Int. J. Nonlinear Anal. Appl. 15 (2024) 4, 251–264
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.29551.4188

Heterogeneous task allocation in mobile crowd sensing using a modified approximate policy approach

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

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

[b]*Department of Electrical Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran*

(Communicated by Seyed Hossein Siadati)

Abstract

Currently, cloud computing provides the necessary infrastructure and software services to provide the requested services needed by users on the Internet. Due to the spectacular growth of cloud computing, the number of users and the number of demands are increasing rapidly, which creates a high workload on servers and computing resources. In this situation, for the optimal use of resources, the need for an efficient and effective management approach is fully felt. For this purpose, game theory has been used. The game structure is designed in such a way that the leader (leader) owns a large number of resources and plans the allocation of resources based on the request received from mobile users. The goal from the leader's point of view is to minimize the cost of using the resources located in the fog nodes, on the other hand, the goal considered from the mobile users' point of view is to minimize the cost of responding and transmitting the message to the desired fog node. For this purpose, the entire region is divided into regions and a fog node is considered for each region. The main goal of this research is to reduce the average delay in the provision of services related to Internet of Things applications in cloud computing platform. For this purpose, an attempt is made to provide a new method for allocating multiple tasks in mobile crowd sensing based on fog computing in Internet of Things using the inverse Stackelberg game theory with the help of fuzzy logic and deep reinforcement learning algorithm.

Keywords: Mobile crowd sensing, Proximal Policy, Fuzzy logic, Task Allocation, Internet of Things, Deep Reinforcement Learning
2020 MSC: 03B52, 68T07

1 Introduction

With the rapid expansion of the sensing, computing and communication capabilities of mobile devices, mobile crowd sensing (MCS), in which a large number of participants equipped with mobile devices collect the activities of an area in real time, has emerged as a new measurement paradigm and has attracted much attention. has attracted in recent years, several platforms have taken advantage of MCS, such as Wiz, Million Factors, Medusa, and Prism, and have been widely used in various fields such as intelligent transportation, health care, environmental monitoring, and urban public management.

*Corresponding author
Email addresses: zohrehvahedi@mshdiau.ac.ir (Zohreh Vahedi), mahdavi@mshdiau.ac.ir (Seyyed Javad Seyyed Mahdavi Chabok), gveisi@gmail.com (Gelareh Veisi)

Received: November 2022 *Accepted:* January 2023

In the MCS structure, the mobile device carried by the participants is used as the basic measurement unit. First, the system structure must assign tasks to appropriate participants. Then the performed tasks and measured data are uploaded to the system by the participants with their mobile devices. Finally, the obtained data is evaluated and processed to perform tasks. In other words, mobile crowd sensing is an emerging model in which citizens share data generated on their mobile devices and developers use this data to extract collective information and provide people-oriented services.

In MCS, task allocation strategy is a fundamental issue and its study has gone through several stages of development. The first research on task allocation focused on single-task allocation. However, with more use of MCS applications, the generated tasks gradually increased. Due to limited sensing resources (e.g., number of mobile devices, types of sensors, and energy resources), multitasking under limited resources has been proposed and has become a research focus. In addition, since the sensing activities of the participants are affected by time and space, the study of spatio-temporal features for multitasking allocation is also of great importance. In this situation, various sensor devices generate a large amount of data and consume a lot of resources, including bandwidth, memory capacity, and a huge amount of energy. As a result, the quality of service required for applications may not be achieved. Therefore, service quality is one of the important challenges in this field. Even with significant advances in technology, mobile devices are still not capable of running software with high processing power (along with suitable quality for users), such as audio and image processing, data mining in social networks.

Another important challenge is the energy consumption to complete the MCS tasks, which is caused by collecting sensor data, checking the data locally, and transferring the data to special servers. If one of the tasks requires high energy monitoring, it leads to a sharp decrease in participants' willingness to participate in the MCS process. In this regard, in order to solve the challenge of limited resources of mobile devices (limited battery energy and low processor power), it is proposed to transfer parts of software processing from the mobile device to the cloud. In fact, mobile cloud computing increases the ability of mobile devices to run software. However, remote cloud resources are not always available and the cost of using them is high, and on the other hand, the use of local cloud space limits the portability of mobile device users and requires pre-arranged infrastructure.

In recent years, several comprehensive reviews have been presented in the discussion of handover in mobile cloud computing. In the reference, they have designed and implemented a system that enables the client's mobile device to leave its calculations to the mobile devices in the surrounding environment, which alternately exchange information with each other. Assuming the tasks are the same, the authors have presented two versions of the greedy scheduling algorithm called time-optimizing serendipity and energy-aware serendipity. In the time optimization algorithm, for each task, the destination node is the implementation is selected in the shortest time. But the goal of the energy-aware algorithm is to increase the lifetime of the network and provide the time conditions considered for the entire application. This algorithm selects a node with the lowest amount of energy for each task, so that the handover time condition is also fulfilled. Although this algorithm provides a good estimate, it does not necessarily minimize the average energy consumption. Also, although this article has examined the effect of the two criteria of execution time and the amount of energy consumed in allocating tasks to nearby mobile devices, it does not optimize the two criteria of time and energy together in decision making [1].

The cloud environment provides access to many resources based on users' needs, and users can use them in a flexible and scalable way. The number of users who use cloud services is increasing day by day, therefore, according to the dynamics of users' requests, the existence of an efficient and effective scheduling algorithm responsible for the appropriate allocation of tasks to available resources and increasing system performance is very important. it is needed. Scheduling refers to the mapping of tasks to specific resources in order to improve system performance and service quality, which takes into account different parameters such as minimum completion time, support of service level agreements, execution cost, establishing security and improving reliability. In most scheduling algorithms, different criteria such as response time, task completion time, energy, load balance, cost, reliability are considered, which have been able to solve some of them to some extent, but due to the limited access to information about the tasks, it is necessary They do not lead to improved response time and task completion time along with increased productivity rates in the long run.

2 Game theory

In resource management issues, it is necessary to respect both the interests of subscribers and the interests of service providers. In fact, the profitability of both sides should be considered. In such a situation, sometimes the profitability of one side is in contradiction with the profitability of another sector, and therefore, approaches can be chosen to solve such challenges that are based on game theory. In classical game theory, it is necessary to have complete information about the sides of the game. This cannot be realized during the implementation of mobile mass surveillance, and therefore, the aforementioned game algorithms cannot be used. In order to overcome such problems, in this project, the inverse Stackelberg game algorithm is used, in which there is no need to define all the characteristics of the actors.

3 Q learning algorithm

How can the agent learn the optimal policy (π^*) in an arbitrary environment? This is not possible through direct learning because there are no training examples in the form of $< s, a >$ pairs and only reward values in the form $r(s_i, a_i)$ are available. In this situation, learning the numerical estimation function on states is easier. After finding the numerical estimation function, the optimal policy can be obtained. In this situation, one of the estimated functions is the V^* function. It is necessary for the agent to prefer state s_1 to state s_2 whenever the conditions stated in relation (3.1) are true. Of course, the policy chooses between actions and cannot be used between states. However, V^* can also be used to choose between actions [2].

The optimal operation is achieved when the relationship (3.2) is established. In this regard, $\delta(s, a)$ is the symbol of the next state for the action of a in state s. Therefore, the agent can determine the optimal policy by using V^* and sufficient information about the function of instantaneous rewards r and the function of final states δ, so when the agent is aware of the two functions δ and r that the environment uses for changes, it can determine the optimal action for state s using equation (3.2). Unfortunately, by learning V^*, the optimal policy can be determined only when the agent is aware of the two functions δ and r. Such an assumption is equivalent to knowing the momentary outcomes (both momentary rewards and final states) for every ordered state-action pair. This assumption is similar to having a complete domain theory in explanatory learning. In many practical problems, such as robot control, it is impossible for the operator or programmer to accurately predict the result of each movement in each state.

$$V^*(s_1) > V^*(s_2) \tag{3.1}$$

$$\pi^* = \arg\max[r(s, a) + \gamma \cdot V^*(\delta(s, a))] \tag{3.2}$$

For example, suppose that a robot shovel-shaped arm wants to do some digging, and the final state is the location of the soil particles after digging. In such a situation, both δ and r are unknown and unfortunately learning V^* will not be useful because the agent cannot fulfill the relationship (3.2). So, in this situation, what estimation function should the agent use for this challenge? The estimation function Q, which will be defined in the next section, provides an answer to this problem.

Assume that the estimation function $Q(s, a)$ is defined in such a way that its value is always the maximum possible value of the discounted cumulative reward function for each state s and each action a in the first step. In other words, the value of the expression Q is always the sum of the value of the momentary reward of action a and the reward of the optimal policy after that (with a discount of γ) (Relation (3.3)). The term $Q(s, a)$ is exactly the same quantity that is maximized in relation (3.2) for choosing ba in the state of s. Therefore, by rewriting, we will get the relation (3.4). This relationship shows that if the agent learns function Q instead of V^*, he can find optimal actions without needing two functions r and δ. In this case, it is sufficient to choose an action that maximizes $Q(s, a)$ by considering different states for a in s states [3].

At first, it may seem strange that by maximizing a local quantity such as Q, the maximum overall reward can be reached by repeating the actions, and this means that the agent can take the optimal action without even considering what will happen after this action. Find the feature of learning Q is that the function Q is defined in such a way that it contains all the necessary information about the cumulative discount function in the future by choosing action a in state s.

For a better understanding, Figure 1 shows the values of the Q function for each state in the defined square environment. Note that the value of Q for each ordered state-action pair shows the sum of the value of the momentary reward and the value of the function V^* with a discount of γ. The optimal policy is also proportional to the values of Q for different states. To learn Q, the set of rewards r over time can be used by iterative estimation. In fact, according to the equation (3.5), the recursive definition of Q can be used as a basis for loop algorithms to estimate Q. In the following, to explain the algorithm, the symbol $\hat{Q}$ is used to represent the estimation or the learning hypothesis, and the symbol Q is used for the main function.

$$Q(s, a) = r(s, a) + \gamma.V^*(\delta(s, a)) \tag{3.3}$$

$$\pi^*(s) = \arg\max Q(s,a) \tag{3.4}$$

In this learning algorithm, the hypothesis $\hat{Q}$ is introduced with a large table in which a number is assigned for each state-action pair. The learner hypothesis is actually an estimate of the original Q. This table is filled with random values in the first step and then in each step the agent observes the state s and executes an action such as a, then the reward from the action is $r = r(s,a)$ and the final state ($s' = \delta$(observes s,a). In the following, the value of $\hat{Q}(s,a)$ changes in the form of equation (3.6) in such a case. In this situation, the agent uses the value of $\hat{Q}$ in s' to estimate $\hat{Q}$ in the previous step. Although this relation is dependent on the previous estimate of $\hat{Q}$, but it can be said that this state is derived from relation (3.5). Note that although relation (3.5) is based on two functions $r(s,a)$ and $\delta(s,a)$, there is no need to know these two relations to implement relation (3.6). Instead, the agent performs the mentioned action and observes the new state s' and the reward r, so it can be said that in this relationship the two values s' and r model these two functions. Using this algorithm, the factor $\hat{Q}$ tends to the real value of Q. Since it was initially assumed, the system follows the Markov decision model, so the reward function is bounded and the actions are selected so that each pair of states and actions are observed several times. In the following, the steps of the Q learning algorithm are presented, assuming that rewards and actions are deterministic and that the discount constant is in a range between zero and one [4].

$$Q(s,a) = r(s,a) + \gamma.\max Q(\delta(s,a),\acute{a}) \tag{3.5}$$

$$\hat{Q}(s,a) \longleftarrow r + \gamma.\max \hat{Q}(\acute{s},\acute{a}) \tag{3.6}$$

3.1 Q learning algorithm steps

- For each pair of s and a, set the initial value of $\hat{Q}(s,a)$ equal to zero.
- View the current status of s.
- Continue the following loop until infinity:
- Choose an action like a and do it.
- Get the instant reward amount r.
- See the new mode $\acute{s}$.
- Update the value of the function with the following relation.

$$\hat{Q}(s,a) = r + \gamma.\max \hat{Q}(\acute{s},\acute{a})$$

$$s \longleftarrow \acute{s}$$

4 The importance of research

Mobile crowd sensing requires a large number of participants in the process of monitoring the surrounding environment using sensor devices such as smartphones. This is an emerging phenomenon that is the result of the combination of the distributed pattern of the Internet of Things and crowd-based designs. Mobile monitoring applications are deployed on participating nodes such as mobile phones and personal devices that are capable of monitoring the physical environment and providing monitored data to that server application. In such a large structure, monitoring devices continuously generate a large amount of data, which consumes a lot of resources, including bandwidth and energy. While monitoring devices have limited energy resources. When there are network traffic constraints and high workload due to resource constraints, the quality of collected data may be sacrificed.

Therefore, resource limitation is a key challenge in mobile mass surveillance. For example, images collected from a critical area can play a vital role in reducing the effects of that critical phenomenon. These collected images may not be able to be uploaded in a timely manner due to the lack of sufficient bandwidth, which may result in irreparable damages. Another example is when most monitoring applications require the location information of the monitored event, while the global navigation system as a location-based guidance system is a large energy consumer. This is due to the high quality of the collected data. By studying these works, it is worth noting that resource limitations make the need for user participation and high-scale compatibility necessary for target applications [5].

In reviewing the literature on the subject, the writer found that the allocation of heterogeneous functions in the Internet of Things (IoT) platform has not yet been comprehensively investigated. One of the essential requirements of all computer systems, which leads to the improvement of network performance, is resource management and task scheduling in an effective way. The main goal of this research is to reduce the average delay in the provision of services related to Internet of Things applications in cloud computing. For this purpose, an attempt is made to present a new method for assigning multiple tasks in mobile crowd sensing based on fog computing in the Internet of Things using the inverse or hybrid Stackelberg game theory with the help of fuzzy logic and deep reinforcement learning algorithm. Solving the discussed scheduling problem requires an online and adaptable method due to the dynamic conditions governing today's computer networks and the difficulty of their modeling. The proposed method will be able to achieve an effective scheduling strategy over time automatically and based on its previous experience.

5 Research question

How can we plan optimization for mobile task allocation based on Stackelberg game theory?

6 Research objectives and innovation

In reviewing the literature on the subject, it became clear to the author that improving the quality of service and minimizing energy consumption in MCS systems are important challenges in this field. In this situation, the fog can be used to transfer data wirelessly between objects defined in the network more easily. In addition, one of the essential requirements of all computer systems that leads to the improvement of network performance is resource management and task scheduling in an effective way. The main goal of this research is to reduce the average energy consumption of providing services related to Internet of Things applications in the fog environment [6].

For this purpose, an attempt is made to provide a new method for allocating multiple tasks in mobile crowd sensing based on fog computing in the Internet of Things by using the deep reinforcement learning (DRL) method. Solving the discussed scheduling problem requires an online and adaptable method due to the dynamic conditions governing today's computer networks and the difficulty of modeling them. Therefore, using investigative and law-based methods for this purpose is not very practical. The proposed method will be able to achieve an effective scheduling strategy over time automatically and based on its previous experience. In this situation, two of the most important implementations in this project are described below:

- Improvement of agent training: In general, one of the major challenges that arise when training agents with policy-based gradient algorithms is the possibility of losing the target due to the weak and sudden performance of agents. Recovering from the situation in this case is very difficult because the agent starts weak traces and uses it to train policies. In such a situation, the algorithms with policies are no longer able to use the data and these examples are not efficient for them. One of the ways to overcome this challenge is to use proximal policy optimization (PPO) algorithms. The main idea of introducing PPO is to introduce an alternative objective to avoid the weakening of efficiency by ensuring uniformity of policy improvement. This objective function definition model also has the advantage that unused data can be reused in the policy design for the training process.
- Use of fuzzy logic: In the real world, many times there are situations that make it very difficult to accurately determine the state of the states. In these situations, fuzzy logic can be used in order to design a flexible approach to make appropriate decisions. In other words, uncertainties can be modeled for any situation by applying this logic. In other words, it is determined by reviewing the latest articles in this field Deed that the allocation of tasks with the help of mobile crowd sensing in the fog space has not been fully evaluated yet. PPO algorithms are easy to implement, require little computing volume, and do not depend on the selection of a specific parameter. In this situation, the fuzzy algorithm can be used to prioritize tasks before applying them to the allocation system in order to improve the overall performance of the system. The two major challenges considered in this project include proper and efficient distribution of heterogeneous tasks and utilization of the limited capacity of resources defined in fog nodes. In fact, the main goal is the optimal utilization of the available capacities with proper allocation of duties. In this situation, the following assumptions are considered:

– At any time, only one task can be executed by each unit of the virtual machine.
– The scheduling method is such that no change can be made in the allocated resource until the completion of the considered activity.
– If the resource is assigned to a task, it needs to be queued when the next task arrives.
– The mechanism used in queuing is first-in-first-out. The sooner a task is entered into the system, the more likely it is to receive a free resource.
– The momentary reward and the next state after the completion of the task are calculated and stored in the memory of the scheduler system along with the values of the initial state.

In the training phase, the state values are applied as input to the system and to evaluate the performance quality, an error function is used to report the difference between the obtained values and the reference values to the optimizer function. With the help of the steepest decreasing slope approach, the weights used in the network are corrected and updated. Figure 1 shows the system performance model. The request recognition unit is fully aware of the system's needs, which may include things such as the amount of necessary calculations, the amount of storage capacity, the amount of available bandwidth, the rules related to the synchronization between functions, the conditions for creating security and protecting the privacy of users. The request parsing service component divides the received service into multiple levels using granular parsing with various settings created by the processor [7]. In the next step, task management, using an effective solution, is responsible for optimizing the required resources of each of the analyzed values and then mapping them to the required processors. It is also responsible for managing the tasks related to the indicators that determine the work status, explaining the process of the planning sequence and allocating resources in the necessary time to help the scheduling algorithm.

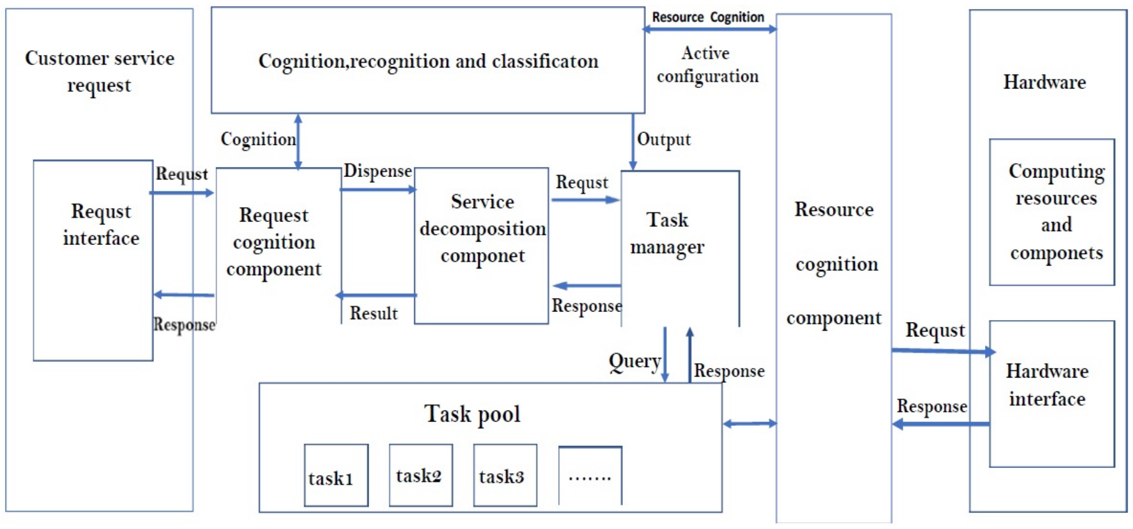

Figure 1: The system performance model

The resource recognition component is responsible for the management of existing and available resources. Also, monitoring the performance of new imported resources, continuous optimization, determining the planning strategy and notifying all types of errors are other duties of this part.

The layering used in this project is shown in Figure 2. In this situation, the functions of the layers are as follows:

- The lowest layer consists of devices equipped with Internet of Things technology and send requests to the higher layer to be executed. Measurement operations by sensors and production of raw data are done in this layer.
- The second layer is the communication layer. Smart devices such as mobile phones and smart watches are placed in this layer, which are responsible for sending information to higher layers.
- The third layer acts as a computing layer and consists of a number of cloud spaces and virtual machines for information processing. Communication between cloud and fog is realized through this layer.
- Application software is placed in the highest layer. In fact, in this layer, the results obtained from the information analysis and processing are used and implemented in a practical way.

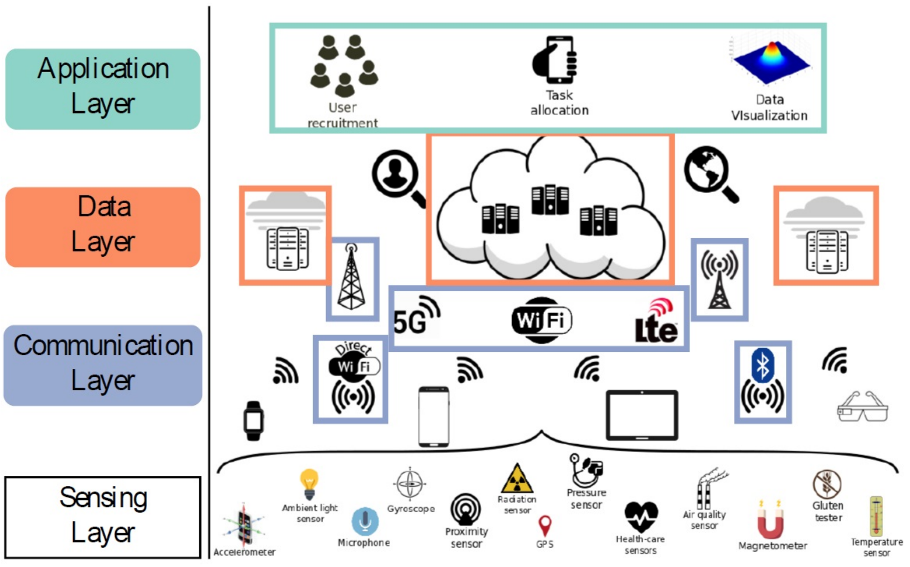

Figure 2: The layering used in the project

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

6.1 The scheduler receives all user requests

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 scheduler. In other words, the inputs include the input of the task scheduler, the graph of tasks and the graph of processors, and the outputs include the output of the task scheduler, the allocation of a processor to each task.

In this section, a private scheduler is created for the incoming workflows to decide which part of the request is executed on which resource.

In the final stage, a computing node is assigned to each task for execution.

The rate of nodes used in mobile crowd sensing is determined based on criteria such as the cost and energy consumption of nodes and the amount of coverage created for different areas. In fact, the better coverage rate is achieved by increasing the number of available nodes and improving the sampling rate. The minimum number of monitoring nodes for each service is stated in relation (6.1). By using the deep reinforcement learning algorithm, tasks are assigned to the participants. In this situation, the current state is entered as an input to the network under the name state st, and based on the current policy, an action from the set of actions is defined ($at \in A$) is achieved and the state changes to $st + 1$. Then, the reward amount is determined for the agent through the R function. The amount of reward is determined by considering the current state and the action taken. The rewarding function is defined according to the equation (6.2). The γ coefficient is present in the equations as a delay or reduction coefficient and helps to improve the convergence. In relation (6.4), the amount of fitness in each situation is obtained by calculating the amount of cost associated.

$$\mathrm{cont}(k) = n_b * t_b * t_s * \left(\frac{den_k}{\sum_{p \in B} den_p} \right) \tag{6.1}$$

$$R_t = r_{(t+1)} + \gamma r_{(t+2)} + \gamma^2 r_{(t+3)} + \ldots + \gamma^T r_{(t+T+1)} = \sum_{k=0}^{T} \gamma^k r_{(t+k+1)} \tag{6.2}$$

$$r = \begin{cases} -1, & if\ \text{coop}(j) < \tau \\ \text{Fitness}, & \text{else.} \end{cases} \tag{6.3}$$

$$\text{Cost} = \sum_{i=1}^{n} \sum_{j=1}^{m} E(i,j).T(i,j) \tag{6.4}$$

den_k: density in block k

t_s: number of IoT services

t_b: the number of blocks (regions) in the division of the environment

n_b: number of basic monitors per service in each block

τ: Threshold value for the number of participating nodes

$E(i,j)$: the amount of energy consumed by node i to collect data in service j

$T(i,j)$: participation or non-participation of node i in service j (in vector form with binary values)

Cost: Total cost in the allocation plan

The value of $E(i,j)$ is obtained through the equation (6.5). The coverage level used is also defined in relation (6.6). In this situation, the value of suitability is calculated as the ratio of the coverage level to the amount of costs created according to equation (6.7). In fact, in this situation, the quality of coverage and the amount of money spent in determining the best conditions for allocation are considered as essential factors.

$$E(i,j) = \frac{\text{dist}(i,dc)}{D(i)} * E_{\text{base}} \tag{6.5}$$

$$\text{Cover} = \frac{N(a)}{N(s)} \tag{6.6}$$

$$\text{Fitness} = \frac{\text{Cover}}{\text{Cost}} \tag{6.7}$$

$\text{dist}(i,dc)$: the distance of the monitoring node from the control center

$D(i)$: the maximum observable radius for node i

E_{base}: The average amount of energy used for a service

$N(a)$: Minimum number of monitoring nodes

6.2 Reverse Stackelberg game

In a game theory, the profit of all players should be considered. In this project, the actors include leading actors and subordinate actors, whose objective function is defined in relations (6.8) and (6.9), respectively. By reviewing the articles, it is clear that one of the most common games considered for the collective monitoring problem is the Stackelberg game. This game is based on complete information and is a two-way game [8]. The leading actors first select an action from the set (Ω_L) and the subordinate actors, observing this situation, take their respective actions from the set (Ω_F).

$$O.F(1) = \max \phi_m(t_m, p_m) \quad s.t. \ {}_n p_m^n t_m^n \leq \delta_m \tag{6.8}$$

$$O.F(2) = \max \Psi_m(t^n, p^n) \quad s.t. \ {}_m t_m^n \leq k_n \tag{6.9}$$

In this situation, all actors seek to maximize the objective function defined for them. It should be noted that the action of dependent function actors is done with the action of leading actors. This sequence is shown in figure 3. The main challenge in this situation is the high diversity and non-homogeneousness of the activities demanded by the users (leading actors), which causes disruption in reaching the optimal equilibrium point (Nash equilibrium point). In

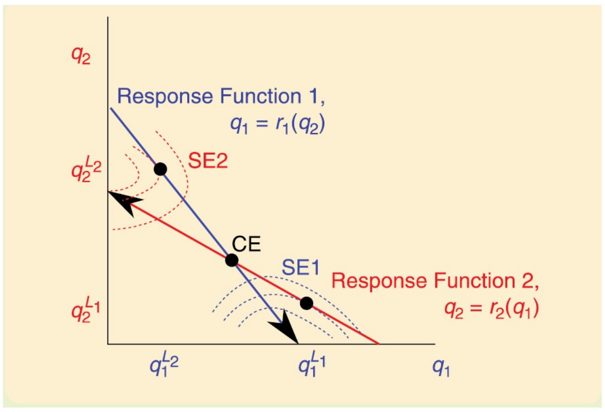

Figure 3: Finding the equilibrium point in the Stackelberg gamea

this project, the inverse Stackelberg game theory is used, which no longer requires full knowledge of the status of all actors, and through its implementation, the equilibrium point can be achieved.

The main advantage of the inverse Stackelberg game compared to the classic Stackelberg game is the possibility of its execution in situations where the actors have different reactions based on the requester's opinions. In this situation, the optimal point of the game can be calculated through the equation (6.10) and as follows Without the presence of the requester, in most cases this solution is the ability to provide favorable policies for all types of actors.

$$(U_L^d, U_F^d) \arg\min \varsigma_L(U_L, U_F) \qquad \forall (U_L, U_F)\Omega_L \times \Omega_F \tag{6.10}$$

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

k_n: Maximum time period for the nth user

6.3 Expression of the objective function

Allocation of tasks in mobile mass surveillance is part of a group of systems known as complex adaptive systems (CAS). These systems can communicate with each other on a wide level and have features such as the following:

- Aggregation of equipment (formation of common groups with each other)
- Has very non-linear characteristics
- Transferring data between different sources fluidly and continuously
- Great diversity in the behavior of systems towards each other

One of the most common techniques used for CAS systems is to use the reinforcement learning approach to train the agent in order to find the optimal policy in order to achieve the objective functions defined within the environment under study. When the dimensions of the problem increase significantly and the operational space becomes large, Q learning algorithms cannot be used in a traditional way. Because at this value the value function cannot be valued easily. To solve this challenge, approximation can be used. In this approach, neural networks along with Q learning algorithm can approximate the valuation function. This approach is known as deep Q network, in which neural network tables are used for Q table. This approach increases the quality of convergence by reducing parameter changes and using approximation functions. However, the network computing space for this purpose, the proposed solution is to use the proximal policy optimization algorithm (PPO), which was introduced in 2017. In this situation, without the need to discretize the operational space, the outputs are considered as a probabilistic distribution and try to improve the search space by sampling the distribution function. For a better distribution of tasks at the beginning, a fuzzy system is used, which facilitates their prioritization by separating tasks based on important and practical characteristics. Finally, by evaluating the conditions presented in relations (6.11) to (6.13), it is possible to adopt a suitable approach such that if relation (6.11) is established, then server i is left without performing any activity. be made If the constraint (6.12) is satisfied, server i will transfer the activities necessary for processing to server j, and if the relationship (6.13) is satisfied, server i will send the results of the activities to server j.

In the continuation of the processing process, the PPO technique will be used to achieve a more suitable convergence. In this case, based on the adaptive KL penalty, the target is estimated in the form of relation (6.14) and then the main objective function defined in the form of relation (6.15) is realized.

$$\max\{\beta_{ij}^s(t), \gamma_{ij}^s(t)\} < \max\{\beta_{ji}^s(t), \gamma_{ji}^s(t)\} \tag{6.11}$$

$$\max\{\beta_{ij}^s(t)\} > \max\{\gamma_{ij}^s(t)\} \tag{6.12}$$

$$\max\{\beta_{ij}^s(t)\} \leq \max\{\gamma_{ij}^s(t)\} \tag{6.13}$$

$$\beta_{ij}^s(t) = \varepsilon[Q_i^s(t) - Q_j^s(t)] - \xi_{ij}(t) \tag{6.14}$$

$$\gamma_{ij}^s(t) = \varepsilon[D_i^s(t) - D_j^s(t)] - \xi_{ij}(t) \tag{6.15}$$

$$J^{CPI}(\theta) = E_t[r_t(\theta).A_t] \tag{6.16}$$

$$J^{KLPEN}(\theta) = \max E_t[r_t(\theta).A_t - \beta KL(\pi_\theta(a_t|s_t) \| \pi_{\theta_{okl}}(a_t|s_t))] \tag{6.17}$$

ε: Step length size

$Q_i^s(t)$: Queue of activities of server i in state s at time step t

$D_i^s(t)$: the result queue of server i in state s at time step t

$\xi_{ij}(t)$: The cost of transmitting information from node i to node j in the t^{th} time step

$\pi_\theta(a_t|s_t)$: the selected probability of activity a despite establishing state s in time step t

θ: vector of policy weights

Value functions are used to evaluate the objective function. They can provide various services for the agent, such as recognizing the state of the state and identifying appropriate actions. These functions are defined according to the equation (6.17). In deep learning method, neural network is used for estimation. In this case, the value of Q function is estimated in each episode. In order to improve performance stability, the gradient descent algorithm has been used to minimize the output error.

6.4 Approximate policy

Real-world problems have huge dimensions, which make the application of Q-learning algorithms to realize them face serious problems. In this case, a phenomenon is created which is called the curse of dimensions (CoD) [9]. To solve this challenge, neural networks have been used for approximation. In this structure, the agent designs the appropriate policy considering the state of the system and the environment helps to produce a new state by performing the relevant action and the reward obtained. According to the obtained result, the parameters of the agent are updated. In addition, the use of approximation functions creates stability in the learning process and ensures convergence to a suitable extent. According to the argument stated in, using DNN in reinforcement learning algorithms reduces the change of parameters and increases the quality of convergence. In order to make the output distribution function continuous, the approximate policy (PPO) can be used, which has several advantages, including the following:

- High effectiveness in huge operating spaces
- More complete convergence features
- Ability to automatically learn stochastic policies

The modification process is carried out in such a way that a search is performed in a set of policies, which is called the policy space. Although the objective function is performed with the help of actions determined by the policies, by replacing the objective function with the PPO algorithm, the search in the parameter space is performed in such a way that the appropriate values for the parameters are determined. In this situation, the step size control is updated in the form of relation (6.18) using the learning rate. In this situation, since the distance between the two spaces is not the same as each other, it becomes difficult to choose the optimal step size (α). In other words, if (α) is meant to be small, the number of iterations increases and the training becomes time-consuming and costly, and the policy may get trapped in the local optimal point. If (α) is assumed to be large, the corresponding step in the policy space passes through the neighborhood of appropriate policies and may cause a loss of efficiency, and this process continues and the possibility of recovery becomes very weak. In order to introduce a solution to overcome this challenge, it is first necessary to calculate the relative policy performance identifier as the difference between the value of the objective function in the current step and its value in the next step according to equation (6.19). In the following, the PPO technique should be used to achieve the improvement process. In fact, the PPO technique is one of the algorithms that analyze the constrained policy optimization problem well. In this regard, the amount of adaptive penalty is calculated and then the objective function is obtained according to equation (6.21).

$$\Delta\theta = \alpha ._{\vartheta} J(\pi_\vartheta) \tag{6.18}$$

$$J(\pi') - J(\pi) = E_{\tau\pi'}\left[{}_{t=0}^{T}\gamma^T . A^\pi(s_t, a_t)\right] \tag{6.19}$$

$$J^{CPI}(\theta) = E_t[r_t(\theta).A_t] \tag{6.20}$$

$$J^{KLPEN}(\theta) = \max E_t[r_t(\theta).A_t - \beta KL(\pi_\theta(a_t|s_t)\|\pi_{\theta_{old}}(a_t|s_t))] \tag{6.21}$$

E: Expected value from a set of samples

J: the corresponding objective function for each segment

$\pi_\theta(a_t|s_t)$: Probability of selecting activity a at state time s and time step t

θ: the vector of weights designed for the policy

Approach DRL has 2 important challenges as follows:

I Lack of stability in the learning process

II Complex design of reward system

6.5 Modeling the problem

To express the desired model, assume that the number of leading actors is equal to M (activity proposers in the MCS space) and the number of functional actors is equal to N (users who earn money by accepting received requests and performing tasks) is equal to N. In this case, the goal of each leading actor is to complete the desired tasks with the lowest possible cost. In this situation, the amount of profit earned by each subordinate actor and each leading actor is calculated through relations (6.22) to (6.23) respectively.

$$\Psi_n(t^n, p^n) =_m p_m^n t_m^n -_m C_m(t^n) \tag{6.22}$$

$$\phi_m(t_m, p_m) = \varphi_m(t_m) -_n p_m^n t_m^n \tag{6.23}$$

p_m^n: Paid fee for executing the m requested task to the n^{th} user

t_m^n: the time interval of the execution of the m^{th} activity through the n^{th} user

$C_m(t^n)$: The full cost of executing the m^{th} activity for the n^{th} user

$\varphi_m(t_m)$: the processing income of m activity

In an MCS system, activities are distributed in such a way that the profit of all actors is comprehensively maximized. Since the leading actor takes action first, the action of the actors is dependent on it ($U_F = f(U_L)$). In fact, such a function leads to providing a kind of curve of dependent actors. The main challenges in this situation originate from the fact that in the context of IoT, the type of activities offered by the leading players are both high in number and spread in a completely heterogeneous manner in the space under study [10]. This leads to disruption in the implementation of Stackelberg's game theory in reaching the equilibrium point.

7 Results evaluation criteria

7.1 The quality of service delivery

Suppose the sensor network consists of a group of mobile users (MU) with vector $V = \{v_0, v_1, ..., v_n\}$ and each request sent to the network consists of m different tasks ($J = j_1, ..., j_m$). The quality of service delivery, which is considered one of the most important indicators defined in the field of job allocation, is defined according to the relation (7.1).

$$Q_i = \frac{{}_{j=1}^{N_i} r_i}{N_i} \tag{7.1}$$

7.2 Period of activity

One of the important criteria is defined as minimizing the time period for completing the activity. This time period includes the time period when the results of the submitted request are sent to the user. In this situation, the minimization of the average completion interval m of the activity (AM) defined in relation (7.2) is considered. Usually, the duration of communication follows the exponential distribution that results from the relation (7.3).

$$\min AM = \frac{1}{m} ._j \; {}_J M(j)|_\Pi \tag{7.2}$$

$$t_{ij} =_0^\infty t.\lambda_i.e^{-\lambda_i t} dt = \frac{1}{\lambda_i} \tag{7.3}$$

t_{ij}: Expected amount of communication time between users and requesters

7.3 Coverage quality

One of the goals of the presented approach is to maximize the coverage quality of the area under study, which is analyzed through relations (7.4) to (4-31). The final evaluation criteria of the plan include the monitoring cost (energy consumption rate) and the monitoring task service quality (coverage rate) of the proposed plan to calculate the quality of assigning tasks to participating users. In some previous models of mobile monitoring, users participated independently in the monitoring process, and therefore some tasks attracted more users due to high rewards, and tasks with lower rewards covered fewer users. In this situation, data collection and their integrated management is important.

$$Cov(t) = \frac{1}{|TG_t| \times |SG_t|} .cy \; TG_t \; reg \; SG_t \; Cov(cy_t, reg_t) \tag{7.4}$$

$$Cov(cy_t, reg_t) = \frac{1}{\gamma} .\omega \; W \; AS_t(\omega, cy, reg).p(\omega, cy, reg) \tag{7.5}$$

The collection of volunteer participants to carry out the activity: $W = \{\omega_1, \omega_2, ..., \omega_n\}$

All mobile Friday monitoring activities: $T = \{t_i\}$

the activity cycle of the ti sensor: $TG_{ti} = \{cy_{ti}^k\}$

the divided areas ti m: $SG_{ti} = \{reg_{ti}^k\}$

cycle pair: (cy_{ti}^x, reg_{ti}^y)

The obtained activity assignment solution: AS

representative of repeated samples: γ

The estimated probability of the candidate passing through the specified in the specified time period: p

7.4 Data collection

In this project, the data set defined in the reference is used. This data is the result of the tracking of requested tasks on the clusters of Google, which was collected in a period of 5 hours and 15 minutes. Each request consists of a number of tasks, each task belongs to a separate task. A task may also include a number of functions. Each row in the dataset represents the execution of a separate task in a 5-minute interval and includes the following features:

- duration (in seconds) (Time)
- Job ID (unique code for each specific job) (JobID)
- Task ID (unique code for each specific task) (TaskID)
- Type of work: an integer in the range [0,1,2,3] to define the type of work requested (JobType)
- Normalized task cores: The average number of CPUs used (Normalized Task Core)
- Normalized task memories: Average memory used for each individual task (Normalized Task Memory)

In this structure, data sets are anonymized in various ways. Job and task names are replaced with numeric identifiers and all timestamps are introduced corresponding to an anonymous reference value. Time information is also reported in 5-minute intervals. The amount of memory consumption and the amount of CPU cores used are normalized with the help of an unknown linear transformation, and no information is available about the meaning of the specified job types. This tracking includes 21 time intervals of 5 minutes. In total, 3535028 observations, 9221 unique tasks and 176590 tasks are defined in the target system. There are two unexpected types of functions in datasets:

Some functions only require memory, which are referred to as inactive ideal functions and only require storage in memory.

There are some functions that neither need a core nor memory. These functions do not use any resource, but their task pointer is maintained in the system in order to quickly respond to the services requested by users.

Table 1 shows the specifications of different categories of works. Table 2 shows the average numerical values obtained for each category. It should be noted that groups labeled CPU have higher memory-to-core rates than groups labeled Mem. When a category has many of the same type, it is necessary to determine whether the type of work considered is common to all, or whether all the multiple tasks must be performed differently for all categories. Table 3 shows the correlation coefficient between types of work for this collection. The value of coefficient changes varies from 1 (strong correlation) and 0 (no correlation) to −1 (strongly uncorrelated) [11].

Table 1: Work specifications for different categories

Cluster description	Active Long CPU	Active Long CPU Fewtasks	Active Long Mem	Active Short CPU	Active Very Short CPU Fewtasks	Active Very Short Mem	Inactive Short	Inactive Long	Total
Total	184	576	44	354	777	4818	1375	1090	9218
Type 0	37	93	6	212	72	1260	318	293	2291
Type 1	36	104	0	67	12	2280	531	203	3233
Type 2	90	237	28	75	692	1276	519	398	3315
Type 3	21	142	10	0	1	2	7	196	375

Although the results of table 3 alone are not enough for capacity planning, but still strong correlation coefficients are related to several data points and show the number of related tasks. In fact, it is necessary that tables 1 and 3 be evaluated together to explain the distribution of types of work among the categories [12].

Table 2: Numerical values for the average of different categories

Cluster description	Active Long CPU	Active Long CPU Fewtasks	Active Long Mem	Active Short CPU	Active Very Short CPU Fewtasks	Active Very Short Mem	Inactive Short	Inactive Long
Active task	0.03176	0.00247	0.04950	0.00423	0.00001	0.00005	0	0
Duration	0.95974	0.96592	0.97458	0.06958	0.01690	0.01727	0.03483	0.97473
Mem/core ratio	0.06671	0.04229	0.27867	0.01017	0.00821	0.04892	–	–
Tasks count	0.03876	0.00766	0.02799	0.04239	0.00049	0.00116	0.00087	0.00441

Table 3: Correlation coefficient between different types of work

Cluster description	Active Long CPU	Active Long CPU Fewtasks	Active Long Mem	Active Short CPU	Active Very Short CPU Fewtasks	Active Very Short Mem	Inactive Short	Inactive Long
Type 0	−0.02	−0.06	−0.03	0.26	−0.12	0.01	−0.04	0.04
Type 1	−0.05	−0.11	−0.06	−0.09	−0.27	0.33	0.22	−0.22
Type 2	0.04	0.05	0.05	−0.14	0.41	−0.27	0.01	−0.01
Type 3	0.10	0.40	0.11	−0.04	−0.05	−0.24	−0.32	0.32

References

[1] F. Bao, R. Chen, and J. Guo, *Scalable, adaptive and survivable trust management for community of interest-based internet of things systems*, IEEE 11th Int. Symp. Autonomous Decentr. Syst. (ISADS), IEEE, 2013, pp. 1–7.

[2] E. Barrett, E. Howley, and J. Duggan, *Applying reinforcement learning towards automating resource allocation and application scalability in the cloud*, Concur. Comput.: Pract.Exper. 25 (2013), no. 12, 1656–1674.

[3] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, *A survey on Mobile crowd sensing systems: Challenges, solutions, and opportunities*, IEEE Commun. Surv. Tutor. 21 (2019), no. 3, 2419–2465.

[4] M. Chen, T. Wang, K. Ota, M. Dong, M. Zhao, and A. Liu, I*ntelligent resource allocation management for vehicles network: An A3C learning approach*, Comput. Commun. **151** (2020), 485–494.

[5] T. Das, P. Mohan, V.N. Padmanabhan, R. Ramjee, and A. Sharma, *PRISM: Platform for remote sensing using smartphones*, Proc. 8th Int. Conf. Mobile Syst. Appl. Services, 2010, pp. 63–76.

[6] S. Ghasemi Falavarjani, M.A. Nematbakhsh, and B. Shahgholi Ghahfarokhi, *A multi-criteria resource allocation mechanism for mobile clouds*, Int. Symp. Comput. Networks Distrib. Systems, Springer, Cham, 2013, pp. 145–154.

[7] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, *FlierMeet: A mobile crowdsensing system for cross- space public information reposting, tagging, and sharing*, IEEE Trans. Mobile Comput. **14** (2014), no. 10, 2020–2033.

[8] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, *ActiveCrowd: A framework for optimized multitask allocation in Mobile crowd sensing systems*, IEEE Trans. Human-Machine Syst. **47** (2016), no. 3, 392–403.

[9] S. He, D.-H. Shin, J. Zhang, and J. Chen, *Toward optimal allocation of location dependent tasks in crowdsensing*, IEEE INFOCOM 2014-IEEE Conf. Comput. Commun., IEEE, 2014, pp. 745–753.

[10] M. Hussin, N.A.W.A. Hamid, and K.A. Kasmiran, *Improving reliability in resource management through adaptive reinforcement learning for distributed systems*, J. Paral. Distrib. Comput. **75** (2015), 93–100.

[11] M. Hussin, Y.C. Lee, and A.Y. Zomaya, *Efficient energy management using adaptive reinforcement learning-based scheduling in large-scale distributed systems*, Int. Conf. Parall. Process., IEEE, 2011, pp. 385–393.

[12] G. Javadzadeh and A.M. Rahmani, *Fog computing applications in smart cities: A systematic survey*, Wireless Networks **26** (2020), no. 2, 1433–1457.