INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Implementation of Hybrid Adaptive Learning Algorithm for Task Offloading

Priyanka Sarma[1]*, **Atowar Ul Islam**[1]

1 University of Science & Technology, Ri-Bhoi, Meghalaya, India

OPEN ACCESS

Received: 15-07-2024

Accepted: 10-09-2024

Published: 08-10-2024

Citation: Sarma P, Islam AU (2024) Implementation of Hybrid Adaptive Learning Algorithm for Task Offloading. Indian Journal of Science and Technology 17(38): 3929-3936. https://doi.org/10.17485/IJST/v17i38.2280

* **Corresponding author**.

psarma157@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: Offloading tasks in edge computing (EC) plays an important role in optimizing resource utilization and enhancing the system performance. This paper studies various AI-based computation offloading (CO) strategies and proposes a hybrid Adaptive Learning Algorithm, that reduces the latency significantly compared to traditional RL-based on-policy and off-policy algorithms namely Q-Learning (QL) and State Action Reward State Action (SARSA) for CO in an EC environment. The paper evaluates and compares the efficiency of the proposed algorithms in optimizing dynamic offloading decisions, focusing on factors such as latency. **Methods:** The research is inspired by existing literature that has discovered the various applications of QL and SARSA in CO and mobile EC. This paper inspects how these algorithms perform in terms of reducing overall latency and proposes an adaptive hybrid algorithm. **Novelty:** Although other AI-based techniques exist in earlier research on CO in EC, the primary novelty proposed is the addition of a hybrid adaptive RL algorithm. Factors impacting the task offloading decision are crucial in the proposed algorithm. Moreover, the decision is not based only on static policies; the current state is the driving factor for decision-making. **Findings:** The exploration and exploitation handshake is very promising in reducing latency, as seen in the simulation results. The proposed algorithm outperforms Q-Learning and SARSA in terms of latency.

Keywords: Computation offloading; Mobile Edge Computing; Reinforcement learning; Qlearning; SARSA

1 Introduction

Some computationally demanding jobs, such as online gaming, facial recognition, augmented reality, and virtual reality (AR/VR), have been significantly influenced by the restricted processing capabilities of smart devices, which have increased in the recent era of 5G. Billions of sensory data being generated by smart devices must also be processed and stored relatively quickly. Therefore, edge computing (EC), a new computing paradigm, has emerged to get around these restrictions and offer a viable way to close the gap between the constrained resources on mobile devices and the ever-rising computing demands made by mobile applications. Unlike traditional cloud computing service providers, Mobile Edge Computing (MEC) enhances access to networks and consumer devices with data processing capabilities [(1)]. Thus, user data is transferred to

edge nodes in spite of centralized servers for processing and storing, as edge servers handle data processing and storing more quickly than centralized servers, resulting in more efficiency and improved service quality.

Computation offloading techniques for EC have been thoroughly studied to gain better energy efficiency or a better computing experience. Typically, the CO problems of EC are divided into two categories: partial offloading, which entails sending a part of the application components to an edge server, and total offloading, which sends all calculations to the remote edge server [2]. Partial offloading puts an additional burden on computing resources and energy reserves as it requires figuring out the computational cost for every application component, according to Orsini et al. [3]. However, these computations provide a way to shrewdly choose the best parts to offload, which minimizes data transmission, lowers latency, and uses less energy overall. Min et al. [4] show the effect of a Q-learning-based offloading scheme in achieving optimal computation offloading strategies. Alfakih et al. [5] focused on Task Offloading (TO) as well as allocation of resources for MEC using SARSA. Additionally, studies by Wang et al. [6] and Khanh et al. [7] have highlighted the application of reinforcement learning methods for improving task offloading performance in IoT and heterogeneous IoT environments, respectively. The study investigated how computation-intensive applications may be supported by mobile edge computing (MEC) in 5G networks. It focused on a multi-user MEC system in which compute activities are offloaded by user equipment (UEs) via wireless channels to an MEC server. The goal of optimization was to reduce delay and energy consumption cost for each user. MEC that is based on reinforcement learning (RL) to handle the system's dynamic character. To achieve this author introduced and simulated two reinforcement learning (RL)-based schemes: Q-learning and Deep Reinforcement Learning (DRL). The method considerably lowered the sum cost.

Wang et al. [8] proposed a task-offloading approach called Lyapunov optimization-based dynamic computation offloading (LODCO) to enhance mobile-edge computing (MEC) systems by integrating renewable energy sources, specifically energy harvesting (EH) technologies. The LODCO algorithm optimizes task execution cost in every time slot by dynamically making online decisions based on the current system state, without the need for distributed information. Simulation results demonstrate that LODCO outperforms benchmark greedy policies, significantly reducing computation failures and execution costs. This makes LODCO a promising solution for designing sustainable and efficient MEC systems.

A new task offloading technique (TaSRD) was presented by Cao et al. [9]. It developed optimization measures grounded on the time and energy models. Although they neglected the energy optimization on the MEC server, the testing findings demonstrate that the algorithm has a good effect in lowering job completion times and energy usage.

While these approaches prove beneficial in addressing allocation of resource and task offloading challenges, the latency reduction is an area of interest. The emergence of machine learning techniques offers a promising alternative for tackling optimization problems such as task offloading and resource allocation. This shift provides new opportunities for exploration in research and enhances the efficacy of objective optimization strategies. In order to balance the decrease in energy usage and delay, Wang et al. [10] suggested a task-scheduling method based on reinforcement learning. According to experimental findings, the suggested method outperforms alternative comparison algorithms.

Effective resource allocation should align with the decision-making process regarding whether to partially or fully offload tasks. Resource allocation is influenced by the application's partitioned and parallelized offloading capabilities. If offloading proves unfeasible, partitioned and parallelized applications are assigned only a single computing node. The selection of applications to be offloaded to the mobile edge (ME) should meet the requirements for computing time and energy consumption [11]. Guo et al. [12] shares similarities with Eshratifar et al. [13], but it goes beyond minimizing computation time by also lowering energy consumption at the ME. The authors suggest the identification of various hotspots within the User Equipment (UE) density region, facilitating Mobile Devices (MDs) access to the ME through the enhanced Node B (eNB).

An algorithm that considers various parameters viz. task characteristics, resource availability while offloading a task can be promising. The lack of dynamic real-time decision-making to offload tasks is proposed in this paper. This paper proposes an adaptive algorithm that is extended from both Q-Learning and SARSA algorithm. It devices the advantages of both algorithms. The aggressive optimization capabilities of Q-Learning is combined with the exploration safety of SARSA. The action space is quickly discovered using Q-Learning. For highly sensitive environments as in Edge, the capabilities of SARSA are used to refine the policy of offloading. A parameter is introduced β to indicate the threshold. If the stability measure is above β, Q learning update is processed otherwise SARSA update. The effectiveness of the hybrid approach depends on how well it can adapt to different phases of the learning process and balance exploration with exploitation

Therefore, the primary contributions of this paper are outlined as follows:

a) To optimize latency for intensive tasks for Edge computing environment, we proposed a hybrid adaptive algorithm for task offloading. The proposed algorithm makes the offloading decision based on the current system state, thereby reducing the latency.

b) The paper contributes by providing a thorough comparative analysis of adaptive hybrid algorithm, Q-Learning and SARSA, in regards to task offloading within Edge computing environments. The simulation results shows the performance of hybrid adaptive algorithm along with Q-Learning and SARSA.

The remainder of this paper is planned as follows. Section 2 describes the methodology. Section 3, discusses the results and analysis. Section 4 concludes this paper.

2 Methodology

To minimize the task offloading process we proposed RL-based ML algorithms. A comprehensive evaluation of QL, SARSA and Hybrid adaptive algorithms for task offloading in EC, the following steps need to be used in the methodology.

Methodology steps involved in Q-Learning Algorithm

a) Problem Formulation

- State Space (S): Describe the various states of the EC environment, including the available resources, the state of the network, and the tasks that need to be completed.
- Action Space (A): Define the possible actions, such as offloading a task to a specific edge i.e. remotely or executing it locally.
- Reward Function (R): Design a reward function that incentivizes desired outcomes, such as minimizing latency and energy consumption. The reward could be a function of the task completion time, resource utilization, and energy efficiency.

b) Initialization

- Initialize the Q-table, a matrix where each entry Q (s, a) represents the expected utility of taking action a in state s.
- Set the learning rate (α), discount factor (γ), and exploration rate (ε) parameters.

c) Algorithm execution

- The Algorithm is discussed below.

d) Policy extraction

- After sufficient training episodes, derive the optimal policy from the Q-table by selecting the action that has the highest Q-value for each state.

Pseudo-code of Q learning, SARSA and Hybrid Adaptive for task offloading.

Algorithm 1: Pseudo code of Q-learning algorithm

- For each s, αinitialize the table entry $\dot{Q}(S,\ A)$to zero
- Observe the current state $\dot{S}$
- Do forever:
- Select an action A and execute it
- Receive immediate reward r
- Observer the new state S'
- Update the table entry for $Q'(S,\ A)$as follows: $Q'(S,\ A) < -r + \gamma\ max\ Q'(S',\ A')$
- $S < -S'$

The above **algorithm1** is designed to solve problems where an agent makes decisions sequentially in an environment to maximize cumulative rewards. Agent learns an optimal policy by iteratively exploring the environment, taking actions, and updating its knowledge based on the received rewards. The algorithm maintains a Q-table, which stores the expected rewards for different actions in different states.

Algorithm 2: Pseudo code of SARSA algorithm

- Initialize Q value: Start from initial state
- Being in State S
- -Choose action A' [according to policy $\pi(\mathrm{S}', \mathrm{A}')$]
- Observe reward R and next state S'.

- Choose action Á in state S⊠. [according to policy $\pi(S', A')$]
- 5.Update with SARSA update rule-
- $Q(S,\ A) < -Q(S,\ A) + \alpha(R + \gamma.Q(S' + A') - Q(S,\ A))$
- Go to step 2
- Stop when all Q values have converged.

In Figure 1, these states are impacted by actions following a policy and is rewarded when the agent executes the selected action. Q-learning, an off-policy strategy in which the agent gains knowledge from activities directed by an alternative policy. SARSA operates as an on-policy approach, depending on policy-dictated actions for learning. RL has demonstrated its efficiency in tasks such as resource allocation, cloud computing, and computation offloading[11].

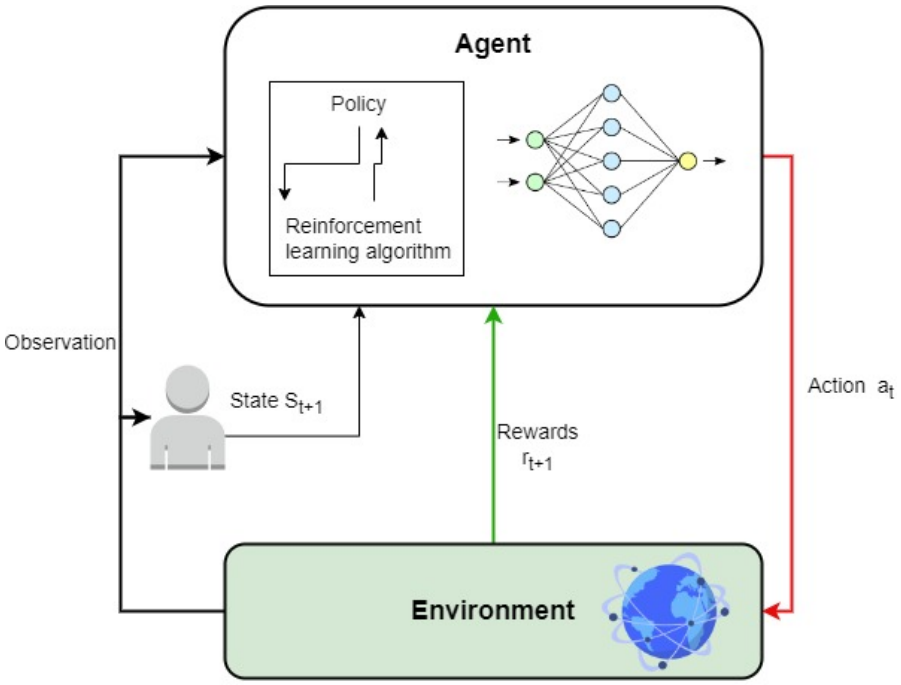

Fig 1. System model of RL

Algorithm 3: Pseudo code of Adaptive hybrid algorithm

- initialize Q-table with all zeros
- **iterate for each episode**
- initialize state
- while not done
- policy derived from Q (e.g., ε-greedy) is used to select the action
- take action, observe reward, next state
- calculate dynamic context factor based on current environment
- adjust reward using context factor
- update Q-value using adapted learning rate based on environmental feedback
- state ← next state
- **end while**
- **end for**

This proposed **algorithm 3** is an adaptive algorithm that exhibits the goods of both Q-learning and SARSA. The aggressive optimization capabilities of Q-Learning are combined with the exploration safety of SARSA. The action space is quickly discovered using Q-Learning. For highly sensitive environment as in Edge, the capabilities of SARSA are used to refine the policy of offloading. A parameter is introduced β to indicate threshold. If the stability measure is above β, Q learning update

is processed otherwise SARSA update. The effectiveness of the hybrid approach depends on how well it can adapt to different phases of the learning process and balance exploration with exploitation.

3 Results and Discussion

3.1 Experimental Setup and Performance Evaluation

To decide whether or not to offload at a particular moment, the model uses N tasks from U users. For every user, we provide data sizes as input and output. Our goal is to find the ideal policy offloading function. The offloading size in MEC networks may be represented as NU, which rises as the number of jobs N for each user U increases. We assume several tasks distributed between N = 10 to 50 for each mobile user with an average of 10 to 50 tasks per user. Table 1 shows all parameters that are used in proposed reinforcement learning algorithms. We assume 3.55 X10^6 J/bit as the equivalent power consumption and 3.75X10^7 s/bit for local processing times of MDs. The remaining network characteristics, such as bandwidth, are subject to change based on the network, but we'll assume that it is 150 MB for both uplink and downlink traffic between a user and an edge server.

Table 1. Parameter setup for simulation

Edge Nodes	Tasks	Q-Learning Parameters	SARSA Parameters	Hybrid Adaptive Parameters	Episodes
No. of Nodes: 3000	No. of Tasks: 10,50	Alpha = 0.1, Gamma = 0.9, Epsilon = 0.1			100

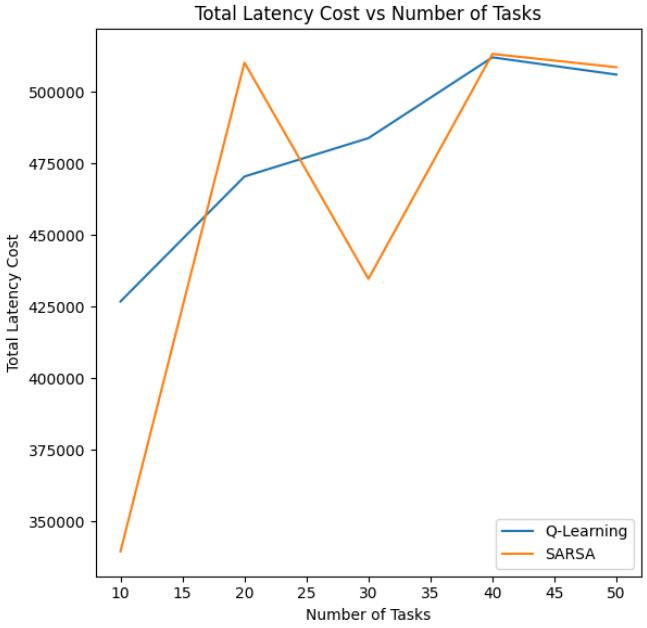

Fig 2. Latency comparison for Q learning and SARSA

Q-learning and SARSA are very similar, while SARSA makes use of an on-policy approach [14]. This motivated us to apply it to enhance the task offloading performance to the EC environment, especially as it doesn't require explicit policy function learning from the agent. SARSA works better than Q-learning, as can be shown from the findings in Figure 2. Once the simulation runs, it is evident from Figure 2 that both Q-Learning and SARSA show an increase in mean latency as the task count rises. Higher tasks lead to higher competition among nodes for resources, which is to be expected. Figure 3 and Figure 4 shows the Episodes vs rewards plot and latency vs Episodes plot respectively. Additionally, compared to Q-Learning, SARSA exhibits more steady performance with reduced fluctuation in mean latency across various task counts. When comparing SARSA to Q-Learning, the mean latency is smaller. This indicates that when it comes to lowering the average execution time, SARSA performs better.

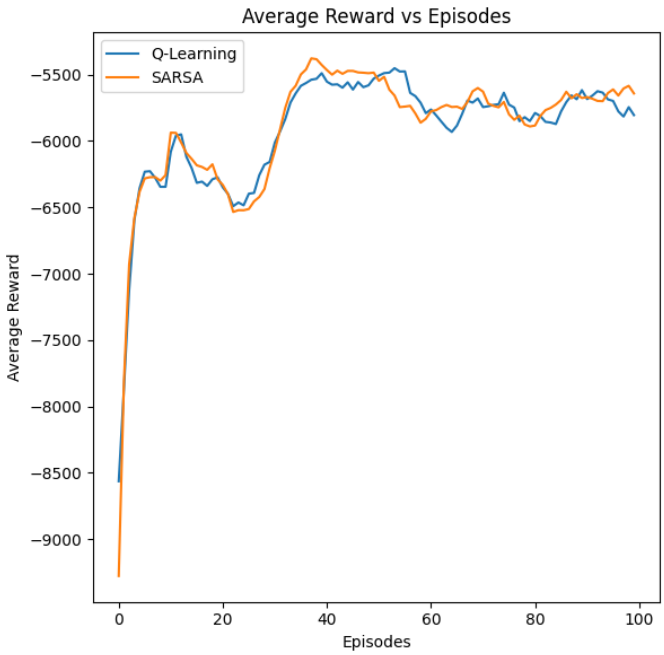

Fig 3. Reward vs Episodes for Q learning and SARSA

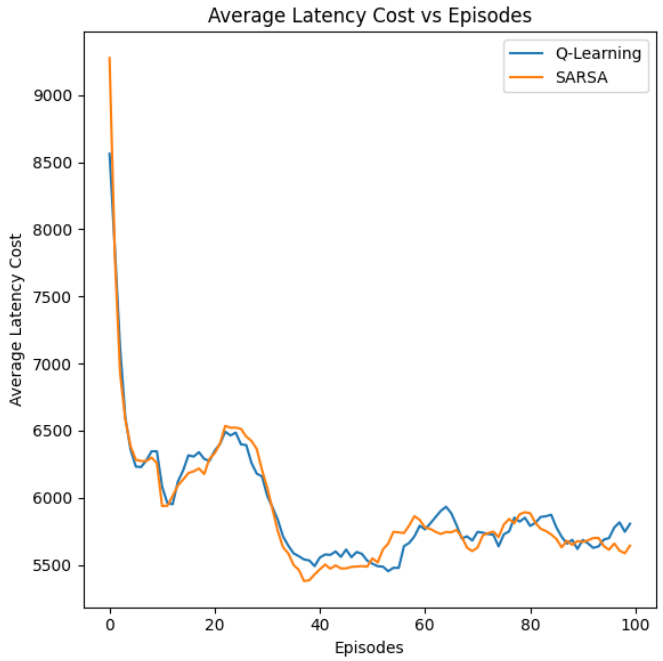

Fig 4. Latency vs Episodes for Q learning and SARSA

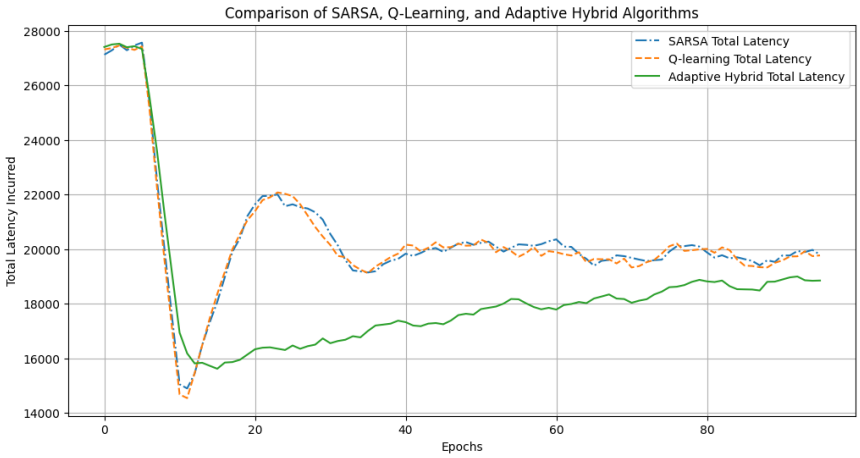

Fig 5. Comparison of Q-learning, SARSA and Proposed Algorithm

The performance of the proposed algorithm is shown in Figure 5. It is evident from the figure that the proposed algorithm outperforms the other two methods with respect to latency. This is possible only for the approach taken for the exploration safety using SARSA and the aggressive optimization of Q-Learning algorithm. Thus, taking advantage of both algorithms in a complex environment as in edge computing provides better results in terms of latency reduction as the proposed novelty of our approach.

Simulation results exhibit a good performance compared to the other two methods. All three algorithms start at a similar latency; however, the proposed approach quickly diverges showing a steep descent in the latency. At about 20 to 40 epochs, SARSA and Q-Learning show varied performance and the proposed algorithm shows reduced latency. This is a clear indication of the exploration settings, taking into consideration the proposed approach exploration-exploitation balance. After 40 epochs, the proposed algorithm maintains a reduced latency continuously indicating long-term learning superiority. On the contrary, SARSA and Q-Learning show oscillations indicating less stability.

4 Conclusion

In this comparative study conducted in an edge computing environment, Q-Learning and SARSA algorithms were evaluated. The results revealed that SARSA consistently outperformed Q-Learning by exhibiting lower mean latencies across various task counts. This suggests SARSA's superior efficiency in task allocation, leading to shorter average task execution times. Moreover, SARSA demonstrated remarkable stability in performance, indicating its reliability in edge computing scenarios where consistent results are crucial. While SARSA appears promising for minimizing latency and ensuring consistent performance, further exploration is needed to assess other parameters such as overall execution time and energy efficiency. This is further extended in the proposed approach to take advantage of the aggressive optimization of the Q-learning algorithm and the safety of exploration in SARSA. The proposed algorithm novelty lies in this approach reducing the latency. However, it is promising to incorporate more learning and testing phase in order to enhance the results of the proposed algorithm. Considering the task characteristics can be helpful to make a more optimal decision for offloading the tasks in Edge Environment. In the future work, evolutionary algorithms for optimization will be considered to address the issue of choosing whether to offload locally or offload the task in order to improve the performance of the algorithm to reduce latency. Further, we will incorporate EA's for real-time decision-making task offloading algorithms and also investigate ant colony optimization, Genetic algorithms etc.

References

1) Islam A, Debnath A, Ghose M, Chakraborty S. A survey on task offloading in multi-access edge computing. *Journal of Systems Architecture*. 2021;118:102225–102225. Available from: https://doi.org/10.1016/j.sysarc.2021.102225.
2) Cao B, Li Z, Liu X, Lv Z, He H. Mobility-aware Mult objective task offloading for vehicular edge computing in digital twin environment. *IEEE Journal on Selected Areas in Communications*. 2023. Available from: https://doi.org/10.1109/JSAC.2023.3310100.
3) Orsini G, Bade D, Lamersdorf W. Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications. In: and others, editor. IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W), IEEE. IEEE. 2016;p. 216–221. Available from: https://doi.org/10.1109/FAS-W.2016.54.

4) Min M, Xiao L, Chen Y, Cheng P, Wu D, Zhuang W. Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Transactions on Vehicular Technology*. 2019;68(2):1930–1941. Available from: https://doi.org/10.1109/TVT.2018.2890685.
5) Alfakih T, Hassan MM, Gumaei A, Savaglio C, Fortino G. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. *IEEE Access*. 2020;8:54074–54084. Available from: https://doi.org/10.1109/ACCESS.2020.2981434.
6) Wang T, Luo X, Zhao W. Improving the performance of tasks offloading for internet of vehicles via deep reinforcement learning methods. *IET Communications*. 2022;16(10):1230–1240. Available from: https://doi.org/10.1049/cmu2.12334.
7) Khanh TT, Hai TH, Hossain MD, Huh EN. Fuzzy-assisted mobile edge orchestrator and sarsa learning for flexible offloading in heterogeneous iot environment. *Sensors*. 2022;22(13):4727–4727. Available from: https://doi.org/10.3390/s22134727.
8) Wang J, Hu J, Min G, Zomaya AY, Georgalas N. Fast adaptive task offloading in edge computing based on meta reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*. 2020;32(1):242–253. Available from: https://doi.org/10.1109/TPDS.2020.3014896.
9) Wang J, Hu J, Min G, Zhan W, Zomaya AY, Georgalas N. Dependent task offloading for edge computing based on deep reinforcement learning. *IEEE Transactions on Computers*. 2021;71(10):2449–2461. Available from: https://doi.org/10.1109/TC.2021.3131040.
10) Wang Y, Wang K, Huang H, Miyazaki T, Guo S. Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. *IEEE Transactions on Industrial Informatics*. 2018;15(2):976–986. Available from: https://doi.org/10.1109/TII.2018.2883991.
11) Zhao T, Zhou S, Guo X, Zhao Y, Niu Z. A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. IEEE. 2015;p. 1–6. Available from: https://doi.org/10.1109/GLOCOMW.2015.7414063.
12) Guo X, Singh R, Zhao T, Niu Z. An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems. *IEEE International Conference on Communications (ICC)*. 2016;p. 1–7. Available from: https://doi.org/10.1109/ICC.2016.7511147.
13) Eshratifar AE, Pedram M. Energy and Performance Efficient Computation Offloading for Deep Neural Networks in a Mobile Cloud Computing Environment. In: and others, editor. GLSVLSI '18: Proceedings of the 2018 Great Lakes Symposium on VLSI. 2018;p. 116–116. Available from: https://doi.org/10.1145/3194554.3194565.
14) Jiang K, Zhou H, Li D, Liu X, Xu S. A Q-learning based Method for Energy-Efficient Computation Offloading in Mobile Edge Computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE. 2020;p. 1–7. Available from: https://doi.org/10.1109/ICCCN49398.2020.9209738.