AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 56(1) (2024) 33-54
DOI: 10.22060/miscj.2024.23205.5360

Efficient scheduling algorithm for optimizing system load in fog computing environment: A fuzzy reinforcement learning mechanism

Reyhane Ghafari, Najme Mansouri*

Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran

ABSTRACT: New technologies have emerged over the last few years, such as IoT and fog computing. IoT devices and the enormous amounts of data generated every minute have led to the vast growth of the Internet of Things (IoT). In order to meet the term "Data Never Sleeps", some IoT applications require real-time services and low bit latency. To provide quick processing, storage, and services, Cisco proposed fog computing as an extension of cloud computing. The traditional methods are not capable of addressing the complex scheduling scenarios of fog computing. In this paper, we introduce a novel Fuzzy Reinforcement Learning Scheduling algorithm (FRLS) that enhances schedule accuracy in dynamic computing environments. To optimize task scheduling, the FRLS algorithm integrates fuzzy logic with reinforcement learning. To prioritize critical tasks, fuzzy logic handles uncertainty and prioritizes tasks according to deadlines, sizes, and file sizes. Then, reinforcement learning schedules the prioritized tasks, continually adjusting to dynamic conditions to ensure the best resource allocation. In addition to improving overall system performance, this combination provides a robust framework that can address the complexity and variability of fog computing environments. FRLS is designed to minimize response time while adhering to resource and deadline constraints in fog-based applications. A comparison of FRLS with existing algorithms shows that it significantly improves load balancing, deadline satisfaction, response time, and waiting time. Combining reinforcement learning and fuzzy logic leads to an efficient scheduling solution. In addition, FRLS outperforms non-prioritized algorithms.

Review History:

Received: May, 20, 2024
Revised: Aug. 12, 2024
Accepted: Jul. 23, 2024
Available Online: Sep. 05, 2024

Keywords:

Fog Computing
Reinforcement Learning (RL)
Fuzzy Logic
Scheduling
Internet of Things (IoT).

1- Introduction

The Internet of Things (IoT) has become embedded in our society, transforming everyday items into communication devices, which offers new challenges and opportunities. The current cloud infrastructure cannot support many IoT applications for three main reasons. The cost of data transmission, bandwidth limitations, and processing overhead make it impractical to transfer data from end devices to cloud servers. Moreover, real-time analysis applications, like video apps, gaming apps, etc., can suffer from significant end-to-end delays. Privacy and security concerns make it advisable or even forbidden for specific data to cross the Internet. There is a promising paradigm that can reduce communication overhead, reduce data transfer delays, and avoid network bottlenecks. It combines cloud computing with edge devices for decentralized processing [1, 2]. In 2012, fog computing was introduced as a concept [3]. Users can access data management, processing, and storage capabilities by bridging the gap between the cloud and their computers. It not only distributes configuration, control, and data management across the network but also the devices, so that the cloud handles everything [4]. Figure 1 illustrates the fog computing architecture for task scheduling. These devices have storage, computation, and networking capabilities, so they can access fog/cloud resources. Additionally, static resources can be allocated for new requests, or static and dynamic resources can be combined.

Fog computing provides cost-effective and high-performance task scheduling. Fog computing schedules tasks by allocating resources. With the proper selection of resources, tasks are completed more quickly, quality of service (QoS) is improved, and efficiency is improved. The issue of resource management is addressed with various techniques. ML-based techniques have gained popularity recently [6, 7].

The Reinforcement Learning (RL) mechanism strives to optimize rewards by interacting with the environment. The agent learns according to the predefined goals using the experience gained from the environment. A state s is currently being experienced by the agent. The action a is performed continuously. As a result of the agent's action, the environment enters a new state s and the agent receives a reward. To maximize the expected total reward, the agent uses RL. Trial-and-error learning in a dynamic environment is the basis of RL. As a result of environmental feedback, the agent modifies its adopted strategies to maximize rewards [8,

*Corresponding author's email: najme.mansouri@gmail.com

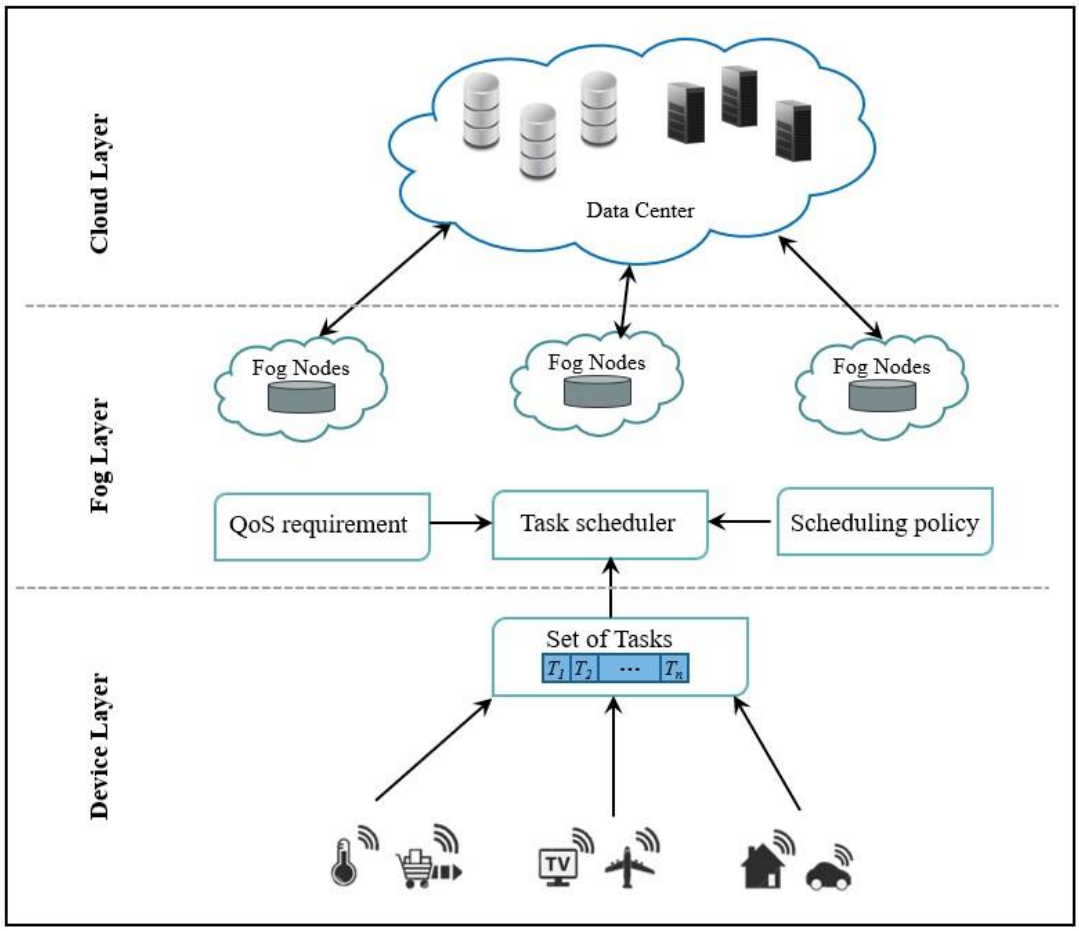


Fig. 1. Task scheduling with fog computing [5].

9]. Figure 2 shows the interaction between an agent and its environment in RL architecture.

Nowadays, decision-making is more critical than ever, despite updated technologies. Several technologies fail to consider human capacity when making decisions [10]. Effective decision-making should enable people with valuable insights to reach a very acceptable decision. Making decisions using fuzzy logic is promising. Fuzzy information extends the classical notion of set in how humans make decisions. In order to facilitate reliable decision-making, fuzzy theory transforms the data into linguistic language. The use of various linguistic languages has been widespread, including low, medium, high, small, medium, large, and many others. Languages are selected based on the type of data and compatibility [11]researchers developed fuzzy-based scheduling algorithms. Fuzzy logic is ideal for decision-making processes since it has a low computational complexity and processing power requirement. Motivated by the extensive research efforts in the distributed computing and fuzzy applications, we present a review of high-quality articles related to fuzzy-based scheduling algorithms in grid, cloud, and fog published between 2005 and June 2023. This paper discusses and compares fuzzy-based scheduling schemes based on merits and demerits, evaluation techniques, simulation environments, and important parameters. We begin by introducing distributed environments, and scheduling process followed by their surveys. This study has summarized several domains where fuzzy logic is used in distributed systems. More specifically, the basic concepts of fuzzy inference system and motivations of fuzzy theory in scheduler are addressed smoothly. A fuzzy-based scheduling algorithm employs fuzzy logic in different ways (e.g., calculating fitness functions, assigning tasks to fog/cloud nodes, and clustering tasks or resources.

Fuzzy logic constructs different degrees of membership, known as membership functions, to aid decision-making.

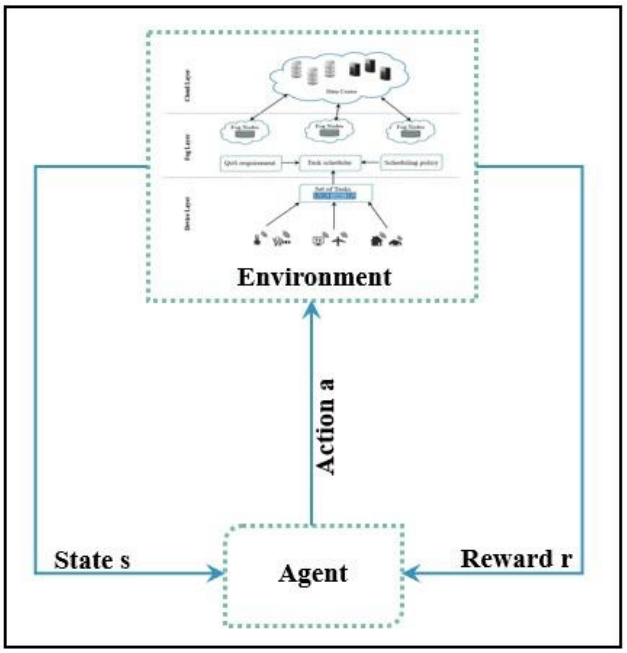


Fig. 2. Agent's interaction with the environment in RL [8].

This method of computing is also called fuzzy logic since it relies on degrees of truth instead of true or false (1 or 0), as used by computers. Linguistic languages use the interval [0,1]. In decision-making research, multi-criteria decision-making (MCDM) is a key focus, which examines the feasibility of alternative options based on the available resources and reflects the subjectivity of the decision-maker when choosing, prioritizing, and arranging a variety of actions. Since linguistic variables and fuzzy variables could be included in the objectives and limitations, fuzzy theory is combined with MCDM to investigate subjective ambiguity [12].

Few studies examined task prioritization along with task scheduling in the fog layer. There is also the issue of limited heterogeneous fog resources, impacting the task's response time and the load balancing and waiting time at the fog nodes. Therefore, it is vital to schedule tasks for available fog resources while considering response time, load, and waiting times. Load balancing is a focus for many researchers [13]. In fog computing, fuzzy reinforcement learning (FRL) has emerged as a powerful approach for making decisions in uncertain environments. FRL methods for task scheduling often fail to consider priority, load, response time, and waiting time simultaneously. In this paper, we introduce a novel reinforcement learning framework for task scheduling in fog environments that incorporates adaptive fuzzy inference mechanisms. It improves scheduling efficiency and robustness against the uncertainties inherent in fog computing.

In order to overcome these challenges, we designed a task scheduling mechanism that reduces response time and waiting time while meeting deadlines for each task. Task properties are considered in the model. The different features of the tasks, such as deadlines, size, and file size, make it difficult to prioritize them. Therefore, we recommend a fuzzy logic technique for prioritizing tasks before assigning them to fog nodes. As IoT requests rise, conventional optimization methods become increasingly unsuitable for allocating tasks across the fog of resources. In order to distribute tasks among fog nodes in the fog layer, we propose a Reinforcement Learning (RL) mechanism. Extensive simulation studies are conducted to prove the superiority of the proposed approach.

This paper contributes the following main contributions:

- IoT tasks are scheduled in fog nodes in order to balance the load while meeting the task deadlines. Waiting time minimization is also part of our model.
- Utilize fuzzy logic to prioritize tasks according to deadline, size, and file size.
- Q-Learning model is an integrated machine learning technique based on reinforcement learning.
- Extensive experiments are conducted to analyze and compare the proposed algorithm with existing algorithms.

Section 2 reviews related work in the remainder of the paper. The models of the system are presented in Section 3. Section 4 presents the proposed scheduling algorithm. Section 5 presents and analyzes the simulation results. Section 6 concludes the paper by discussing future research directions.

2- Related Work

Fog computing is a relatively new research area. Despite a few proposals, task scheduling in fog networks is gaining momentum. We have described some task scheduling and resource management works in this section.

Ghanavati et al. [14] developed a task-scheduling algorithm for fog computing platforms. The proposed approach includes two parts: 1) A bio-inspired optimization approach, Ant Mating Optimization (AMO), and 2) a distributed optimization method that optimizes task distribution. The goal is to find a compromise between system lifetime and end-user energy consumption. The proposed approach is more energy efficient and faster.

According to Assamarai et al. [15], a new task-scheduling algorithm focuses on deadline satisfaction and makespan. It aims to balance job completion deadlines with the overall efficiency of the system. Whenever there is a high or medium bandwidth to the cloud, Ant Colony Optimization (ACO) is used. These goals can be achieved through bandwidth-deadline. In terms of both makespan and deadline satisfaction, the proposed algorithm outperformed existing algorithms.

Ahmed et al. [16]in which tasks are going to be mapped on the best possible resources regarding some conflicting objectives. To deal with these issues, we introduce an opposition-based hybrid discrete optimization algorithm, called DMFO-DE. For this purpose, first, a discrete and Opposition-Based Learning (OBL proposed an opposition-based hybrid discrete optimization algorithm. In the MFO algorithm, discrete and Opposition-Based Learning (OBL) versions are first implemented, and then they are coupled with the Differential Evolution (DE) algorithm to enhance convergence speed. Fog computing schedules scientific workflows using Dynamic Voltage and Frequency Scaling (DVFS). HEFT determines the order of tasks in a scientific workflow. The scheduling process minimized the number of virtual machines (VMs), the makespan, and communication between tasks to reduce energy consumption.

Ghafari and Mansouri [17] proposed an enhanced African vulture optimization algorithm for cloud-based fog computing. As a result, villages can learn from each other rather than from all their members. It minimizes makespan, cost, and energy consumption. The Best Worst Method (BWM) handles task delays. Fog is used for tasks that require less latency, and cloud is used for tasks that require more latency. The proposed algorithm outperformed other competitors in terms of makespan, cost, and energy consumption.

Guevara et al. [18] proposed three multi-objective task scheduling algorithms for the cloud-fog continuum: FLAMSKE-INT, FLAMSKE-RR, and FLAMSKE-RL. These algorithms aim to minimize both the makespan and processing costs of workflows while maintaining QoS. The FLAMSKE-INT algorithm employs integer linear programming, while FLAMSKE-RR offers an approximate solution. It demonstrates the novelty of multi-objective scheduling for addressing diverse QoS requirements. The FLAMSKE-RL algorithm is more efficient than other algorithms when dealing with moderate to high network loads while maintaining short execution times.

Saif et al. [19] introduced Multi-Objectives Grey Wolf Optimizer (MGWO) to reduce QoS objectives delay and energy consumption, which is held in the fog broker. MGWO is used for task scheduling by the fog broker to analyze, estimate, and schedule sending requests from terminal devices. It saves energy and delays. Simulated results are compared with state-of-the-art algorithms. Compared with comparison algorithms, the proposed algorithm reduced energy consumption and delay. The increasing workload does not affect algorithms linearly. IoT devices generate many requests.

Yadav et al. [20] presented an opposition-based chemical reaction method for scheduling fog network tasks. They combined heuristic upward ranking and chemical reaction optimization techniques with opposition-based learning techniques. OBCR with OBL produces a more diverse population and helps escape local optima. In order to better explore and exploit the solution space, this algorithm utilizes four operators. Fog computing devices are more stable and have shorter service-time latency thanks to this technique. The proposed technique is more stable and has a shorter service-time latency than other approaches.

Mousavi et al. [21] proposed a constraint bi-objective optimization problem to minimize both energy consumption and response time for servers. D-NSGA-II is a non-dominated sorting genetic algorithm formed by adding a recombination operator to NSGA-II. In this algorithm, the exploration and exploitation abilities of the algorithm are balanced while the selection pressure of agents is controlled. The D-NSGA-II performed better than other algorithms in experiments. Additionally, it can respond to requests before their deadlines.

Table 1 compares related works in terms of the year, parameter considered, technique utilized, evaluation tool, and limitation. Although there are studies on minimizing cost and energy, response time must be reduced and load-balanced under deadline constraints. As part of our task scheduling algorithm with limited resource availability, we minimize response time, waiting time, and load balancing while considering deadline constraint tasks. Some papers fail to take into account the importance of meeting the deadline for each task in the fog computing network. Thus, we propose a task scheduling algorithm that reduces response times as well as waiting times and balances load across fog nodes. A fuzzy reinforcement learning task scheduling algorithm is proposed in which fuzzy logic is used to prioritize the tasks, while reinforcement learning is used to distribute the tasks to fog resources in the fog layer.

3- System model

This section provides the system model for the proposed system. In a fog computing environment, fuzzy reinforcement learning (FRL) is used to solve the task scheduling problem. As part of the innovation, FRL is used to schedule tasks while taking into account critical factors like deadline constraints, simultaneously minimizing response time and waiting

Table 1. Comparison of related works.

References	Year	Considered Parameter	Utilized technique	Simulator	Limitations
Ghanavati et al. [14]	2020	- Makespan - Energy	AMO	Matlab	- Network bandwidth effects are not considered, - Does not support dynamic, real-time task offloading for mobile users and varying network conditions.
Assamarai et al. [15]	2023	- Makespan - Task completion deadline	ACO	Java	- Energy consumption and system temperature are not addressed, - Evaluation focuses only on makespan and deadline satisfaction, ignoring other important metrics.
Ahmed et al. [16]	2021	- Makespan - Energy consumption	MFO, DE and OBL	iFogSim	- Assumes reliable virtual resources, overlooking the variability in real fog environment, - Computational complexity and scalability need more thorough analysis.
Ghafari and Mansouri [17]	2023	- Makespan - Energy consumption - Cost	EAVOA	Matlab	- The BWM for task prioritization is slow for large tasks, - Data privacy isn't addressed.
Guevara et al. [18]	2022	- Makespan - Processing cost	RL	Python	- The paper does not discuss the algorithm's scalability or the computational resources required, - Lacks comparison with state-of-the-art algorithms outside their framework.
Saif et al. [19]	2023	- Delay - Energy consumption	MGWO	Matlab	- It focuses narrowly on reducing delay and energy consumption, overlooking other critical objectives like cost and load balancing, - Does not consider the heterogeneity of resource.
Yadav et al. [20]	2022	- Latency - Stability	CRO and OBL	iFogSim	The impact of variable bandwidths between nodes is not considered, - Does not address privacy and security issues.
Mousavi et al. [21]	2022	- Energy consumption - Response time	D-NSGA-II	Matlab	- Dependencies between tasks are not considered, - Only energy consumption and latency are considered, excluding communication and computing costs.

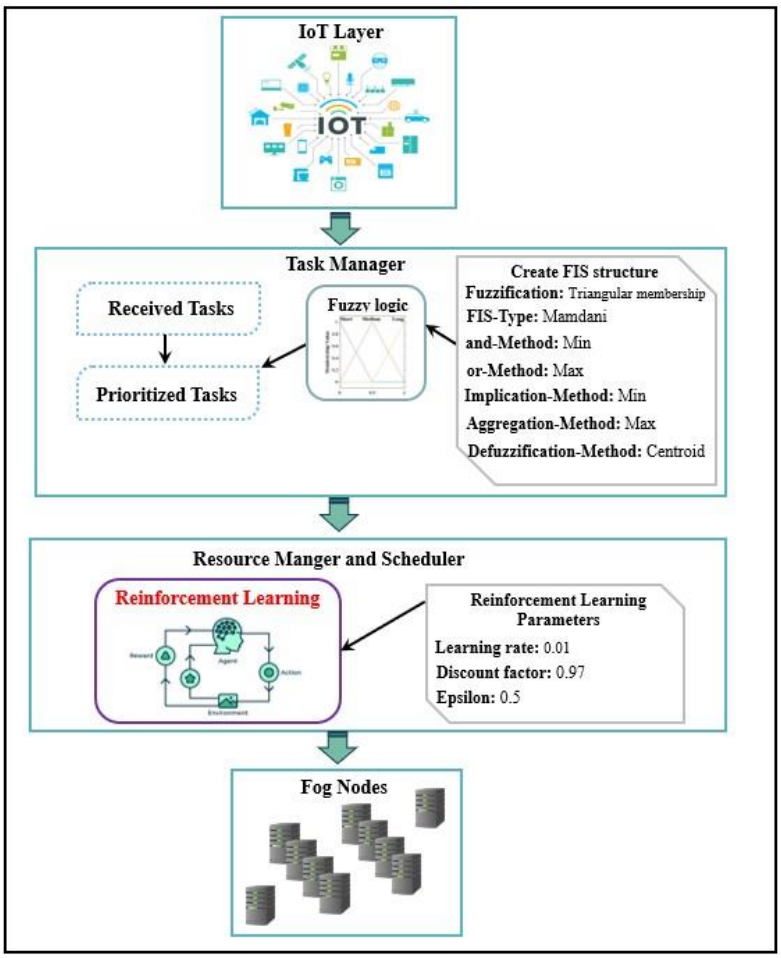


Fig. 3. The system model of the proposed task scheduling approach.

time, and balancing loads within the constraints of limited resources.

Using fuzzy logic and reinforcement learning, fuzzy reinforcement learning (FRL) addresses complex decision-making problems, such as those involving uncertainty and ambiguity. FRL can be highly effective in fog computing scenarios because it is capable of handling imprecise information and learning optimal scheduling policies over time. In order to optimize resource allocation and ensure timely processing, it is necessary to schedule tasks generated by various IoT devices efficiently to available fog nodes. Task scheduling is a complex problem as a result of the dynamic nature of the environment, the heterogeneity of tasks, and the varying capabilities of fog nodes.

The proposed FRL task scheduling algorithm is based on fuzzy logic for task prioritization and reinforcement learning for task distribution. Fuzzy logic is used to prioritize tasks based on deadline urgency, resource requirements, and task size. With fuzzy rules, tasks are categorized into different priority levels, which allows the system to handle inherent uncertainty. Prioritizing tasks is followed by reinforcement learning to determine the optimal distribution strategy. In order to minimize cumulative rewards, such as minimizing latency and balancing load, the RL agent interacts with the fog environment to determine which actions (task assignments) are optimal. In this way, ambiguity in task requirements and system states is effectively managed, and continually adapted to weather conditions, and scheduling efficiency is improved over time, resulting in better utilization of fog resources and enhanced system performance.

To solve the problem of task scheduling in fog environments, this approach seamlessly integrates fuzzy logic and reinforcement learning. Fuzzy logic is used to prioritize tasks, and RL is used to learn the optimal distribution of tasks so that tasks are scheduled effectively and in a manner that aligns with urgency and resource requirements. Due to the limited availability of resources, it ensures load balancing while considering deadline constraints. This approach is robust and highly effective for managing complex and dynamic scheduling tasks in fog computing environments due to its dynamic adaptation, facilitated by RL.

Figure 3 shows the proposed architecture for fuzzy reinforcement learning. This module shows how an effective RL-based algorithm called FRLS is used to handle response

time, waiting time, and load balance. Initially, the task manager receives a list of tasks from the user. Task managers accept tasks, and prioritize them. In the proposed algorithm, the priority is calculated and tasks are arranged according to the priority value. Resources and task information are then sent to the resource manager and scheduler. Priority information is received by the resource manager and scheduler. As a result, it allocates resources based on RL.

IoT devices produce a high volume of real-time delays-sensitive requests. The memory, processing power, and storage capacities of most IoT devices are limited. The proposed model considers tasks independent, non-preemptive, and not subdivided. T denotes a set of tasks i.e., $T=\{T_1,T_2,...,T_n\}$. A task i is defined as $T_i=(T_i^s,T_i^d,T_i^{in},T_i^{out})$. Let F be the set of heterogeneous fog nodes i.e., $F=\{F_1,F_2,...,F_m\}$. Each fog node is represented as $F_j=\{F_j^{PR},F_j^{Del}\}$. We define binary decision variables. $a_{ij}\in\{0,1\}$, for all $i\in T$, for all $j\in F$ stands for the allocation of the task T_i to the fog node F_j. $a_{ij}=1$ if the task T_i is assigned to the fog node F_j, 0 otherwise. Table A.1 in the Appendix provides a summary of symbols.

3- 1- Response time model

In the task scheduler, a response time is defined for the task T_i. The interval represents the time between gathering the input file and promoting the output file. F_j is the delay between the task scheduler module and node F_j^{Del}. Using Equation (2), the execution time of the task T_i on node F_j can be described as ET_{ij} and the time spent waiting over T_i in the queue as WT_{ij}. Based on these parameters, the time of response over the task T_i is estimated as follows [22]:

$$RT_i=\sum_{j=1}^{m}\left(2\times F_j^{Del}+ET_{ij}+WT_{ij}\right)\times a_{ij},\quad \forall i\in\{1,2,...,n\} \tag{1}$$

Where execution time is calculated as follows [22]:

$$ET_{ij}=\frac{T_i^s}{F_j^{PR}},\quad \forall T_i\in T,\forall F_j\in F \tag{2}$$

Where T_i^s represents the *i-th* task's size and F_j^{PR} is the processing rate of CPU on F_j.

3- 2- Load model

In order to maximize resource utilization, avoid bottlenecks, prevent overload and low load, as well as to reduce response times, load balancing is an important issue. The load on each node is calculated during this phase. The load of a node is the total length of all tasks assigned to it [23]:

$$\frac{TotalLengthofTask,time(t)}{CPU\ \text{Proces sing}RateofNode,time(t)} \tag{3}$$

Therefore, the load can be calculated by dividing the number of tasks in a node's service queue by its service rate at time *t*, using Eq. (3).

4- Proposed Fuzzy Reinforcement Learning Approach

Fuzzy logic-based scheduling algorithms are presented in this section. Task attributes are difficult to prioritize in a dynamic environment. Section 4.1 calculates task weights based on factors such as deadline, size, and file size. Section 4.2 presents a reinforcement learning task scheduling method for scheduling tasks between fog nodes.

4- 1- Calculation of task weights using fuzzy logic system

Fuzzy inference systems (FIS) use fuzzy logic to map input values to output values. As a result of its ability to adapt, interpret rules, and study a variety of inputs, fuzzy logic was the best solution for our problem. An efficient way to deal with uncertainties in a system is through fuzzy logic [24].

The task manager receives requests for task execution from end users. Next, the task priority value is calculated based on the task deadline, the task size, and the task file size. In this way, the task will be completed at the top of the priority list and with a shorter deadline. The proposed system uses fuzzy logic control to take into account three factors, namely the task deadline, the task size, and the file size. The input values are normalized in the interval (0,1). The normalization method is Min-Max. There are four steps involved in fuzzy inference, namely (1) Fuzzification (2) Predefined rule base (3) Fuzzy inference system and (4) Defuzzification. Figure 4 shows the architecture of a fuzzy model [25].

Fuzzy logic controllers take three input parameters: deadline, size, and weight of the task. Table 2 summarizes fuzzy inputs and outputs with linguistic value sets. The membership functions for inputs and outputs are shown in Fig. 5.

IF-THEN rules are fuzzy rules that are used to make decisions. It specifies control objectives and domain policies using linguistic rules. The FIS comprises rules that follow IF (conditions are met) THEN (set of results can be performed). Table 3 shows 27 fuzzy if-then mapping rules generated from input data. The Mamdani fuzzy inference system maps an input space to an output space based on predefined rules.

After that, the defuzzification process is completed. Defuzzification converts fuzzy values into crisp values by reversing fuzzification. The Center of Gravity (COG) method is used to defuzzify a document. Prioritize tasks based on their newly calculated priority. Tasks will determine resource manager and scheduler priorities. The priority of a task is determined by its size, deadline, and file size. Fog resources are used for high-priority tasks. In algorithm 1, tasks are scheduled using fuzzy logic. Algorithm 1 uses fuzzy logic to make scheduling decisions, followed by reinforcement learning.

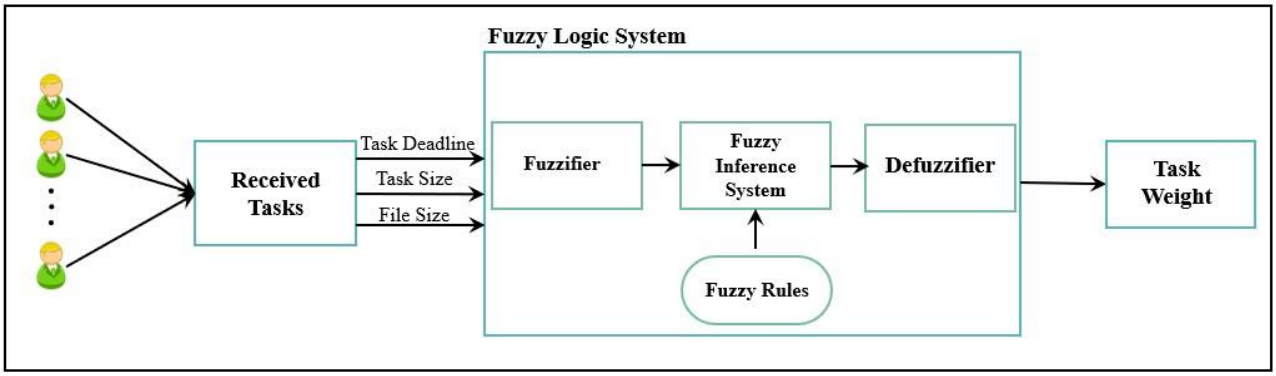


Fig. 4. Fuzzy model architecture.

Table 2. Input/output variables with linguistic values.

I/O variables	Linguistic variables
Task deadline	{Short, Medium, Long}
Task size	{Less, Medium, More}
Task file size	{Small, Medium, Huge}
Task weight	{Very Low, Low, Low Medium, High Medium, High, Very High}

4- 2- Reinforcement learning-based task scheduling algorithm

RL is vital to creating an accurate scheduling strategy in today's dynamic computing environment. One of the significant characteristics of RL is its ability to make decisions independently. RL involves learning a policy that achieves the best reward from interactions with an environment. To determine the optimal location for processing application tasks based on QoS requirements, the RL agent employs the Q-learning technique. The Q-learning method uses model-free reinforcement learning to update the Q-value function without estimating a model after each iteration with the environment [26].

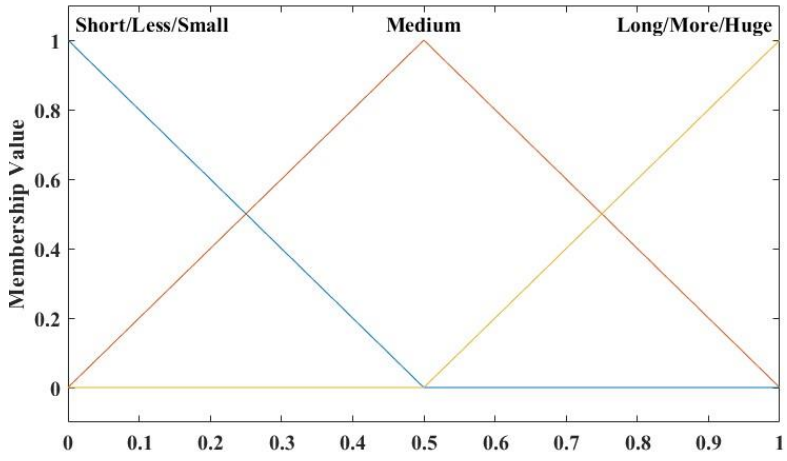
The fog computing was modelled using a Markov Decision Process (MDP), consisting of the following elements:

• **State Space (*S*):** Each fog node includes its status, the status of every IoT device generating tasks, and its current load.

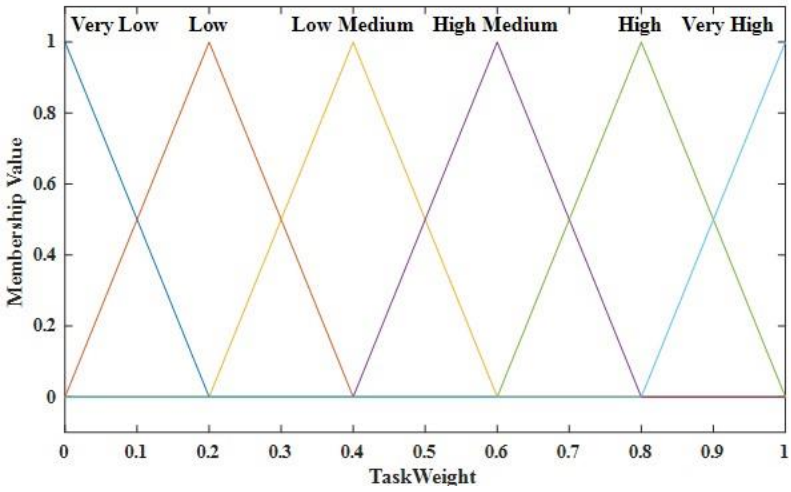
• **Action space (*A*):** The learning agents perform different actions on the environment based on a defined action selection policy in each learning episode. This algorithm uses all fog node resources to meet request requirements according to the environment state. Selection is based on load balancing and waiting time. Fog nodes are assigned to tasks through action sets.

• **Reward (*R*):** For every action taken, the environment rewards the agent immediately. This problem rewards based on response time. Agents receive a 1 reward if they respond before the deadline. When response time and deadline are equal, the agent receives a reward of 2. It receives reward 3 if it selects a node that cannot meet the deadline.

The reward function is designed to guide agents toward optimal performance within a fog computing environment by rewarding timely responses and penalizing missed deadlines. The reward structure encourages prompt task completion and efficient resource utilization by rewarding agents if they respond before the deadline. Optimal scheduling and optimal use of available time are rewarded if the response



(a)



(b)

Fig. 5. Membership function for task deadline, task size, file size, and task weight.

time matches the deadline. The agent will receive a reward of 3 if the selected node cannot meet the deadline. Even though this reward structure seems counterintuitive, it is intended to emphasize the importance of meeting deadlines within fog computing. It ensures that agents are motivated to act rapidly and precisely, balancing immediate response with optimal resource utilization. It emphasizes the importance of precise scheduling by providing higher rewards for timely responses and lower rewards for early completion. Missing deadlines carries the highest penalty (3), emphasizing the importance of timely task completion in fog computing. As a result, this method aligns agent behaviour with the system's performance goals, promoting efficient and reliable task scheduling. It can be expressed as follows [27]:

$$R = \begin{cases} 1 & if\ (RT_i - T_i^d) < 0 \\ 2 & if\ (RT_i - T_i^d) = 0 \\ 3 & if\ (RT_i - T_i^d) > 0 \end{cases} \quad (4)$$

Using a fog computing environment, the reward function is designed to encourage agents to respond quickly. The rewards are outlined below: The agent receives a reward of 1

Table 3. Fuzzy rule-base [25].

R. No.	Input variable			Output variable
	Deadline	**Task size**	**File size**	**Task weight**
1	Short	Less	Small	Very High
2	Short	Less	Medium	Low
3	Short	Less	Huge	Low
4	Short	Medium	Small	Low
5	Short	Medium	Medium	Low Medium
6	Short	Medium	Huge	Low Medium
7	Short	More	Small	Low Medium
8	Short	More	Medium	High Medium
9	Short	More	Huge	High Medium
10	Medium	Less	Small	Low
11	Medium	Less	Medium	Low Medium
12	Medium	Less	Huge	Low Medium
13	Medium	Medium	Small	High Medium
14	Medium	Medium	Medium	High Medium
15	Medium	Medium	Huge	High Medium
16	Medium	More	Small	High
17	Medium	More	Medium	High
18	Medium	More	Huge	High
19	Long	Less	Small	Low
20	Long	Less	Medium	Low Medium
21	Long	Less	Huge	Low Medium
22	Long	Medium	Small	Low Medium
23	Long	Medium	Medium	High Medium
24	Long	Medium	Huge	High
25	Long	More	Small	High
26	Long	More	Medium	Very High
27	Long	More	Huge	Very High

for responding before the deadline. Agents are encouraged to complete tasks quickly and efficiently as a result.

On-time Response (when response time equals the deadline): It receives a higher reward of 2 if it responds precisely by the deadline. A higher reward is given when tasks are managed within the expected time frame.

Inability to Meet the Deadline: In such a case, the agent receives the highest reward of 3. Despite its counterintuitive nature, this could be designed to penalize the agent more severely. A higher reward may indicate a more significant penalty or cost if you miss the deadline, depending on our system's design.

• **Action Selection Policy:** The ε-greedy policy is used here to select an action for each environment state. The policy of ε-greedy is described in Algorithm 2 [25].

• **Discount Factor (γ):** If the discount factor γ is 0, the agent learns the action by learning the immediate reward, and if it is 1, it learns the action by learning the cumulative sum of future rewards.

• **Update the value table of actions (Q-Table):** Q-table value is updated when reward signals are received from the environment. The action selected in the state determines the

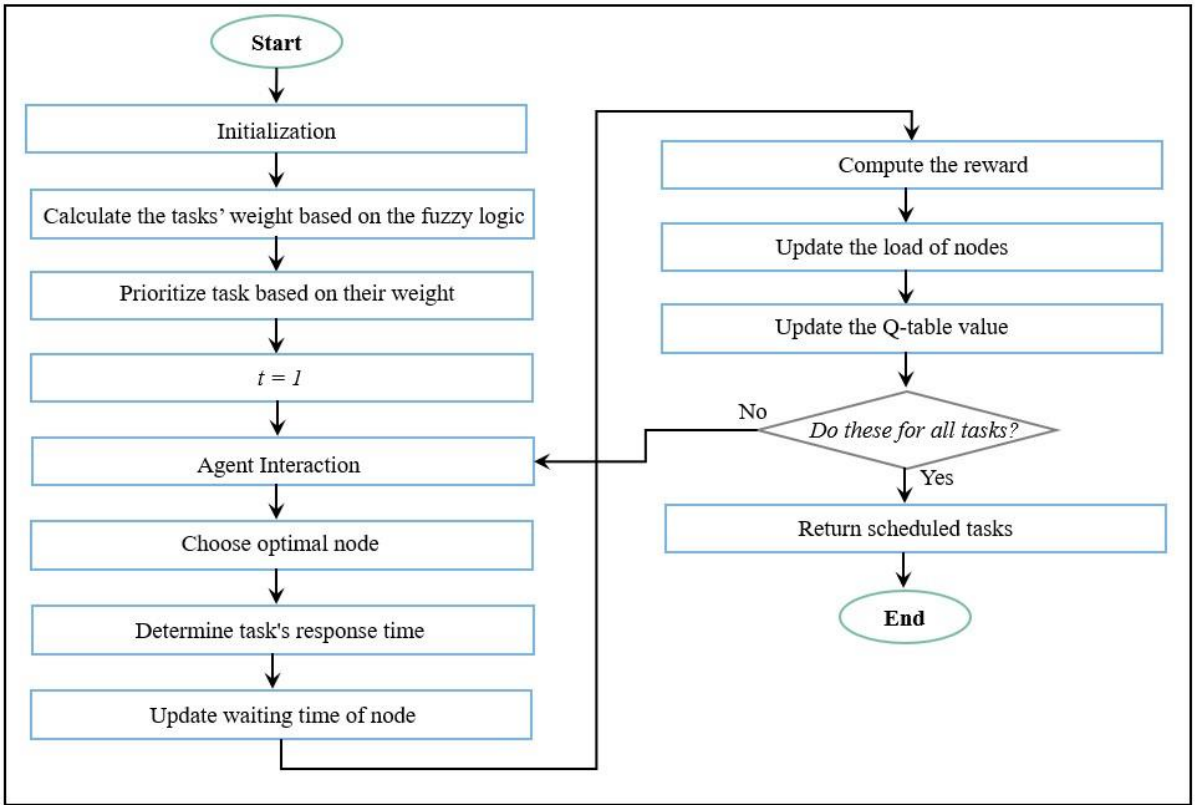


Fig. 6. Flowchart of proposed FRLS algorithm.

Q-value function. In the environment, the Q-value function should move to the best state when the agent selects an action. The Q-table is updated using Eq. (5):

$$Q(s,a) = (1-\alpha) \times Q(s,a) + \alpha \times \left[R + \gamma \times \min Q(s^{'}, a^{'}) \right] \quad (5)$$

Where s, a, α, and $s^{'}$ represent the current state, action taken in the current state, learning rate, and next state, respectively. The immediate reward achieved by executing action a in state s is symbolized by R, while $a^{'}$ denotes the action that minimizes the Q-value in state $s^{'}$.

Figure 6 shows the flowchart of the proposed algorithm. Furthermore, Algorithm 1 provides a detailed representation of the proposed algorithm.

5- Experimental result

The purpose of this section is to conduct a comprehensive, scientific, and rigorous simulation experiment to evaluate the effectiveness of the proposed fuzzy reinforcement learning task scheduling algorithm in a variety of scenarios.

5- 1- Simulation environment configuration

A comparison of FRLS with First Come First Serve (FCFS) [28], Max-Min [29], Earliest Deadline First (EDF) [30], MGGS [31], CODA [32], IWC [33], and FUGE [34] particularly cloud environments/computing. The dynamic and heterogeneous nature of resources in such distributed systems makes optimum job scheduling a non-trivial task. Maximal resource utilization in cloud computing demands/ necessitates an algorithm that allocates resources to jobs with optimal execution time and cost. The critical issue for job scheduling is assigning jobs to the most suitable resources, considering user preferences and requirements. In this paper, we present a hybrid approach called FUGE that is based on fuzzy theory and a genetic algorithm (GA algorithms was performed regarding the load of objectives, response time, deadline satisfaction percentage (DST%), and waiting time. During the simulation, three algorithms were compared at the same time. Initially, there are 60 fog nodes, and there are 100 to 500 incoming tasks. Second, the number of fog nodes ranges from 30 to 90, with 300 fixed IoT tasks. As a third case, the proposed algorithm (FRLS) compares to situations where the tasks are not prioritized, the number of nodes is 50, and the number of tasks is 100 to 500. The key parameters of our simulations are presented in Table 4. We used MATLAB on a computer with a core i5 running Windows to verify that our proposed algorithm is effective.

Algorithm 1 FRLS: Fuzzy Reinforcement Learning Scheduling Algorithm

Input: Task set, Fog node set

Output: Assigning incoming task to the most appropriate node

Initialize $\varepsilon, \alpha, \gamma$
Calculate the tasks' weight based on the fuzzy logic
For each task do
    For $x \in \left\{\left(T_i^s, T_i^d, T_i^{FS}\right)\right\}$ **do**
        Calculate $\mu(x)$ **using triangular member function**
    End
    For each rule do
        if $\mu\left(T_i^d\right), \mu\left(T_i^s\right), \mu\left(T_i^{FS}\right)$ **then**
            Fit the membership levels
            Find the Output linguistic level based on the fuuzy rule-base
        End
    End
    Aggregate output set using Max Aggregation technique
    Determine the crisp value from the defuzzification process for Task weights value
End
Prioritize task based on their weights
Initialize values of Q-table to zero
For each task do
    The agent engages with its environment, interacting with nodes, and examines all potential nodes utilizing the Q-table.
    It chooses the optimal node according to the Algorithm 2
    Determine the task's response time
    Update waiting time of node
    Compute the reward using Eq. (4)
    Update the load of nodes
    Update the Q-table value
End
return scheduled tasks

Algorithm 2: ε**-greedy policy**

Input: ε

Output: Action

Choose random number $rd \in [0,1]$
if $rd < \varepsilon$ **then**
    Select a random action ***//Exploration***
else
    Select the action with minimum Q-value ***//Exploitation***
end if
return action

Table 4. Parameter setting.

Parameter	Value
IoT task size (type 1)	[100–372] MI
IoT task deadline (type 1)	[100-500] ms
IoT task size (type 1)	[1028–4280] MI
IoT task deadline (type 1)	[500-2500] ms
Input file size	[0.3,1.5] MB
Output file size	[0.1,1] MB
Processing rate of CPU	[500–2000] MIPS
Delay	[1, 5] ms

5- 2- The impact of task numbers

As the number of tasks and fog nodes increases from 60 to 120, the performance of the proposed algorithm is evaluated. Figure 7 illustrates how increasing the number of tasks can affect the system's performance because it increases the load. With more tasks, response time increases. The fog nodes will have a longer wait time.

Figure 7(a) compares the load of the proposed FRLS algorithm with FCFS, Max-Min, EDF, CODA, MGGS, IWC, and FUGE algorithms. As shown in the chart, although algorithms that emphasize load balancing, such as the proposed FRLS algorithm FUGE, and MGGS, have higher load balance in all scenarios, our proposed algorithm performs better in most scenarios. FRLS increases the load balancing by 18%, 14%, 18%, 3%, 6%, 3%, and 6% compared to FCFS, Max-Min, EDF, CODA, MGGS, IWC and FUGE algorithms, respectively. In the proposed method, the Q-table values are used to determine a device for task scheduling using a greedy action selection policy. A reward is received based on the load and waiting time. Fog resources outperform other algorithms in calculating load parameters.

Figure 7(b) illustrates that the FRLS algorithm's response time is shorter than other algorithms since it considers the load on the selected fog node before performing the operation.

Figure 7(c) shows that tasks prioritized by fuzzy logic before scheduling are more likely to be on time with the proposed FRLS algorithm. CODA improves DST% by 26% compared to FRLS.

In Fig. 7(d), the proposed approach has a better waiting time than the other approaches. In the above experiment, the proposed strategy reduced waiting time by 28%, 21%, 26%, 6%, 11%, 12%, and 11% when compared with FCFS, Max-Min, EDF, CODA, MGGS, IWC and FUGE, respectively.

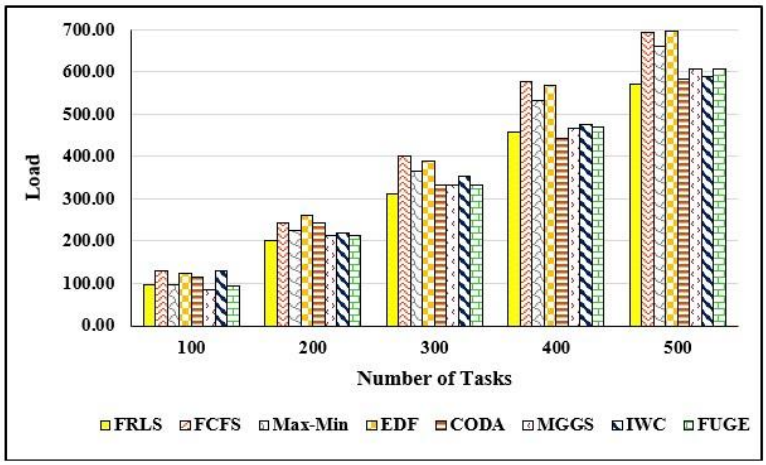
In fog computing task scheduling, the proposed algorithm uses reinforcement learning and fuzzy logic to achieve superior results. It continuously optimizes scheduling policies in response to changing environments using reinforcement learning to adapt dynamically to changing environments. Due to this adaptive learning capability, our algorithm remains robust and efficient even in dynamic and unpredictable fog computing scenarios. Moreover, fuzzy logic can be used to prioritize tasks intelligently and nuancedly based on multiple criteria, such as task size, file size, and deadline. Scheduling decisions are thus made more effectively and fairly than with traditional heuristics or meta-heuristics. The combination of these two powerful approaches improves response time and waiting time, resulting in more efficient resource utilization and dynamic load balancing.

5- 3- The impact of fog nodes

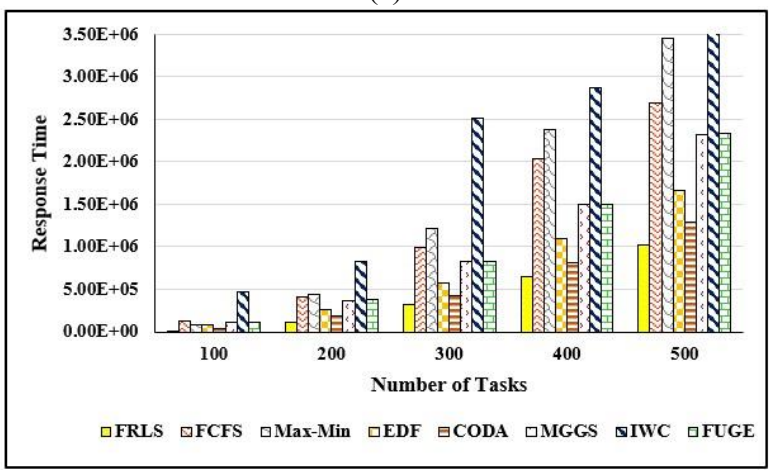
During this experiment, we increase the number of fog nodes from the set of [30, 50, 70, 90] in order to identify the effect on system performance. The number of tasks is limited to 300. Figures 8(a) to 8(d) show the results of this experiment.

According to Fig. 8(a), the proposed scheduling algorithm has obvious advantages in achieving load over the other algorithms. FRLS strategy has a superior load to FCFS, Max-Min, EDF, CODA, MGGS, IWC, and FUGE algorithms by 17%, 15%, 17%, 3%, 6%, 1%, and 7%, respectively, according to a contrast analysis of the experimental results EDF always has the highest load values.

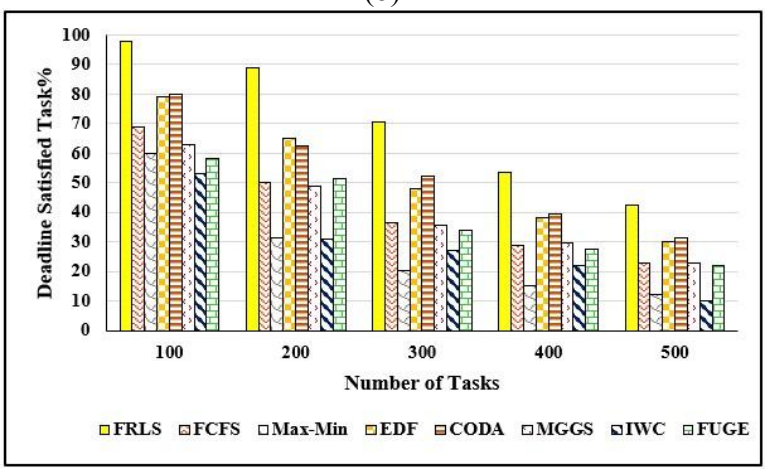
As shown in Fig. 8(b), the proposed algorithm and other algorithms have different response times. We aimed to reduce IoT application response times in these experiments. Response time is reduced more by the proposed algorithm than by the related approaches. Due to the consideration of deadlines when assigning tasks, EDF and CODA significantly reduce response time compared to FCFS and Max-Min. The proposed FRLS algorithm offers a 24% improvement over



(a)

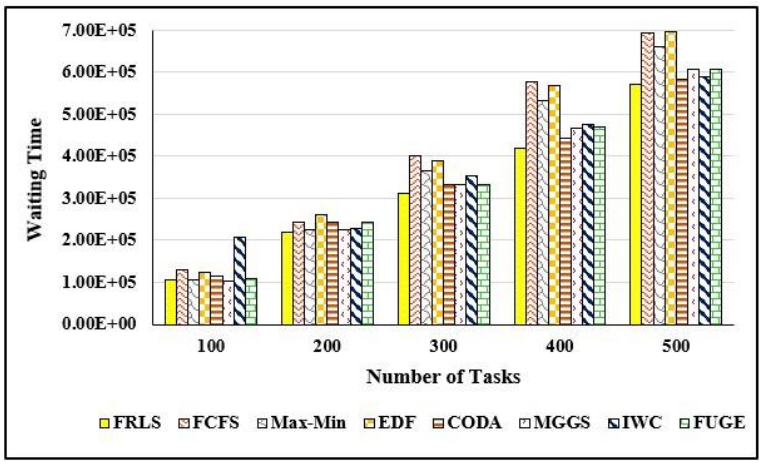


(b)



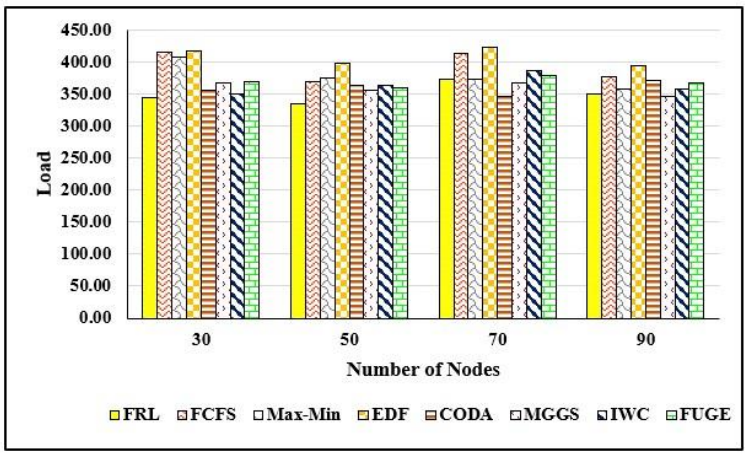
(c)

Fig. 7. Effects of increasing task numbers.(Continued)

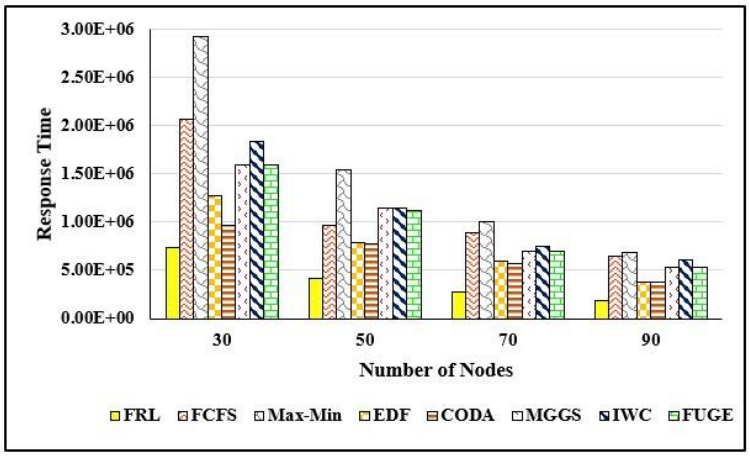


(d)

Fig. 7. Effects of increasing task numbers.

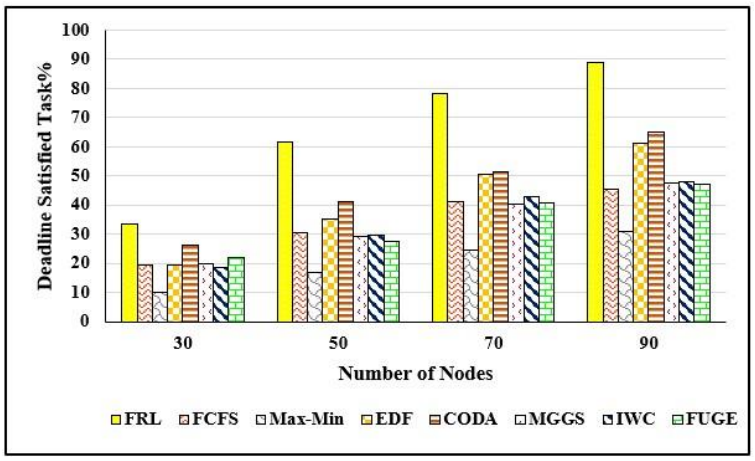


(a)

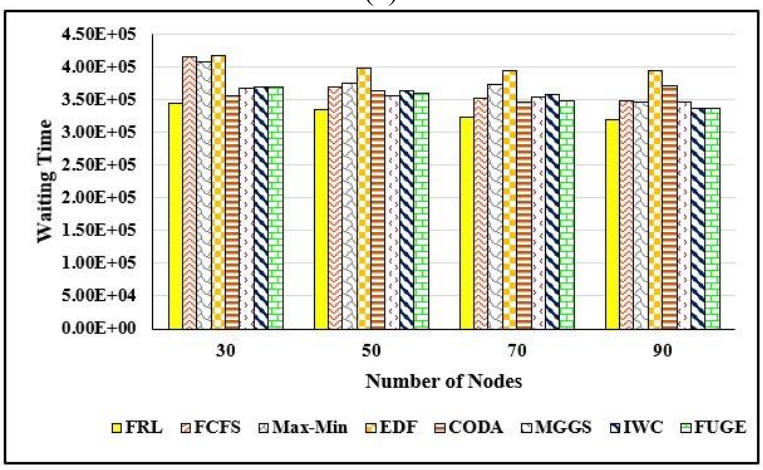


(b)

Fig. 8. Effects of increasing fog nodes. (Continued)



(c)



(d)

Fig. 8. Effects of increasing fog nodes.

the second-best algorithm for 30 instance numbers of fog nodes, compared to the second-best algorithm.

In general, the DST% of all algorithms increases as the number of fog nodes increases. Compared to the other algorithms, the proposed algorithm performs the best since it prioritizes tasks based on deadline requirements and considers task size and file size. As the number of fog nodes increases from 30 to 90, FRLS achieves 33% to 89% of the DST%. As a result, FRLS can provide great QoS for IoT requests. CODA provides the second-best results in these respects. Figure 8(d) shows that our proposed algorithms outperform other algorithms in terms of waiting time. In 30 through 90 instances of nodes, FRLS minimized waiting times by 17%–19% less than EDF. For 30 through 90 instances of nodes, FRLS minimized waiting time by 3%–14% less than CODA.

This algorithm excels in scalability and flexibility, making it well-suited to handle the increasing number of tasks and fog nodes found in modern fog computing environments. Moreover, fuzzy logic effectively deals with imprecise information and uncertainty, resulting in more reliable scheduling decisions in uncertain environments. Continuous improvement is made possible by reinforcement learning, resulting in sustained superior performance. Further, the approach enhances system stability and reliability by evenly distributing workload among fog nodes. In fog computing task scheduling, our algorithm reduces computation overhead by making efficient decisions, which makes it superior to more complex optimization techniques.

5- 4- The impact of incorporating fuzzy logic

We tested the proposed algorithm's performance on an increased number of tasks from 100 to 500 and a reduced number of fog nodes of 50 in this final experiment. A comprehensive evaluation of the algorithm assessed how

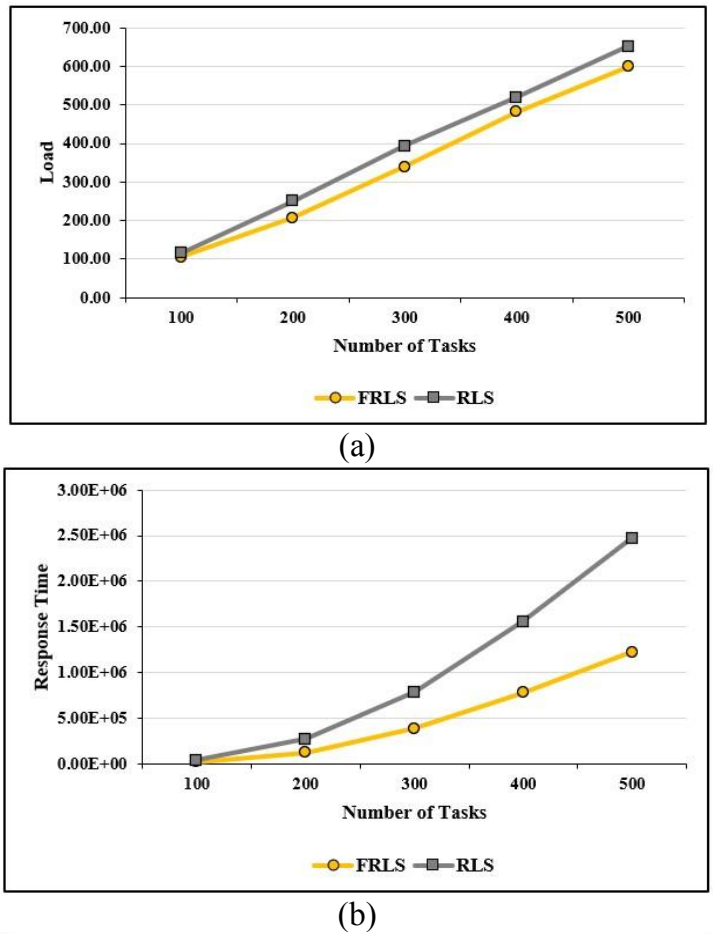
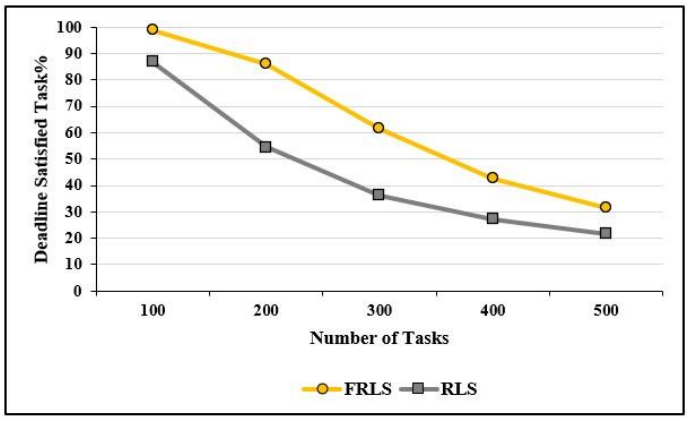


Fig. 9. Comparison of FRLS with and without fuzzy logic. (Continued)

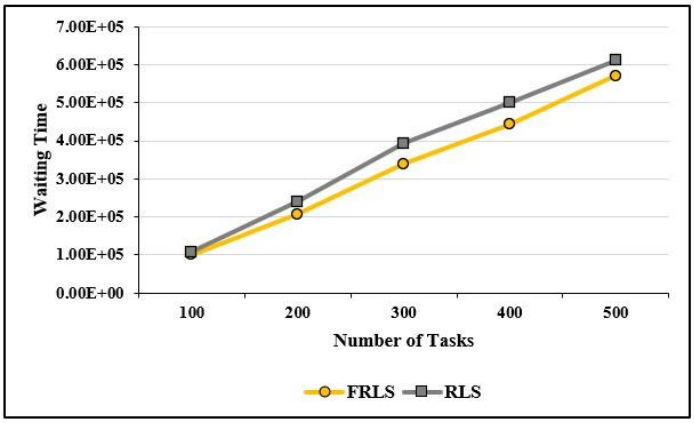
well it scales and adapts when computational demands and resources vary. Our primary focus was on comparing the Fuzzy Reinforcement Learning Scheduling (FRLS) algorithm (with the fuzzy logic step and task prioritization), with the Reinforcement Learning (RL) algorithm, which lacks these components. The evaluation criteria included several critical performance metrics: load balancing, response time, Deadline Satisfaction Task percentage (DST%), and waiting time.

This experiment revealed several important findings. Figures 9(a) and 9(b) illustrate that the proposed algorithm performs significantly better with an increasing number of tasks. With this performance improvement, the algorithm indicates that it is highly scalable and can handle large volumes of tasks efficiently without degrading its performance. Scalability is essential for real-world applications where tasks and fog nodes can vary dynamically and unpredictably. Furthermore, Figure 9(c) shows that the DST% (Percentage of Deadline Satisfaction Task) decreases with increasing task complexity. As a result of this trend, the complexity of tasks tends to require more time and computational resources. Despite this, we achieve higher deadline satisfaction rates using our FRLS algorithm. A key reason for this algorithm's success is that it meets deadlines, which is a crucial requirement for many applications, such as real-time data processing.

In this study, fuzzy logic is used to prioritize tasks. In dynamic and complex environments, fuzzy logic provides a robust framework for coping with uncertainty and imprecision. Fuzzy logic can be incorporated into the FRLS algorithm to prioritize tasks based on various factors, including task urgency, resource availability, and system load. By prioritizing tasks, the algorithm manages tasks more efficiently and meets deadlines more consistently. Further evidence of the benefits of fuzzy logic is provided



(c)



(d)

Fig. 9. Comparison of FRLS with and without fuzzy logic.

by a comparison between the FRLS algorithm and the RL algorithm. FRLS improves DST% by up to 31% compared to RL. A significant improvement in task scheduling and resource allocation can be achieved by using fuzzy-based prioritization. As shown in Figure 9(d), the proposed algorithm reduces waiting time significantly compared to RL. In order to improve overall system responsiveness and user satisfaction, waiting time must be reduced.

As a result of this experiment, fuzzy logic has important advantages for task prioritization in the FRLS algorithm. Fuzzy logic plays a critical role in enhancing reinforcement learning-based scheduling algorithms' performance in terms of load management, response time, deadline satisfaction, and waiting time. Fog computing applications require sophisticated and adaptive scheduling solutions because of dynamic task loads and resource constraints. This study shows that the FRLS algorithm, with its fuzzy logic-based prioritization, delivers high levels of efficiency and reliability when managing complex and dynamic tasks.

6- Conclusion

Fog systems face challenges in task scheduling due to the variability and dynamicity of the resources, as well as the increased volatility of customer service requests. The purpose of this paper is to propose a fuzzy reinforcement learning task scheduling method for improving load, response time, and waiting time. Fuzzy inference is used to prioritize tasks. The reinforcement learning algorithm is also used to schedule the tasks to the fog nodes, which improves load balancing and response time.

In this study, we compared FRLS with existing scheduling algorithms. Regarding load, response time, waiting time, and percentage of deadlines met, the proposed task scheduling mechanism outperformed the existing algorithms. It is possible to extend the scheduling criteria by adding some additional parameters such as costs, energy consumption, budget, and resource elasticity. Data privacy and security can also be addressed in the future, as well as scheduling.

Appendix

Table A. 1. Summary of symbols.

Symbols	Description
$\boldsymbol{T}$	List of tasks
$\boldsymbol{F}$	List of fog nodes
$\boldsymbol{n}$	Number of tasks
$\boldsymbol{m}$	Number of fog nodes
T_i^s	*i-th* task's size
T_i^d	*i-th* task's deadline
T_i^{in}	*i-th* task's input file size
T_i^{out}	*i-th* task's output file size
T_i^{FS}	*i-th* task's file size
F_j^{PR}	*j-th* fog node's CPU processing rate
F_j^{Del}	the *j-th* fog node's delay
a_{ij}	Binary decision variables

References

[1] Salaht, F. A., Desprez, F., and Lebre, A. "An overview of service placement problem in fog and edge computing." ACM Computing Surveys (CSUR), Vol. 53, No. 3, (2020), 1–35.

[2] Archana, R. "Multilevel scheduling mechanism for a stochastic fog computing environment using the HIRO model and RNN." Sustainable Computing: Informatics and Systems, Vol. 39, , (2023), 100887.

[3] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile cloud computing (pp. 13–16).

[4] Fahimullah, M., Ahvar, S., Agarwal, M., and Trocan, M. "Machine learning-based solutions for resource management in fog computing." Multimedia Tools and Applications, Vol. 83, No. 8, (2024), 23019–23045.

[5] Rahimikhanghah, A., Tajkey, M., Rezazadeh, B., and Rahmani, A. M. "Resource scheduling methods in cloud and fog computing environments: a systematic literature review." Cluster Computing, (2022), 1–35.

[6] Ghafari, R., and Mansouri, N. "A novel energy-based task scheduling in fog computing environment: an improved artificial rabbits optimization approach." Cluster Computing, (2024), 1–46.

[7] Iftikhar, S., Gill, S. S., Song, C., Xu, M., Aslanpour, M. S., Toosi, A. N., Du, J., Wu, H., Ghosh, S., and Chowdhury, D. "AI-based fog and edge computing: A systematic review, taxonomy and future directions." Internet of Things, Vol. 21, , (2023), 100674.

[8] Zabihi, Z., Eftekhari Moghadam, A. M., and Rezvani, M. H. "Reinforcement Learning Methods for Computation Offloading: A Systematic Review." ACM Computing Surveys, Vol. 56, No. 1, (2023), 1–41.

[9] Gasmi, R., Hammoudi, S., Lamri, M., and Harous, S. "Recent Reinforcement Learning and Blockchain Based Security Solutions for Internet of Things: Survey." Wireless Personal Communications, Vol. 132, No. 2, (2023), 1307–1345.

[10] Abdullah, L. "Fuzzy multi criteria decision making and its applications: a brief review of category." Procedia-Social and Behavioral Sciences, Vol. 97, , (2013), 131–136.

[11] Jalali Khalil Abadi, Z., and Mansouri, N. "A comprehensive survey on scheduling algorithms using fuzzy systems in distributed environments." Artificial Intelligence Review, Vol. 57, No. 1, (2024), 4. https://doi.org/10.1007/s10462-023-10632-y

[12] Al-Araji, Z. J., Ahmad, S. S. S., Kausar, N., Anis, F. G., Ozbilge, E., and Cagin, T. "Fuzzy Theory in Fog Computing: Review, Taxonomy, and Open Issues." IEEE Access, (2022).

[13] Ebneyousef, S., and Shirmarz, A. "A taxonomy of load

balancing algorithms and approaches in fog computing: a survey." Cluster Computing, (2023), 1–22.

[14] Ghanavati, S., Abawajy, J. H., and Izadi, D. "An energy aware task scheduling model using ant-mating optimization in fog computing environment." IEEE Transactions on Services Computing, (2020).

[15] Alsamarai, N. A., Uçan, O. N., and Khalaf, O. F. "Bandwidth-Deadline IoT Task Scheduling in Fog–Cloud Computing Environment Based on the Task Bandwidth." Wireless Personal Communications, (2023). https://doi.org/10.1007/s11277-023-10567-1

[16] Ahmed, O. H., Lu, J., Xu, Q., Ahmed, A. M., Rahmani, A. M., and Hosseinzadeh, M. "Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing." Applied Soft Computing, Vol. 112, , (2021), 107744. https://doi.org/10.1016/j.asoc.2021.107744

[17] Ghafari, R., and Mansouri, N. "E-AVOA-TS: Enhanced African vultures optimization algorithm-based task scheduling strategy for fog–cloud computing." Sustainable Computing: Informatics and Systems, Vol. 40, , (2023), 100918.

[18] Guevara, J. C., Torres, R. D. S., Bittencourt, L. F., and Da Fonseca, N. L. S. "QoS-aware Task Scheduling based on Reinforcement Learning for the Cloud-Fog Continuum." 2022 IEEE Global Communications Conference, GLOBECOM 2022 - Proceedings, (2022), 2328–2333. https://doi.org/10.1109/GLOBECOM48099.2022.10001644

[19] Saif, F. A., Latip, R., Hanapi, Z. M., and Shafinah, K. "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing." IEEE Access, Vol. 11, , (2023), 20635–20646.

[20] Yadav, A. M., Tripathi, K. N., and Sharma, S. C. "An opposition-based hybrid evolutionary approach for task scheduling in fog computing network." Arabian Journal for Science and Engineering, Vol. 48, No. 2, (2023), 1547–1562.

[21] Mousavi, S., Mood, S. E., Souri, A., and Javidi, M. M. "Directed search: a new operator in NSGA-II for task scheduling in IoT based on cloud-fog computing." IEEE Transactions on Cloud Computing, (2022).

[22] Mujtiba, H. S., and Rasool, B. G. "Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog-cloud environment [J]." Journal of Computational Science, Vol. 64, , (2022), 101828.

[23] Bhardwaj, T., and Sharma, S. C. "Fuzzy logic-based elasticity controller for autonomic resource provisioning in parallel scientific applications: a cloud computing perspective." Computers & Electrical Engineering, Vol. 70, , (2018), 1049–1073.

[24] Mothku, S. K., and Rout, R. R. "Fuzzy logic based adaptive duty cycling for sustainability in energy harvesting sensor actor networks." Journal of King Saud University-Computer and Information Sciences, Vol. 34, No. 1, (2022), 1489–1497.

[25] Raju, M. R., and Mothku, S. K. "Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach." Computer Networks, Vol. 224, , (2023), 109603.

[26] Hortelano, D., de Miguel, I., Barroso, R. J. D., Aguado, J. C., Merayo, N., Ruiz, L., Asensio, A., Masip-Bruin, X., Fernández, P., and Lorenzo, R. M. "A comprehensive survey on reinforcement-learning-based computation offloading techniques in Edge Computing Systems." Journal of Network and Computer Applications, Vol. 216, , (2023), 103669.

[27] Talaat, F. M., Saraya, M. S., Saleh, A. I., Ali, H. A., and Ali, S. H. "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment." Journal of Ambient Intelligence and Humanized Computing, Vol. 11, No. 11, (2020), 4951–4966.

[28] Zhao, W., and Stankovic, J. A. "Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems." In 1989 Real-Time Systems Symposium (pp. 156–157). IEEE Computer Society.

[29] Aladwani, T. "Types of task scheduling algorithms in cloud computing environment." Scheduling Problems-New Applications and Trends, (2020).

[30] Stankovic, J. A., Spuri, M., Ramamritham, K., and Buttazzo, G. Deadline scheduling for real-time systems: EDF and related algorithms (Vol. 460). Springer Science & Business Media.

[31] Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., and Chowdhury, M. U. "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments." Neural Computing and Applications, Vol. 32, , (2020), 1531–1541.

[32] Ghafari, R., and Mansouri, N. "An efficient task scheduling in fog computing using improved artificial hummingbird algorithm." Journal of Computational Science, Vol. 74, , (2023), 102152.

[33] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., and Murphy, J. "A woa-based optimization approach for task scheduling in cloud computing systems." IEEE Systems journal, Vol. 14, No. 3, (2020), 3117–3128.

[34] Shojafar, M., Javanmardi, S., Abolfazli, S., and Cordeschi, N. "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method." Cluster Computing, Vol. 18, No. 2, (2015), 829–844. https://doi.org/10.1007/s10586-014-0420-x

HOW TO CITE THIS ARTICLE
R. Ghafari, N. Mansouri, Efficient scheduling algorithm for optimizing system load in fog computing environment: A fuzzy reinforcement learning mechanism, AUT J. Model. Simul., 56(1) (2024) 33-54.

DOI: 10.22060/miscj.2024.23205.5360