Professor in the Electrical Engineering Department of the University of Jaén. His research interests include renewable energy power systems, power quality, modeling and control of power converters and smart grids.

Tomasz Marciniak is Associate Professor in the Telecommunication, Computer Science and Electrical Engineering Department of the Bydgoszcz University of Science and Technology. His research interests include networking, artificial intelligence, IoT technologies, and bioinformatics.