A Novel Approach for Heterogeneous Task Allocation in Mobile Crowd Sensing Using Deep Reinforcement Learning Based on Reverse Stackelberg Game

Zohreh Vahedi[1], Seyyed Javad Mahdavi Chabok[1*], Gelareh Veisi[1]

[1] Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

Abstract-- Today, the cloud space has become the main platform for communication and interactions between users. With the advent of Internet of Things (IOT) technology, cloud systems face a huge volume of requests every day that need to be implemented in real time and efficiently. There are many situations where the exact determination of the status of the requests and how to implement them face serious challenges. In fact, in the IOT environment with the large variety of users and the modeling of many heterogeneous requests from them, it is crucial to adopt a flexible approach. Therefore, deriving a suitable scheduling mechanism that can minimize both task execution delay and cloud resource utilization is of great importance. One of the most common methods offered to manage the huge activities created in the network is the use of resources placed in the edge nodes of the network under the title of Mobile Crowd Sensing. However, in various applications of mobile crowd sensing, such as in the field of health or in intelligent transportation systems, the structure of requests sent by actors has a very high diversity. This diversity in requested activities and the high number of requests suggest the need for proper management of available resources. In this article, the Reverse Stackelberg game theory method consisting of fuzzy logic is used in order to quickly adopt an effective strategy for the optimal allocation of resources by gaining experience from its past performance. In this regard, in order to achieve the desired quality in assigning heterogeneous tasks among users, deep learning is used, which has useful features such as being online and highly adaptable. On the other hand, in order to perform tasks that require low delay, information about the location of the user and mobile, it is necessary to use the capabilities of the fog processing environment in interaction with the cloud space so that the capacity of the end layers of the network can be used well. The obtained results show that by using the proposed approach, more than 35% of CPU usage cost is saved compared to other state of the art methods.

Index Terms-- Mobile Crowd Sensing, Task Assignment, Deep Reinforcement Learning, Internet of Things, Reverse Stackelberg game, Fog space.

Tob Regul Sci. ™ 2022;8(1): 3927-3951
DOI: doi.org/10.18001/TRS.8.1.297

I. INTRODUCTION

Today, with the increasing use of sensor-equipped smartphones, Mobile Crowd Sensing (MCS) has become an emerging paradigm for implementing urban monitoring tasks. Conventional urban measurement systems (such as air quality monitoring stations and surveillance cameras) usually require special infrastructure with high costs to maintain and develop. MCS is a new model in which citizens share the data generated on their mobile devices and developers use these data to extract demographic information and provide people-oriented services. Mobile crowd sensing has emerged as a new model of monitoring that has effectively taken advantage of the collective intelligence and mobility of users in combination with the advanced capabilities of smart gadgets. The increase of mobile devices and the progress of the communication industry, computing power, storage space and multi-model monitoring capabilities have created this new monitoring model under the title of mobile crowd sensing or human-centered monitoring. In fact, mobile crowd sensing as an important structural platform can serve in new applications of the Internet of Things (IoT). In this model, various sensor devices produce a large amount of data and many resources such as bandwidth, memory and energy are consumed in this direction.

With the advent of IoT technology, traditional structures such as the use of cloud space have not been able to meet the needs of subscribers because they have faced many challenges, including bandwidth limitations and long service delays. According to CISCO's estimate, by 2022, more than 50 billion devices will be able to communicate with the IoT infrastructure [1]. In this situation, in order to meet needs such as improving bandwidth capacity, location information, using delay-sensitive services, and considering the equipment's mobile mode, it is necessary to use an interface between IoT equipment and the cloud space. This interface, which is known as fog computing, is defined as follows [2]:

Fog computing is a virtual platform capable of providing processing services, storage, and network functions between edge devices and traditional cloud computing data centers.

Fog computing is an extension of cloud computing that is serviced by edge nodes in the network, whose most important capabilities can be used in functions that require little location and mobility. In this case, people and vehicles with smart sensors can act as users in the fog computing environment and send monitored data to Data Centers (DCs) [4]. Despite the potential advantages of the fog processing, there are still challenges related to the dynamism, heterogeneity, and high complexity of requests sent by users that need to be evaluated. In this regard, one of the important challenges is the amount of energy consumed to complete MCS tasks, which includes collecting data obtained from sensors, local data and possibly transferring predicted data to the cloud server. Since the users who participate are significantly assigned the amount of energy needed to perform the tasks, minimizing the energy consumption on the users is crucial [5]. Another challenge is the scheduling of tasks in an efficient manner, so that inappropriate scheduling of tasks can lead to the non-effective use of hardware resources [6]. In fact, in order to have a suitable level of Quality of

Service (QoS), it is necessary to manage resources well, which can be classified into 6 categories: Application placement, Resource Scheduling, offloading, load balancing, resource allocation and resource provisioning. The details of the resource management classification are shown in Fig. 1. Resource allocation includes three types of functions: Resource mapping, Resource utilization, and Resource monitoring. Scheduling mechanism is very necessary to optimally use servers and use resources to enhance their performance. When fog resources are requested by multiple tasks, it becomes necessary to use an efficient scheduling approach for task allocation [7]. In the following, different resource allocation approaches to improve QoS are studied.

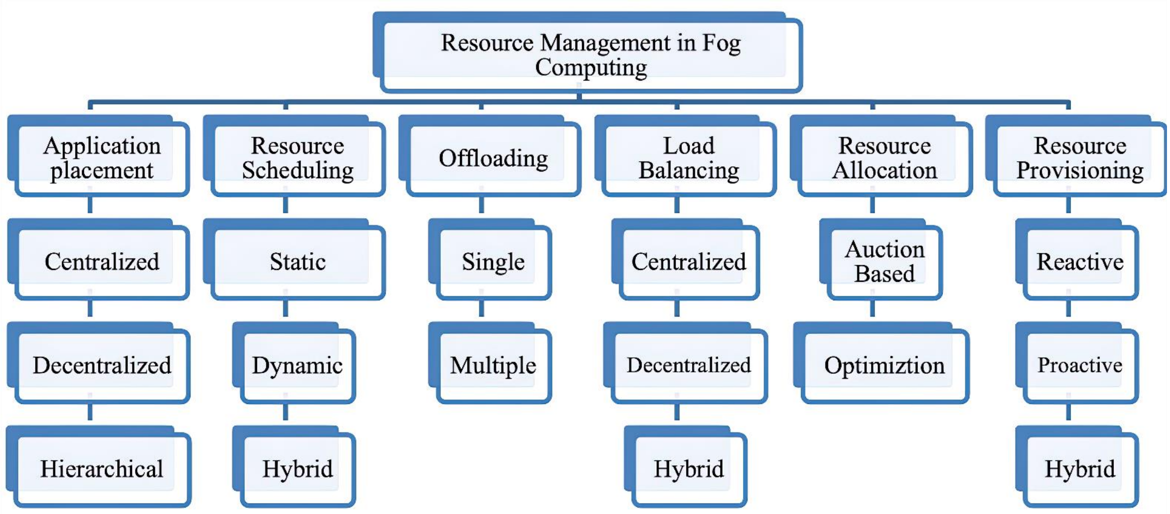

Fig. 1. Classification of resource management in fog computing [8]

In reference [9], to provide better response services in cellular networks, Femtocell and Picocell with different transmission and coverage capacities have been used to load user equipment. The purpose of providing this solution is to maximize long-term exploitation by considering the requirements of service quality level. In this regard, the Double Deep Q-network (DDQN) strategy is used, which can determine the possible optimal working point by approximating the active value function. Although this approach has advantages such as the possibility of using a large number of subscribers from the subchannel at the same time and using the event-triggered method to reduce energy consumption in the learning process, but when the density of stations is determined to be higher than the limit, challenges such as interference Severe differences between sub-station performance, loss of some system capabilities, service capacity reduction and high energy consumption can be predicted.

In reference [10], differential game theory is used to find the optimal amount of processing resources in order to maximize the income for users. In this regard, the cost of delay including the overflow caused by the delay of the network provider's performance and the creation of long queues created for the services of the cloud-fog structure is considered and calculated from Bellman's dynamic optimal planning technique for the feedback Nash equilibrium point.

Reference [11] uses an incentive mechanism based on DRL in game theory. The proposed game theory includes several leaders and several followers. In this situation, leaders and followers are numerous and mobile, and this itself makes the problem difficult. The proposed approach in this article consists of two parts. In the first part, the allocation of resources, activities and tasks to mobile users has been done, which includes a model with several leaders and several followers, and Stackelberg's game theory has been used to implement it. In the second part, the Stackelberg equilibrium point is calculated and in the continuation of the game, it becomes a multi-state or multi-agent Markov decision process, and based on an incentive mechanism, it helps the players to learn their optimal strategy directly from the game history. In the above-mentioned articles, the amount of energy consumed is not taken into account and has not been calculated.

In reference [12], in order to optimally plan the activities of multiple sensors, they used the method of analyzing the problem of scheduling sensor tasks with the ability to Minimum Energy Single-sensor task scheduling (MESS) in order to not only minimize the amount of energy consumption but also guarantee the quality of service. In [13], the writers proved that the task allocation problem of minimizing the maximum aggregate sensing time is NP hard even under the offline model. They have used a polynomial-time approximation algorithm for offline allocation and polynomial-time greedy algorithm for online allocation model. Although their approach can achieve high energy efficiency, they did not consider heterogeneous tasks. One of the solutions for implementing multi-task mobile crowd sensing is to use the Active Crowd approach, which has been used by reference authors [14] for two modes of passenger movement (intentional movement and unintentional movement). For this purpose, the Greedy genetic algorithm was used to ask users to move to the location of the request in order to minimize the distance in the case of delay-sensitive tasks. The plan presented in [15] has used the capabilities of fog computing for the hierarchical scheduling strategy, but the obvious problem in this plan is the inability of the plan to deal with the fluctuations in user requests in the fog layer and the unstable situation in the cloud task processing queue.

In [16], the Deep Q Network (DQN) approach has been used to allocate resources in Mobile Edge Computing (MEC). The optimizer system is defined in terms of minimizing energy cost, computational cost and delay cost. In this research, the allocation of activities is done offline and due to the longtime delay in the presented approach, it cannot be used well in online problems. In [17], calculation in fog space is used to assign functions. The investigated criteria include service quality, bandwidth and time delay reduction. In this regard, the deep learning model has been used to define the decision reward. The main challenge raised in this research is the homogeneity of activities. In general, fog computing faces two major challenges of bandwidth limitation and energy supply resources [18]. In [19] Minimum Bandwidth Code (MBC) was utilized to pool the underutilized edge node's computing resources and minimum latency codes for providing trade-off between computation load and computation latency.

However, the coding size cannot be used as a general approach for fog computing. A similar work has been done in references [20-21] to reduce the amount of coding, which can only be implemented for special application scenarios. In research [22], the DRL method has been used to find the optimal answer for resource allocation. This approach uses DDQN to examine the problem as a path planning challenge. This method depends on the location and is sensitive to the time of occurrence of events, and at the same time, the limitation of resources is considered in maximizing the distance. In general, one of the major challenges that arise during the training of agents with gradient-policy algorithms is the possibility of losing the target due to the weak and sudden actions of the agents. Recovering from the situation in this case is very difficult because the agent starts weak traces and uses them to train policies. In such a situation, the algorithms with policies are no longer able to use the data and these data samples are not efficient for them. For better evaluation, the comparison between different task allocation approaches is presented in table 1.

Table 1. Comparison of the methods used for task allocation

Ref.	evaluation criteria	Approach
[7]	Response time, cost and number of user losses	Lyapunov optimization method
[9]	Total runtime and resource cost	Multi-objective task scheduling algorithm with adaptive neighborhood strategy
[8]	Weighted sum of delay and energy	Deep reinforcement learning method
[10]	Response time	Particle Swarm Optimization
[11]	quality of service	task clustering based on fuzzy logic and linear optimization method
[12]	Energy consumption, the amount of profit considering resource limitations	Convex optimization for power allocation and genetic algorithm for programming
[14]	Execution time	An improved mixed-integer linear programming greedy task allocation method
[16]	Response time and cost	Heuristic multi-objective method based on fuzzy
[17]	Response time, fog space energy consumption and system lifetime	Improved heuristic method for task allocation

Since the different positions of the fog nodes and their performance have a direct impact on the quality of service of IoT applications, therefore, in [23], a scheduling problem was formulated in the hierarchical structure of the fog and this problem was solved with the help of a strategy based on DRL. Based on the simulation results, the proposed scheme can provide better performance than the existing scheduling methods. In [24], the problem of optimal selection of mobile vehicles to maximize the amount of information collected in specific time intervals has been studied and investigated. Evaluation results show that the proposed method can achieve the better performance and the effectiveness compared with the scheme without learning-based algorithms. In [25], an approach based on the prediction of repetitive behavior of users and the use of smartphones in collecting information from the environment in assigning monitoring tasks to users is presented. Considering that the location of the user plays an important role in the costs, therefore monitoring tasks have been allocated according to the potential paths of the participating users, which in addition to reducing the monitoring cost, the quality of the data has also improved. In this regard, the two objectives of reducing the cost of measurement and increasing the quality have been investigated. In [26], a new task assignment framework is presented that predicts the movement model of users as much as possible and maps it to the sequence of existing tasks. Based on the repetitive movement pattern, the task allocation problem becomes a pattern matching problem. The results of experiments on real world data and simulated data show the effectiveness of the proposed framework in assigning mobile mass monitoring tasks. In [27], a modified clustering algorithm and a routing protocol for mobile mass monitoring networks are introduced under the title of Energy Efficient and Aware Buffer Protocol (EBRP), in which the dynamic characteristics of nodes are well taken into account. The performance evaluation of the proposed protocol shows that a significant improvement has been made in terms of data delivery rate, cost and node survival rate. In [28], a two-phase hybrid worker recruitment framework named Hy Selector, which recruits workers in two phases, has been presented. First in the offline phase, with the help of the idea of influence propagation in communication and social network, a plan to employ people voluntarily during their daily activities is presented to solve the problem of slow start in MCS. Then, in the online phase, in order to reduce the computational complexity, an algorithm is provided to motivate the participants to complete the monitoring tasks in the monitored areas. In [29], a multi-task allocation problem is presented that considers the heterogeneity of participants in system decisions for resource allocation. In this method, a version of greedy particle swarm optimization in combination with genetic algorithm is presented, whose goal is to maximize the number of completed tasks by considering certain constraints. The simulations performed on the real data set show that the algorithm performs well under different conditions and parameter settings.

Game theory is a powerful framework that can optimize interactions between multiple actors who behave based on personalized goals. In fact, this approach can be used in the design of decentralized mechanisms where all actors seek to achieve their own interests. For example, in [30-31] a platform-based incentive mechanism for MCS using the Stackelberg game approach is presented. In [32], a multimedia application of quality-aware MCS is presented based on the Stackelberg game. In [33], a delay-sensitive approach for use in game theory is introduced. In [34], collective monitoring systems are modeled with a non-participatory two-stage game and the behavior of leading actors is investigated under the participation of all actors. However, in all the studied articles, the activity function model considered for the leading actors is assumed to be the same, which is not consistent in the IOT space and considering the types of activities desired by users. The proposal of matching Stackelberg games and inverse Stackelberg games by considering incentives in control applications was first proposed in 1980. Examples of the use of this approach include strategies lacking complete information [35-36], systems including uncertainty [37], modeling the delay in communication [38], calculating the behavior of different actors [39], and estimating the current price in the electricity market [40], load adaptive control [41] and machine planning [42]. The comparison results of different ideas in MCS are shown in table 2.

Table 2: Comparison of task allocation's methods (Idea & Target parameters)

Ref.	Year	Idea	Target parameters
[43]	2014	Allocating monitoring tasks to users based on predicting users' repetitive behavior	Monitoring cost, data quality
[44]	2014	Optimization techniques of random networks and distributed scheduling	Resource efficiency
[45]	2016	Allocation of monitoring tasks using linear techniques and greedy method	Users' time limit
[46]	2018	Optimizing the allocation of monitoring tasks using the combination of greedy particle swarm algorithm and genetics	Number of tasks completed
[47]	2018	Reducing temporal and spatial correlation between heterogeneous tasks with heterogeneous multi-task assignment problem design and greedy search algorithm	Data quality, service cost
[48]	2018	Allocation of tasks based on Simulated Annealing method (SA)	Energy consumption, Real timeness, coverage quality

According to the review of the articles, it was found that the optimal allocation of tasks in mobile collective monitoring, taking into account the heterogeneity of activities and several goals, such as reducing the cost and energy consumption in fog computing infrastructures and improving service quality indicators, has not yet been comprehensively investigated. Therefore, in this article, the DRL has been used to allocate tasks in mobile crowd sensing. This technique is able to achieve an effective scheduling strategy over time automatically and based on its previous experiences. In this situation, the participation rate of users is adjusted in an adaptable and flexible manner based on the quality of the requested service, and to attract more participation when the participation rate is lower, dynamic pricing of the service cost has been used. The main contributions of this paper can be mentioned as follows:

- Use of inverse Stackelberg game theory for heterogeneous task allocation

- Using DRL algorithm to determine the reward amount of each agent in order to find the appropriate strategy

The structure of the paper is as follows:

In section 2, the basic concepts of reinforcement learning and Stackelberg game theory will be expressed. In the third section, the mathematical model of the problem, objective function, and data set characteristics will be discussed. In the next section, the obtained results are evaluated with using of figures and tables, and in the last section, conclusion will be presented.

II. Basic Concepts

In this section, the basic concepts related to reinforcement learning and Stackelberg game theory are explained.

A. Reinforcement learning

Reinforcement learning (RL) is one of the branches of machine learning in which an entity known as an agent learns through continuous interaction with its surrounding environment how to choose the best possible actions with the aim of maximizing the amount of cumulative reward. In fact, reinforcement learning is related to solving sequential decision problems. These problems are defined as a system consisting of an operator and an environment. The environment provides information about the conditions of the system under study, which is called the state. The operator observes the state of the states and uses it to perform the action. The environment accepts the action and gives a reward to the agent. When the Agent-Action-State-Reward cycle is completed, a time step has elapsed. This cycle continues until the end of the operation. The function generated by the agent is called a policy, which is responsible for mapping states to tasks. For the allocation of monitoring tasks based on reinforcement learning, an agent (task scheduler) with the experience it gains from allocating monitoring tasks in interaction with the environment through learning, makes better decisions and achieves goals that are difficult to schedule properly directly. By receiving rewards during the implementation of the monitoring system and maximizing the expected rewards over time, this agent becomes an expert in the proper allocation of monitoring tasks to virtual machines, and ultimately leads to a reduction in monitoring costs and an improvement in service quality. Finally, task assignment based on the proposed reinforcement learning solution is done online and the required information is delivered to the users after being collected by the target sensors.

In general, the RL algorithm is an enhanced solution in handling complex decision-making processes. In other words, reinforcement learning is one of the types of machine learning algorithms, with the aim of allowing the agent to learn how to behave in the unknown environment so that at each stage among the set of allowed actions, it performs the appropriate action according to the situation and choose the feedback he gets from the environment (this feedback includes a reward or penalty signal). This way of mapping situations to actions should be such that the reward is maximized in the long run. In fact, the agent finds the way to reach the goal without training and only by gaining experience in his surrounding environment. Unlike most machine learning methods, in this method, the learner has no prior knowledge about the choice of action, but instead has to discover which actions to choose will yield the most rewards over time. These actions not only affect the immediate reward that the agent receives from the environment, but also affect the reward of the next situation and all the rewards after that [49].

B. Stackelberg game theory

In an MCS system, Distribution of tasks is done in an integrated form with considering the benefits of all players. One of common solution in this regard is to use Stackelberg's game theory [50]. Stackelberg game is a two-sided game that is played with complete information. In this game, the leader player selects an action from the Ω_L set, and the follower player, who is aware of the leader player's choice, selects an action from the Ω_F set. In this regard, each player tries to minimize his objective function. Since the leader player takes action first, the action of follower player is dependent on it. The mechanism of modifying the player's behavior is shown in Fig. 2.

Large number and completely non-homogeneous distribution of requested tasks in IoT structure, are the main challenges in this modeling. This causes disturbance in the implementation of Stackelberg's game and failure to reach the equilibrium point. Since Stackelberg's model appeared in 1934, several modified models have been proposed. These methods help to calculate the equilibrium point again in case of different conditions for the leader players and lack of sufficient information about the game platform. One of these variants is the reversed Stackelberg game. The main advantage of the inverse Stackelberg game approach is the possibility of implementing it for situations where users have different responses based on the follower's decision variables.

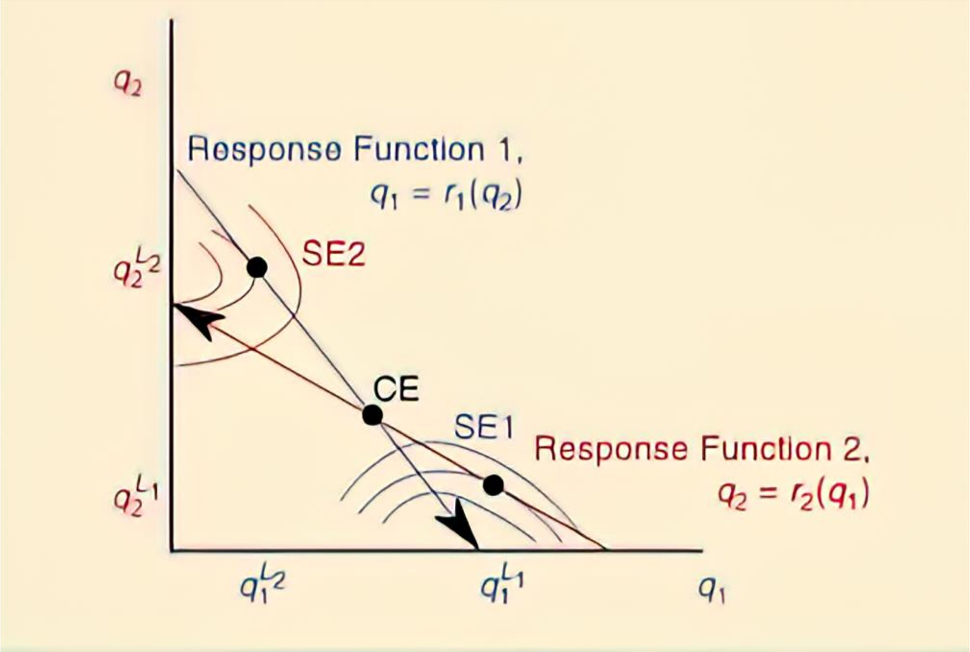

Fig. 2. Schematic of searching for equilibrium point in Stackelberg game [50]

III. PROBLEM FORMULATION

There are two major challenges to this. The first challenge is related to the effective distribution of heterogeneous tasks among a limited number of virtual machines and the second challenge is the limitation of gateway hardware resources. In these circumstances, the goal is to properly distribute tasks among virtual machines in such a way that these resources are used properly and effectively. For this purpose, it is assumed that each virtual machine is capable of performing one task at a time, and the scheduling method is proprietary, and the resource allocated to each task cannot be changed until the end of the execution. If the selected virtual machine is occupied by another task, the current input task must wait in line. The scheduling method in this case is the first input-the first output, and the task that enters the timer earlier in time will receive a resource faster. Upon completion of the task, the instant reward and the next state are returned to the scheduler by the environment, and the values of the initial state (s) of the selected action (a), instant reward (r), and the next state of the environment (s) are stored in the scheduler memory. It turns. In order to achieve an effective scheduling policy, a step has been taken to train the network. In this step, s values are given as input samples to the given network. An error function is used to evaluate the network performance, which predicts the difference in values and calculates the target values and sends the result to the optimizer function. The optimizer function also uses the reduction slope approach to change the network weights in such a way that the error value is reduced. In this way, the network weights are updated at the end of each training phase to achieve an efficient scheduling policy.

Fig. 3 shows the block diagram of presented model. In this regard, the request recognition component is fully aware of the specific needs of different businesses, which may include computing, storage and communications required for computing, login and synchronization conditions, security and privacy needs, quality of service, and more. Service decomposition component Which divides the service into different levels of granularities with different CPU settings. In the next step, task manager, using an effective solution, is responsible for optimizing the resources required for each of the grain analyzers and analysis and mapping for the required processors. Task manager is also responsible for managing work status, scheduling, and allocating resources to requests to assist the scheduling algorithm. The resource recognition component is responsible for managing available resources. Resource cognition component, dynamic optimization, planning strategy and error reporting are other tasks in this section.

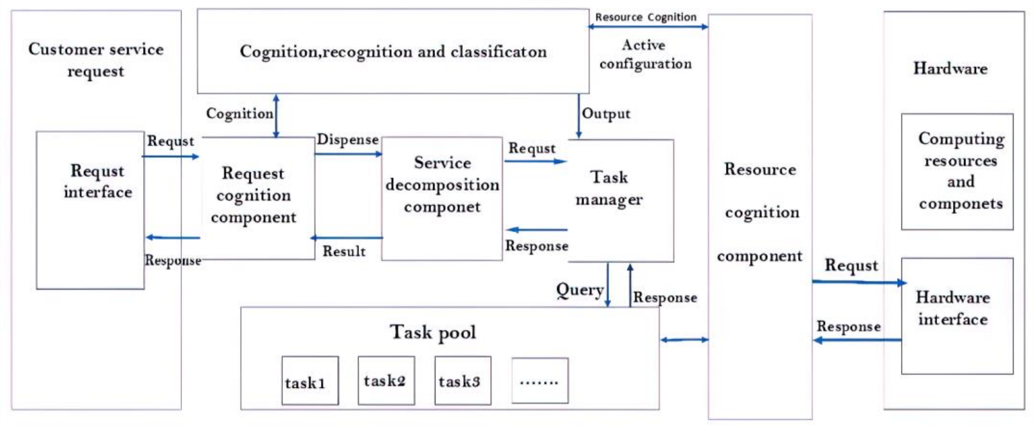

Fig. 3: Schematic of the proposed model structure. At first, the job requested by the users is sent to the request cognition component through request interface. Then, this job is decomposed into different tasks and task manager unit with using the resources identified by resource cognition component and features of each task, allocate proper resources to selected tasks.

Due to limited processing capacity, some fog nodes work together and at the same time connect to cloud nodes at the same time. In this case, the network layering is as follows:

➢The bottom layer contains IoT-enabled devices that send requests to the top layer to be executed.
➢The middle layer of the computing environment is fog. The main components of this layer are smart devices (such as routers, switches) that are located near the end users to receive and process user requests with a local connection, as well as they are able to connect to the cloud.
➢The top layer is the cloud computing layer, which hosts a number of heterogeneous cloud nodes or virtual machines from different cloud service providers. Cloud nodes execute applications received from the fog layer.

At the boundary between the haze layer and user devices, there are several haze devices that act as resource management and task scheduler components called planners. In this structure, computing nodes in the fog and the cloud communicate indirectly with mobile users through scheduling. In fact, all requests are sent to the computing department through the access point to the planner to determine the task. The programmer, who is responsible for analyzing, evaluating and scheduling all tasks to be performed in the fog-cloud system, can communicate with users in a short time due to the proximity to the fog nodes. The steps to perform work in the proposed system are as follows:

1. The programmer receives all user requests.
2. The resources available on the cloud and fog nodes (such as processing capacity and bandwidth), the cost of transfers and processing with the results of the data returned from the nodes are managed by the programmer. In other words, inputs include task scheduler inputs, task graphs, and processor graphs, and outputs include task scheduler outputs, assigning a processor to each task.

3. In this section, a private schedule for incoming workflows is created to decide which part of the request to run on which source.
4. In the final step, a computational node is assigned to each task to execute.

In the proposed method, the rate of monitoring nodes is flexibly determined based on effective metrics such as monitoring cost (energy consumption of IoT nodes) and coverage rate in different areas. The higher the number of monitoring nodes in an area, the better the coverage rate as the sampling rate increases. In fact, the higher the density of nodes, the greater the data redundancy, so in order to reduce energy consumption and eliminate redundant data acquisition, the sampling rate needs to be reduced in this area. In Eq. 1, the number of base monitors is determined for each service in each block. In this regard, $\mathrm{den_k}$indicates the density in block k, $\mathrm{t_s}$ is the number of IoT services, $\mathrm{t_b}$ is the number of blocks (area) in the environment segmentation, and $\mathrm{t_n}$ is the number of base monitors per service in each block.

In the next step, using reinforcement learning, the assignment of monitoring tasks to participating users begins. In the current model, the MCS agent has a function I that receives the current state of the environment as the input s_t from the environment and specifies how the agent views the state of the environment. The current state is expressed by $s_t\in$ S, where S represents the sum of all possible states. In this case, the agent selects an action $a_t\in A(s_t)$ from the set of possible actions in this state and by selecting the action a_t goes to the next state s_{t+1} and immediately receives a $r_{t+1}\in R$ reward from the environment through the R function. In this way, the next state and the expected reward can be predicted according to the current state and a certain action. The function R_t that we are trying to maximize is defined as Eq. 2 [51]. The variable γ is a discount factor that determines the importance of future rewards and helps to converge the value of the function. This factor can take values between zero and one. In the current model, in case s by performing operation a, the probability of transfer to mode S '(here the probability of assigning monitoring task I to user j) is equal to the amount of reward that the agent receives from performing operation in current mode (Eq. 3) . The fit value is obtained according to Eq. 4 in which the parameter τ is a threshold value required for the number of participating nodes.

$$\mathrm{cont(k)} = \mathrm{n_b} * \mathrm{t_b} * \mathrm{t_s} * \left(\frac{\mathrm{den_k}}{\sum_{\mathrm{p\in B}} \mathrm{den_p}}\right) \quad (1)$$

$$\mathrm{R_t} = \mathrm{r_{(t+1)}} + \gamma \mathrm{r_{(t+2)}} + \gamma^2 \mathrm{r_{(t+3)}} + \cdots + \gamma^{\mathrm{T}} \mathrm{r_{(t+T+1)}} = \sum_{\mathrm{k=0}}^{\mathrm{T}} \gamma^{\mathrm{k}} \mathrm{r_{(t+k+1)}} \quad (2)$$

$$\mathrm{r} = \begin{cases} -1 & \text{if coop(j)} < \tau \\ \text{Fitness} & \text{else} \end{cases} \quad (3)$$

$$\mathrm{Cost} = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{E(i,j).T(i,j)} \tag{4}$$

$\mathrm{E(i,j)}$ The amount of energy consumed by node i for data collection in service j is. $\mathrm{T(i,j)}$ Participation or non-participation of node i in service j (as a vector with binary values), Cost is the total cost in the allocation plan. The values of $\mathrm{E(i,j)}$ are calculated by Eq. 5. In this case, $\mathrm{dist(i,dc)}$is the value of the distance of the monitoring node from the data center (the place of aggregation and synchronization of the received data) and $\mathrm{D(i)}$is the maximum monitoring radius for node i. $\mathrm{E_{base}}$is the base amount of power consumed for a service. On the other hand, the coverage factor is obtained as a desirable criterion in data aggregation according to Eq. 6. In this relation (a) is the number of selected points where at least f the monitoring node is present and N (s) is the total number of points considered in the IoT environment for data collection. Therefore, choosing an appropriate task allocation strategy can be defined as the ratio of coverage to cost, the ultimate goal of which is defined as its maximization (Eq. 7). In this way, the two goals of reducing the cost of measurement and increasing the quality are combined to get the best way of assigning tasks to nodes.

$$\mathrm{E(i,j)} = \frac{\mathrm{dist(i,dc)}}{\mathrm{D(i)}} * \mathrm{E_{base}} \tag{5}$$

$$\mathrm{Cover} = \frac{\mathrm{N(a)}}{\mathrm{N(s)}} \tag{6}$$

$$\mathrm{Fitness} = \frac{\mathrm{Cover}}{\mathrm{Cost}} \tag{7}$$

The main challenges in this modeling arise from the fact that in the context of IoT, the type of activities requested by leading actors is both large and completely heterogeneously distributed. This interferes with the implementation of the Stackelberg game and does not achieve equilibrium. Since the emergence of the Stackelberg model in 1934, several modified models of the introductory game have been proposed. This helps to calculate the equilibrium point in case of different conditions for the leading actors and insufficient information from the playing field. One such genre is the inverted Stackelberg game. The main advantage of the inverted reverse Stackelberg. Compared to the usual Stackelberg game is the possibility of implementing it for situations where users have different responses based on the requesting decision variables. In fact, the optimal point can be calculated through Eq. (8) to (10), and thus, without the intervention of the requester, in most cases, this solution is able to induce the desired decision values for all types of users.

$$O.F(1) = \max \quad \varphi_m\left(t_m, p_m\right) \quad s.t. \sum_n p_m^n t_m^n \leq \delta_m \tag{8}$$

$$O.F(2) = \max \quad \psi_n\left(t^n, p^n\right) \quad s.t. \sum_m t_m^n \leq \kappa_n \tag{9}$$

$$\left(U_L^d, U_F^d\right) \in \arg\min \varsigma_L\left(U_L, U_F\right) \qquad \forall \quad \left(U_L, U_F\right) \in \Omega_L \times \Omega_F \tag{10}$$

δ_m : The upper limit of the budget for the implementation of the requested activity

k_n : Maximum time interval for n user

A. Objective Function

For this purpose, it is necessary to have a basic information in each area about the type of activities required, the amount of resources available, the number of active users participating and the available computing capacity. In other words, first in each server, the necessary decisions to accept, process / not process and send the results / raw data are taken locally and then based on the available processing capacity and the status of neighboring neighbors, the distribution of activities is done [37]. In this case, by evaluating the conditions expressed in Eq. 11 to Eq. 13, a suitable approach can be adopted in such a way that if Eq. 11 is established, then server i is left idle. If Eq. 12 is present, server i will refer the unprocessed activities to server j, and if Eq. 13 is established, server i will send the results of the activities to server j. Then the PPO technique is used to achieve the improvement process. In fact, the PPO technique is one of the algorithms that solves the problem of optimizing the policy bound to the possible area with the help of effective and simple innovative methods. In this approach, based on the adaptive KL Penalty, the estimated goal is calculated according to Eq. 16 and then the objective function is obtained according to Eq. 17. Fig. 4 shows the steps of implementing the proposed plan as flowchart.

$$\max\left\{\beta_{ij}^s(t), \gamma_{ij}^s(t)\right\} < \max\left\{\beta_{ji}^s(t), \gamma_{ji}^s(t)\right\} \tag{11}$$

$$\max\left\{\beta_{ij}^s(t)\right\} > \max\left\{\gamma_{ij}^s(t)\right\} \tag{12}$$

$$\max\left\{\beta_{ij}^s(t)\right\} \leq \max\left\{\gamma_{ij}^s(t)\right\} \tag{13}$$

$$\beta_{ij}^s(t) = \varepsilon\left[Q_i^s(t) - Q_j^s(t)\right] - \xi_{ij}(t) \tag{14}$$

$$\gamma_{ij}^s(t) = \varepsilon\left[D_i^s(t) - D_j^s(t)\right] - \xi_{ij}(t) \tag{15}$$

$$J^{CPI}(\theta) = E_t\left[r_t(\theta).A_t\right] \tag{16}$$

$$J^{KLPEN}(\theta) = \max E_t\left[r_t(\theta).A_t - \beta KL\left(\pi_\theta\left(a_t \mid s_t\right) \| \pi_{\theta_{okl}}\left(a_t \mid s_t\right)\right)\right] \tag{17}$$

ε : Step length size

$Q_i^s(t)$: The queue of server activities i in state s in the time step t

$D_i^s(t)$: Server i queue results in s state in time t step

$x_{ij}(t)$: The cost of transferring information from node i to node j in step t

$p_q(a_t|s_t)$: The selected probability of activity a despite the establishment of state s in time step t

θ : Vector policy weights

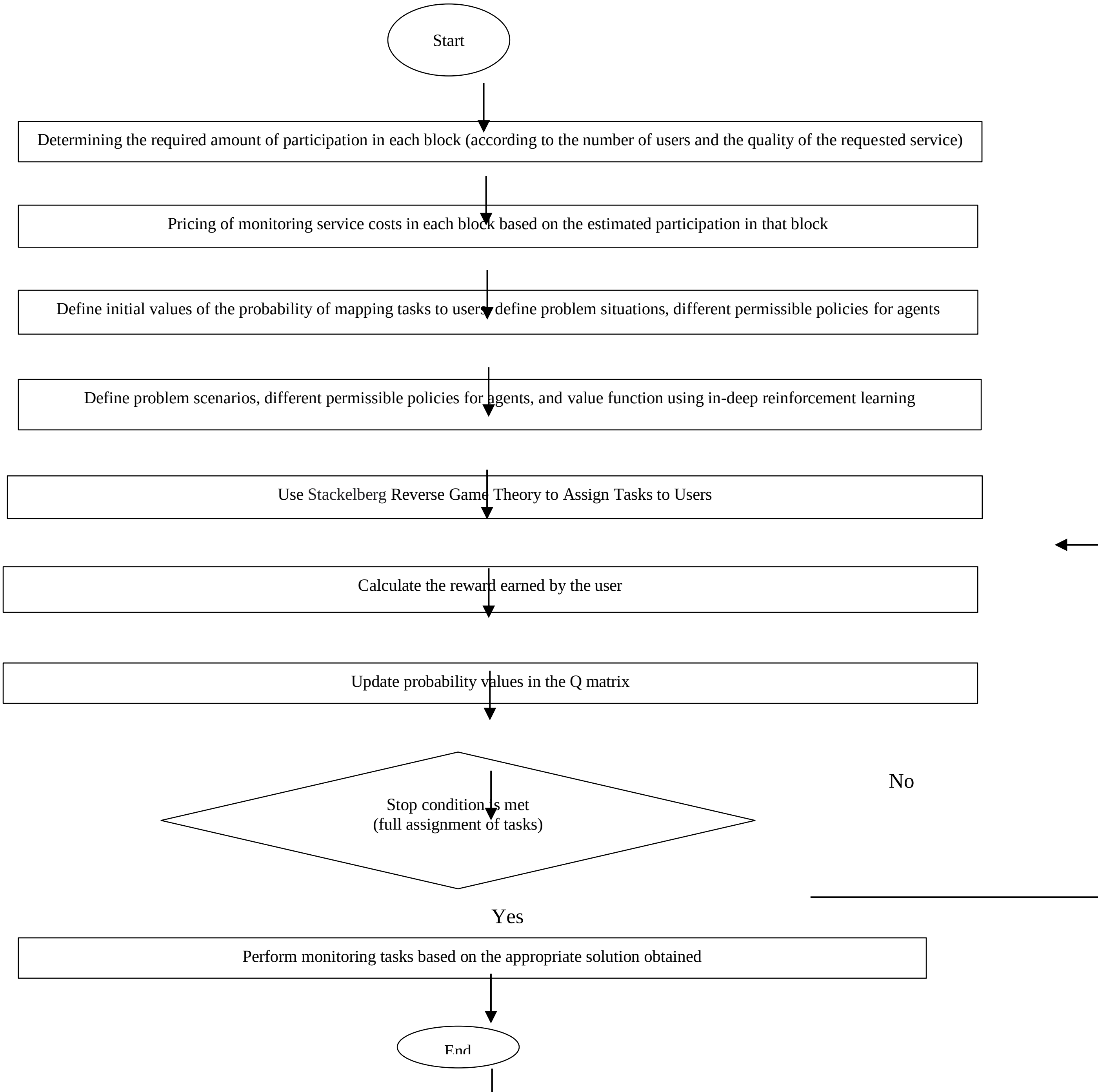
Start
Determining the required amount of participation in each block (according to the number of users and the quality of the requested service)
Pricing of monitoring service costs in each block based on the estimated participation in that block
Define initial values of the probability of mapping tasks to users, define problem situations, different permissible policies for agents
Define problem scenarios, different permissible policies for agents, and value function using in-deep reinforcement learning
Use Stackelberg Reverse Game Theory to Assign Tasks to Users
Calculate the reward earned by the user
Update probability values in the Q matrix
Stop condition is met
(full assignment of tasks)
No
Yes
Perform monitoring tasks based on the appropriate solution obtained
End

Fig. 4: Flowchart of the proposed method

During the training phase, a fixed number of episodes (100) are simulated for each MU to perform a search operation for the current policy in the possible action space and use the data obtained to improve the policy of all activities. In other words, status, action, and reward information is stored for each episode, and these values are used to calculate the cumulative reward for each episode. 1000 repetitions are simulated and then the average reward amount is calculated. The minimum amount of reward, which is equal to the minimum amount of cost, corresponds to full accuracy, and other accuracy values are subsequently calculated. In order to test the effectiveness of the proposed method, 20% of the data set is used, during which the agent follows his trained policy by choosing the action that has the least reward. It is worth noting that in some cases, the values of resource utilization and accuracy criteria were obtained during the simulations.

B. Dataset

In this project, for simulations, Google's cluster data set is used, which contains the required data for the required resources and is available for both activities and virtual machines. The Google branch includes many machines that interact with each other through the high-speed Internet line, and about 40 million tasks are distributed among more than 12,000 machines over a period of 30 days [47-48]. This dataset includes start time, end time, activity characteristic, machine attribute number, CPU rate, maximum CPU rate, amount of allocated memory, maximum memory used, normal memory used, unmapped page cash, total cache, disk input/output time, use of local disk space, maximum disk input/output time, number of cycles for each instruction, type of aggregation performed, memory available per instruction, sampling segmentation and sampled usage CPU. In the proposed approach, the planned policy is taught by repeating Episodes. In each episode, a variable number of MUs (from 10 to 100) and a fixed number of TIs (50,000) are considered. When activities are received, they are planned according to the proposed policy. The episode ends when all activities are completed.

IV. Results Evaluation

In this section, the performance results of the proposed system are examined by considering the variable number of users. In this regard, the number of mobile users (MU) is considered equal to 40. The total delay for all players is shown in Fig. 5. The results show that the higher the number of TIs, the lower the amount of delays, which is due to the more intense competition between the actors. Fig. 6 shows the average delay for each TI when that amount of user changes. The obtained results show the efficiency of the proposed method in comparison with other methods. Fig. 7 shows the accuracy of the training process in comparison with different approaches. The most important point of such results is that increasing the number of MUs leads to a slight decrease in the accuracy obtained. This can be explained by the fact that having more mobile users to choose from may increase the likelihood of incorrectly assigning tasks to some unsuitable virtual machines. At the same time, it is observed that the proposed approach has the highest level of educational accuracy. The result of accuracy analyzing is shown in Fig. 8. This criterion is very important because it can evaluate the effectiveness of the method for data that has not yet been obtained. In these circumstances, the proposed approach has done its job better than other methods with an accuracy between 90.5% to 95.1%. Table 3. shows the degree of confusion of the proposed approach, which has the best score among other reinforcing and deep learning approaches in terms of accuracy of training and testing.

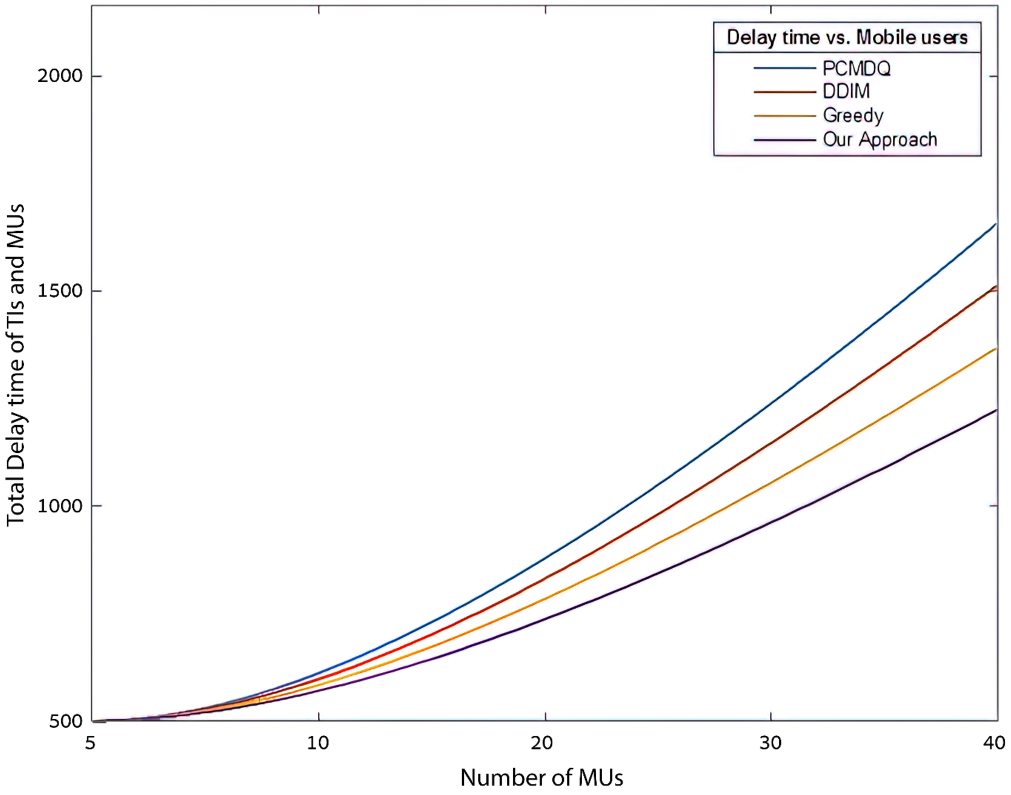

Fig. 5: Evaluation of total delay changes vs. mobile users

In this table, for each mobile user, the rows represent the most optimal virtual machine to which tasks should be assigned. The columns confirm the virtual machines selected by the proposed method that should take over the task. Examining the results, it is clear that out of the total number of tasks that should be scheduled on the first virtual machine, 95.35% (second row and column) is allocated to the machine, and the error rate in this case is equal to 0.044%.

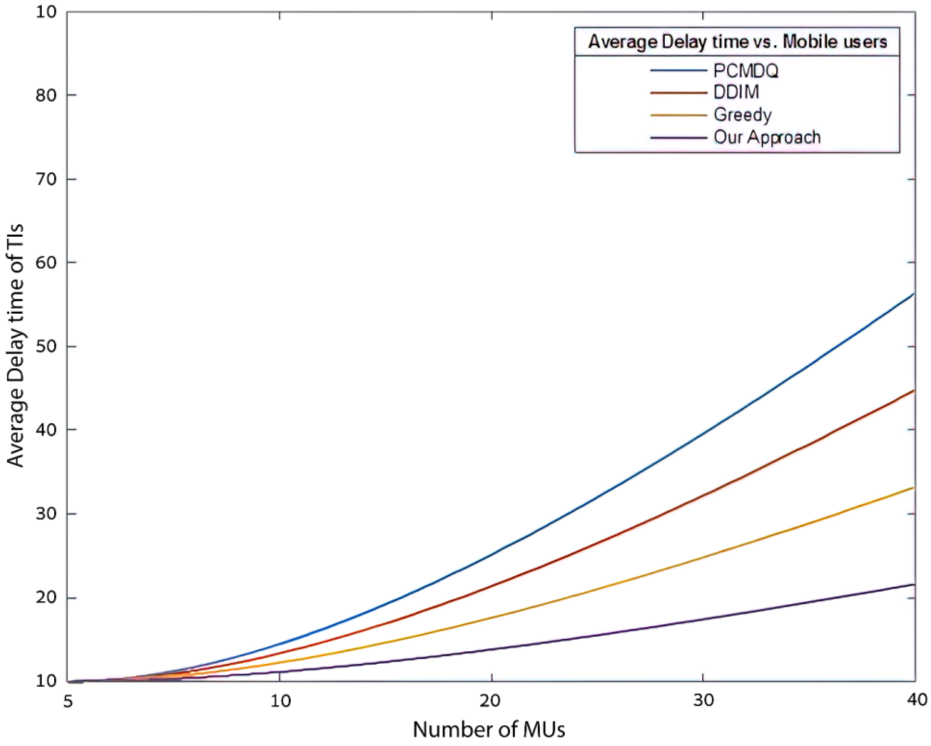

Fig. 6: Evaluation of changes in the average delay of TIs vs. mobile users

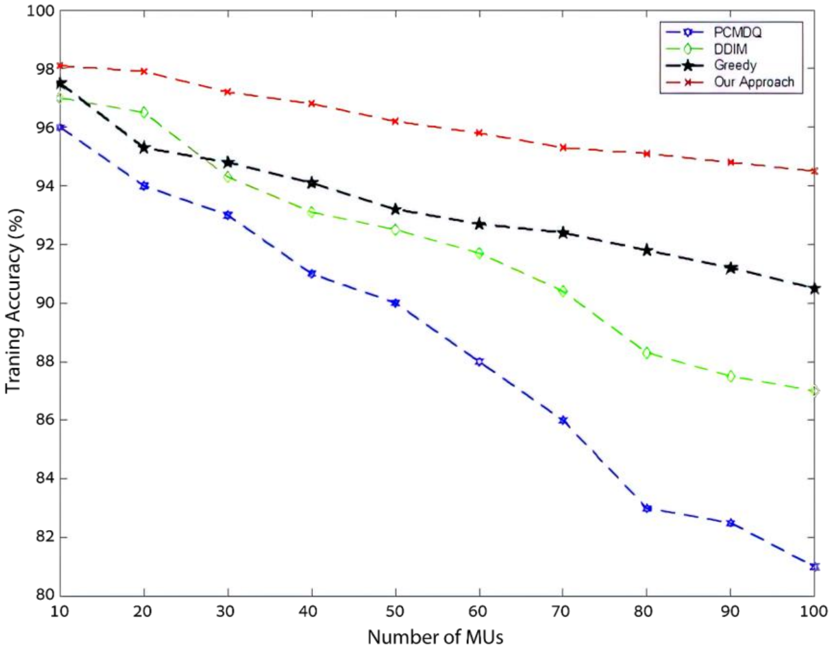

Fig. 7: Accuracy of training in the reinforcement learning process vs. mobile users

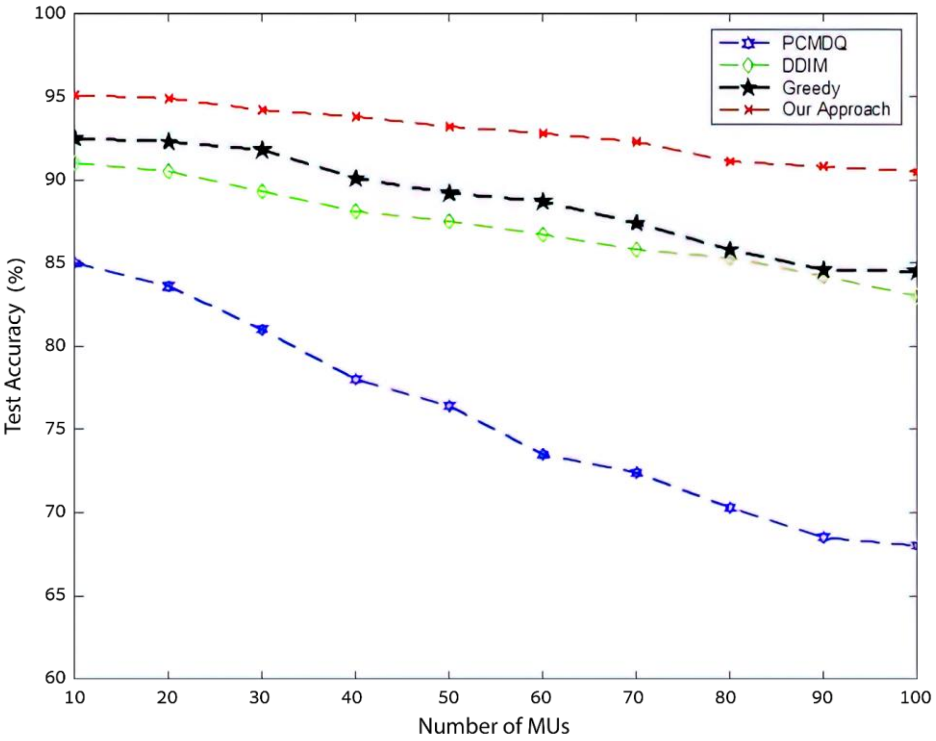

Fig. 8: Test accuracy in the reinforcement learning proces vs. mobile users

Table 3: Assignment matrix to virtual machines

	VM _0	VM_ 1	VM_ 2	VM_ 3	VM _4	VM_ 5	VM _6	VM_ 7	VM_ 8	VM_ 9	Error
VM _0	100 %	0.0%	0.0%	0.0%	0.0 %	0.0%	0.0 %	0.08 %	0.0%	0.0%	0.044 %
VM _1	0.0 %	95.35 %	0.0%	0.0%	0.0 %	0.07 %	0.0 %	0.0%	0.0%	0.0%	0.006 %
VM _2	0.0 %	0.0%	98.23 %	0.0%	0.02 %	0.0%	0.0 %	0.04 %	0.0%	0.01 %	0.207 %
VM _3	0.0 %	0.0%	0.0%	98.23 %	0.0 %	0.0%	0.03 %	0.0%	1.02 %	0.0%	1.370 %
VM _4	0.04 %	0.0%	0.0%	0.0%	100 %	0.0%	0.04 %	0.0%	0.0%	0.0%	0.050 %
VM _5	0.0 %	0.03 %	0.0%	0.0%	0.0 %	97.76 %	0.0 %	0.0%	0.32 %	0.0%	0.063 %

VM _6	0.0 %	0.0%	0.47 %	0.0%	0.0 %	0.0%	100 %	0.0%	0.0%	0.035 %	0.862 %
VM _7	0.0 %	0.0%	0.0%	0.0%	0.05 %	0.0%	0.0 %	99.65 %	0.0%	0.20 %	0.090 %
VM _8	0.0 %	0.0%	0.0%	0.0%	0.0 %	0.0%	0.0 %	0.0%	93.28 %	0.0%	0.002 %
VM _9	0.0 %	0.0%	0.0%	0.67 %	0.0 %	0.0%	0.0 %	0.0%	0.8%	96.01 %	0.560 %

V. Conclusion

A review of the literature on the subject revealed to the author that the allocation of heterogeneous tasks in the context of the IOT has not yet been comprehensively examined. One of the essential requirements of all computing systems, which also leads to improved network performance, is resource management and task scheduling in effective ways. The main purpose of this study is to reduce the average delay in providing services related to Internet of Things applications in the May platform. To this end, an attempt is made to provide a new method for assigning multiple tasks in mobile mass monitoring based on fog computing on the Internet of Things, using inverse Stackelberg game theory or hybrid with the help of fuzzy logic and DRL algorithm. Solving the scheduling problem in question requires an online and adaptable method due to the dynamics of the conditions governing today's computer networks and the difficulty of modeling them. The proposed method will be able to automatically achieve an effective scheduling strategy over time, based on its previous experiences. In the proposed method of this research, we determine the rate of monitoring nodes flexibly and based on effective metrics such as energy consumption of IoT nodes and coverage rate. The main goal is to create maximum coverage with a minimum monitoring cost. In fact, despite the redundancy of data in monitoring, we will achieve higher accuracy in data acquisition. But in applications with low coverage requirements, we will reduce energy consumption by reducing the rate of monitoring nodes. In this way we have implemented a balance between data quality and energy consumption. On the one hand, the participation of nodes should be such as to minimize energy consumption, is tasks should be assigned based on the proximity of the node distance. Of course, the participation rate must reach the minimum required to satisfy the service level agreements. In this case, using deep learning, the movement pattern of people in the community and the extent of their participation in the monitoring process is estimated. After pricing based on participation, the final solution is determined based on maximizing coverage and minimizing users' energy consumption by the reinforcement learning technique.

Abbreviations

MCS: Mobile Crowd Sensing ;IOT: Internet of Things ;DRL: Deep Reinforcement Learning ;MEC: Mobile Edge Computing; MBC: Minimum Bandwidth Codes ;MESS: Minimum Energy Single- sensor task Scheduling DCs: Data Centers ;QOS: Quality Of Service; DDQN: Double Deep Q-Network ;DQN: Deep Q Network ;RL: Reinforcement Learning; PPO: Proximal Policy Optimization; MU: Mobile Users; EBRP: Energy-Efficient and Buffer-Aware Routing Protocol.

Acknowledgements

The authors are thankful to anonymous reviewers for their valuable comments and suggestions that helped improving the quality of the paper.

Authors' contributions

All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Datasets that have been used for experiments in this paper are available at:

https://github.com/google/cluster-data.

Competing interests

The authors have no competing interests.

Author details

[1] Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran.

Ref:

1. Dave, E. "How the next evolution of the internet is changing everything." The Internet of Things (2011).
2. Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "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. 2012.
3. Javadzadeh, Ghazaleh, and Amir Masoud Rahmani. "Fog computing applications in smart cities: A systematic survey." Wireless Networks 26, no. 2 (2020): 1433-1457.

4. Ni, Jianbing, Kuan Zhang, Yong Yu, Xiaodong Lin, and Xuemin Sherman Shen. "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing." IEEE Transactions on Dependable and Secure Computing 17, no. 3 (2018): 581-594.
5. Nie, Jiangtian, Jun Luo, Zehui Xiong, Dusit Niyato, and Ping Wang. "A stackelberg game approach toward socially-aware incentive mechanisms for mobile crowdsensing." IEEE Transactions on Wireless Communications 18, no. 1 (2018): 724-738.
6. Capponi, Andrea, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities." IEEE communications surveys & tutorials 21, no. 3 (2019): 2419-2465.
7. Espadas, Javier, Arturo Molina, Guillermo Jiménez, Martín Molina, Raúl Ramírez, and David Concha. "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures." Future Generation Computer Systems 29, no. 1 (2013): 273-286.
8. Apat, Hemant Kumar, Prasenjit Maiti, and Punyaban Patel. "Review on QoS Aware Resource Management in Fog Computing Environment." In 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), pp. 1-6. IEEE, 2020.
9. Wu, Maoqiang, Dongdong Ye, Shensheng Tang, and Rong Yu. "Collaborative vehicle sensing in bus networks: A Stackelberg game approach." In 2016 IEEE/CIC International Conference on Communications in China (ICCC), pp. 1-6. IEEE, 2016.
10. Zhang, Huaqing, Yong Xiao, Shengrong Bu, Dusit Niyato, F. Richard Yu, and Zhu Han. "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching." IEEE Internet of Things Journal 4, no. 5 (2017): 1204-1215.
11. Espadas, Javier, Arturo Molina, Guillermo Jiménez, Martín Molina, Raúl Ramírez, and David Concha. "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures." Future Generation Computer Systems 29, no. 1 (2013): 273-286.
12. Wang, William Yu Chung, Ammar Rashid, and Huan-Ming Chuang. "Toward the trend of cloud computing." Journal of Electronic Commerce Research 12, no. 4 (2011): 238.
13. Shi, Zhuan, He Huang, Yu-E. Sun, Xiaocan Wu, Fanzhang Li, and Miaomiao Tian. "An efficient task assignment mechanism for crowdsensing systems." In International Conference on Cloud Computing and Security, pp. 14-24. Springer, Cham, 2016.
14. Zhu, Weiping, Wenzhong Guo, Zhiyong Yu, and Haoyi Xiong. "Multitask allocation to heterogeneous participants in mobile crowd sensing." Wireless communications and mobile computing 2018 (2018).
15. Lu, An-qi, and Jing-hua Zhu. "Worker recruitment with cost and time constraints in mobile crowd sensing." Future Generation Computer Systems 112 (2020): 819-831.

16. Huang, Liang, Xu Feng, Cheng Zhang, Liping Qian, and Yuan Wu. "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing." Digital Communications and Networks 5, no. 1 (2019): 10-17.
17. Graesser, Laura, and Wah Loon Keng. Foundations of deep reinforcement learning: theory and practice in Python. Addison-Wesley Professional, 2019.
18. Deng, Ruilong, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption." IEEE internet of things journal 3, no. 6 (2016): 1171-1181.
19. Li, Songze, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. "Coding for distributed fog computing." IEEE Communications Magazine 55, no. 4 (2017): 34-40.
20. Shen, Liquan, Zhaoyang Zhang, and Ping An. "Fast CU size decision and mode decision algorithm for HEVC intra coding." IEEE Transactions on Consumer Electronics 59, no. 1 (2013): 207-213.
21. Masip-Bruin, Xavi, Eva Marín-Tordera, Ghazal Tashakor, Admela Jukan, and Guang-Jie Ren. "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems." IEEE Wireless Communications 23, no. 5 (2016): 120-128.
22. Kumar, Neetesh, Syed Shameerur Rahman, and Navin Dhakad. "Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system." IEEE Transactions on Intelligent Transportation Systems (2020).
23. Chen, Miaojiang, Tian Wang, Kaoru Ota, Mianxiong Dong, Ming Zhao, and Anfeng Liu. "Intelligent resource allocation management for vehicles network: An A3C learning approach." Computer Communications 151 (2020): 485-494.
24. Lu, An-qi, and Jing-hua Zhu. "Worker recruitment with cost and time constraints in mobile crowd sensing." Future Generation Computer Systems 112 (2020): 819-831.
25. Wang, Liang, Zhiwen Yu, Bin Guo, Fei Yi, and Fei Xiong. "Mobile crowd sensing task optimal allocation: A mobility pattern matching perspective." Frontiers of Computer Science12, no. 2 (2018): 231-244.
26. Shi, Zhuan, He Huang, Yu-E. Sun, Xiaocan Wu, Fanzhang Li, and Miaomiao Tian. "An efficient task assignment mechanism for crowdsensing systems." In International Conference on Cloud Computing and Security, pp. 14-24. Springer, Cham, 2016.
27. Lu, An-qi, and Jing-hua Zhu. "Worker recruitment with cost and time constraints in mobile crowd sensing." Future Generation Computer Systems 112 (2020): 819-831.
28. Zhu, Weiping, Wenzhong Guo, Zhiyong Yu, and Haoyi Xiong. "Multitask allocation to heterogeneous participants in mobile crowd sensing." Wireless communications and mobile computing 2018 (2018).
29. D. Yang, G. Xue, X. Fang, and et al., “Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing,” in ACM MobiCom, 2012, pp. 173–184.

30. D. Yang, G. Xue, X. Fang, and J. Tang, “Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones,” IEEE/ACM Trans. Net., vol. 24, no. 3, pp. 1732–1744, 2016.
31. S. Maharjan, Y. Zhang, and S. Gjessing, “Optimal in- centive design for cloud-enabled multimedia crowd- sourcing,” IEEE Trans. Multimedia, vol. 18, no. 12, pp. 2470–2481, 2016.
32. M. H. Cheung, F. Hou, and J. Huang, “Delay-sensitive mobile crowdsensing: Algorithm design and economic- s,” IEEE Trans. Mob. Comput., 2018.
33. Y. Chen, B. Li, and Q. Zhang, “Incentivizing crowd- sourcing systems with network effects,” in IEEE INFO- COM, 2016, pp. 1–9.
34. Y. C. Ho, P. Luh, and R. Muralidharan, “Information structure, Stackel- berg games, and incentive controllability,” IEEE Trans. Automat. Control, vol. 26, no. 2, pp. 454–460, 1981.
35. Y. C. Ho, P. Luh, and G. Olsder, “A control-theoretic view on incen- tives,” Automatica, vol. 18, no. 2, pp. 167–179, 1982.
36. D. H. Cansever and T. Bas ̧ar, “On stochastic incentive control prob- lems with partial dynamic information,” Syst. Control Lett., vol. 6, no. 1, pp. 69–75, 1985.
37. H. Ehtamo and R. P. Ha ̈ma ̈la ̈inen, “Incentive strategies and equi- libria for dynamic games with delayed information,” J. Optim. Theory Appl., vol. 63, no. 3, pp. 355–369, 1989.
38. T. Valle ́e, T. Ch. Deissenberg, and Q. Bas ̧ar, “Optimal open loop cheating in dynamic reversed linear. Stackelberg games,” Ann. Oper. Res., vol. 8, no. 0, pp. 217–232, Jan. 1999.
39. H. Shen and T. Bas ̧ar, “Incentive-based pricing for network games with complete and incomplete information,” in Advances in Dynamic Game Theory (Annals of the International Society of Dynamic Games, vol. 9), S. Jørgensen, M. Quincampoix, and T.L. Vincent, Eds. New York, NY: Birkha ̈user Boston, 2007, pp. 431–458.
40. P. B. Luh, Y. C. Ho, and R. Muralidharan, “Load adaptive pricing: An emerging tool for electric utilities,” IEEE Trans. Automat. Control, vol. 27, no. 2, pp. 320–329, 1982.
41. T. Roughgarden, “Stackelberg scheduling strategies,” SIAM J. Comput., vol. 33, no. 2, pp. 332–350, 2004.
42. Sazdar, Amir Mehdi and Ghorashi, Seyed Ali and Khonsari, Ahmad, 1398, Ensuring the location of participants in the Internet of Things network with the ability to tolerate delays for mass measurement applications, National Informatics Conference of Iran, Tehran. Https://civilica.com / doc / 1002101
43. Wang, Jiangtao, Yasha Wang, Daqing Zhang, and Sumi Helal. "Energy saving techniques in mobile crowd sensing: Current state and future opportunities." IEEE Communications Magazine 56, no. 5 (2018): 164-169.
44. Tomasoni, Mattia, Andrea Capponi, Claudio Fiandrino, Dzmitry Kliazovich, Fabrizio Granelli, and Pascal Bouvry. "Profiling energy efficiency of mobile crowdsensing data collection frameworks for smart city applications." In 2018 6th IEEE International

Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 1-8. IEEE, 2018.

45. Wang, Jing, Jian Tang, Guoliang Xue, and Dejun Yang. "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems." Computer Networks 115 (2017): 100-109.
46. Schrijver, Alexander. Theory of linear and integer programming. John Wiley & Sons, 1998.
47. Xiao, Fu, Zhifei Jiang, Xiaohui Xie, Lijuan Sun, and Ruchuan Wang. "An energy-efficient data transmission protocol for mobile crowd sensing." Peer-to-Peer Networking and Applications 10, no. 3 (2017): 510-518.
48. Han, Yang, Yanmin Zhu, and Jiadi Yu. "A distributed utility-maximizing algorithm for data collection in mobile crowd sensing." In Proc. IEEE Glob. Commun. Conf.(GLOBECOM), pp. 277-282. 2014.
49. L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, “Incentive mechanisms for smartphone collaboration in data acquisition and distributed com- puting,” in IEEE INFOCOM, 2012, pp. 1701–1709.
50. Groot, Noortje, Georges Zaccour, and Bart De Schutter. "Hierarchical game theory for system optimal control: Applications of reverse stackelberg games in regulating marketing channels and traffic routing." IEEE Control Systems Magazine 37, no. 2 (2017): 129-152.
51. Yu, Yan, Qian Shi, and Hak-Ke`ung Lam. "Fuzzy sliding mode control of a continuum manipulator." In 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2057-2062. IEEE, 2018.