Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Intelligent Task Scheduling in Edge-Cloud Environments Using Double Deep Q-Network Reinforcement Learning

Vishakha Makode[1], Taresh Ayaspure[2]

[1]Dept. of computer science & engineering prestige institute of engg. mgmt & research Indore, India
[2]Dept. of computer science & engineering prestige institute of engg. mgmt & research Indore, India

Abstract:
In today's world, with the emergence of IoT devices and critical latency applications, there has been an increased demand for intelligent resource management in diverse computing environments [5][10]. Collaborative computing is the integration of cloud and edge computing, while the issue of how to schedule each task for execution is still an open problem. Traditional heuristics like Round Robin and greedy latency reduction are incapable of adapting to the stochastic and non-stationary characteristics of practical workloads [4][14]. In this paper, an intelligent task scheduling mechanism based on Double Deep Q-Network (DDQN) reinforcement learning [2] has been proposed. The agent observes a four-dimensional state encoding task characteristics and selects binary offloading decisions, guided by a shaped reward signal encoding multiple performance objectives. Experimental evaluation on a heterogeneous synthetic benchmark demonstrates that the proposed DDQN scheduler reduces SLA violations by approximately 85% relative to Round Robin and 72% relative to the greedy baseline, while achieving superior energy efficiency. These results confirm that deep reinforcement learning [1][17][18] provides a principled foundation for adaptive resource management in next-generation edge-cloud systems.

Index Terms: Edge Computing, Cloud Computing, Deep Reinforcement Learning, Double Deep Q-Network, Task Scheduling, Task Offloading, SLA Violation, Energy Efficiency, IoT, Mobile Edge Computing.

I. INTRODUCTION

The fast development of edge intelligence, autonomy and real-time analysis has drastically altered the requirements imposed on the computing infrastructure [10][12]. While cloud-based infrastructures provide enormous computation power, their inherent delays make them unsuitable for latency-critical applications like augmented reality, vehicular autonomy coordination and industrial automation [3][14]. Edge computing overcomes the problem of high latencies by placing the computation resources near the data origin point, thus minimizing round-trip latencies and alleviating network load [3][11].

Nevertheless, edge nodes are inherently resource-constrained. Their limited processing power and energy budgets make them unsuitable for computationally heavy tasks that benefit from the elastic scalability of centralized cloud infrastructure [5][15]. Practical deployments must implement a task offloading layer that

intelligently decides, for each arriving task, whether local edge execution or remote cloud execution is more appropriate given the prevailing system state [6][16].

Traditional scheduling strategies such as Round Robin allocate resources in a fixed cyclic order, ignoring task heterogeneity [13]. Greedy approaches attempt to minimize a single metric such as latency but remain myopic with respect to energy consumption and long-term SLA compliance [4]. Neither class of algorithm is equipped to adapt to dynamic changes in workload distribution, resource availability, or network conditions [15].

Reinforcement learning (RL) provides an effective approach in addressing offloading as a sequence decision process [18]. The Deep Q-Network (DQN) approach expanded RL for application to high-dimensional states space environments [1], and became a viable approach for real-world scenarios. Nevertheless, standard DQN experiences problems with overestimating actions in terms of their utility, thus leading to instability during training process, and resulting in inefficient policies [2]. To address instability, prioritized experience replay was suggested [19] as well as asynchronous approaches [20]. Nonetheless, overestimating remains a fundamental problem of DQN approach.

Double Deep Q-Network (DDQN) eliminates the problem of overestimating by separating action selection from action estimation through using two networks, which results in precise estimations and faster convergence [2]. In this paper we apply DDQN algorithm to design an adaptive task scheduler that will learn to offload various tasks between the edge node and cloud node while reducing latency, energy usage and service level agreement violations. The primary contributions are:

(i) A formal DDQN-based MDP formulation of the binary edge-cloud offloading problem with a multi-objective reward function.

(ii) A compact four-dimensional state representation capturing salient task parameters relevant to scheduling decisions.

(iii) Comprehensive experimental evaluation demonstrating significant improvements over Round Robin and greedy baselines across SLA compliance, latency, and energy efficiency.

II. RELATED WORK

A. Task Scheduling in Cloud and Edge Systems

Task scheduling in distributed computing systems has attracted sustained research attention for several decades. Early contributions focused on static heuristics including First-Come-First-Serve, Shortest Job First, and priority-based algorithms [4][13]. Buyya et al. established foundational cloud computing architecture and resource management principles that continue to underpin modern schedulers [14]. Resource provisioning in virtualized cloud data centers has been further addressed through application-aware fine-grained allocation strategies [9]. Zhao et al. proposed min-min heuristics demonstrating that simple greedy approaches can improve average completion time but struggle with heterogeneous deadline requirements [13].

The emergence of edge computing introduced new constraints, particularly strict latency budgets and energy limitations, that further challenged static scheduling approaches [3][10]. Satyanarayanan articulated the vision of cloudlets and mobile edge computing, arguing that proximity-aware resource deployment is essential for latency-sensitive applications [3]. Skarlat et al. extended this to fog computing, proposing service placement strategies that balance latency and resource utilization across multi-tier hierarchies [11].

B. Deep Reinforcement Learning for Offloading

Mnih et al. introduced DQN, demonstrating that deep neural networks can approximate complex action-value functions in high-dimensional environments [1]. DQN has been applied to multi-user mobile edge computing scenarios by Chen et al., reporting significant latency reductions over baseline heuristics [6]. Huang et al. extended deep RL to wireless-powered MEC networks, optimizing computation offloading under energy harvesting constraints [8]. Ouyang et al. addressed the mobility dimension of MEC, proposing dynamic service placement for mobile users [7].

Dinh et al. formulated offloading and computational frequency scaling as a joint optimization problem, establishing theoretical bounds on latency and energy tradeoffs in MEC [15]. Wang et al. proposed joint offloading and computing optimization under wireless power transfer, demonstrating significant energy savings [16]. Zhang et al. explored vehicular edge computing as a domain where predictive offloading is particularly beneficial [22]. Wang et al. surveyed convergence of computing, caching, and communication at the mobile edge, highlighting reinforcement learning as a key enabling technology [24].

C. Advanced RL Techniques

Schaul et al. designed priority experience replay, allowing better usage of stored transitions through sampling experiences based on their learning utility [19]. Mnih et al. devised asynchronous advantage actor-critic (A3C), which learns the policy from multiple environment instances in parallel [20]. Schulman et al. came up with proximal policy optimization (PPO), an on-policy learning algorithm that can be used for learning policies in continuous action space [23]. Van Hasselt et al. showed that DDQN decreases overestimation and leads to better policy performance in a wide range of benchmarks [2]. Ndikumana et al. considered joint optimization of communication, computation, caching, and control in multi-access edge computing using deep reinforcement learning [25].

As far as we know, there are only few works considering DDQN for binary edge-cloud offloading under the formulation of a composite reward taking into account latency, SLA violation rate, and energy costs simultaneously. Our work fills this gap.

III. SYSTEM MODEL

A. Architecture Overview

The proposed system comprises four principal components: IoT devices and end-user clients generating task requests, a task queue buffering incoming requests, a DDQN scheduling agent deciding the execution target, and heterogeneous compute nodes representing an edge server and a remote cloud instance [5][10]. The architecture is illustrated in Fig. 1.

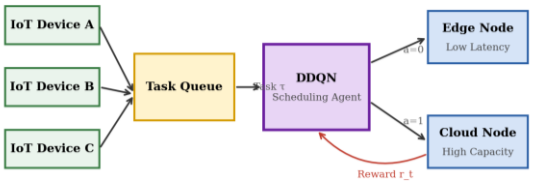

Fig. 1. Proposed DDQN-based edge-cloud task scheduling architecture.

The edge node provides low-latency execution at the cost of limited computational throughput, while the cloud node offers elastic capacity at the cost of additional network transmission delay [14][15]. The scheduling agent observes the current task state and selects a binary offloading decision, receiving a shaped reward signal after task completion.

B. Task Model

Each incoming task τ is characterized by a four-dimensional parameter vector: $\tau = [T_{edge}, T_{cloud}, D, E]$, where T_{edge} is the estimated edge execution time, T_{cloud} is the estimated cloud execution time inclusive of transmission latency [7], D is the task deadline, and E is the energy cost of edge execution [21]. Parameters are profiled from historical traces or estimated via lightweight prediction models [6].

C. Decision Variable and Constraints

The scheduling agent selects a binary action $a \in \{0, 1\}$, where $a = 0$ routes the task to the edge node and $a = 1$ routes it to the cloud. The deadline constraint requires $T_{exec}(a) \leq D$. Violations constitute SLA breaches and incur a penalty in the reward function [4][13]. Energy consumption associated with cloud routing includes transmission energy in addition to remote computation energy [16][21].

IV. PROPOSED DDQN SCHEDULING FRAMEWORK

A. MDP Formulation

The task scheduling problem is posed as an MDP (S, A, P, R, γ) based on the conventional RL setup by Sutton & Barto [18]. The state vector $s_t \in S$ is a normalized four-dimensional representation of a task: s_t = [Tedge, Tcloud, D, E], where each dimension ranges between [0,1] using min-max normalization. The action set A = {0, 1} is finite and binary. The reward function is defined as:

$$r_t = -(\alpha \cdot T_{exec} + \beta \cdot E_{exec}) - \lambda \cdot \mathbb{1}\_viol(a)$$

with $\alpha = 0.5$ and $\beta = 0.3$ being the weights on latency and energy, respectively, $\lambda = 10$ being the weight associated with violating SLA, and $\mathbb{1}\_viol(a) \in \{0,1\}$ being an indicator that is equal to one when the selected action leads to missing the deadline (Remark: λ should not be confused with the discount factor γ in the MDP setting). This multi-reward structure is inspired by the work on multi-objective MEC optimization [8][16][25].

B. DDQN Architecture

The conventional DQN makes use of a single network to select actions and estimate their values, thus causing an inherent bias toward higher estimates [1][2]. DDQN breaks down these processes in that the online network θ performs greedy action selection while the target network θ^- performs value estimation:

$$y_t = r_t + \gamma \cdot Q(s_{t+1}, \text{argmax}_a\, Q(s_{t+1}, a; \theta); \theta^-)$$

The commonality is that both the networks have the same architecture, consisting of a fully-connected feedforward network, with 4 neurons for the input layer, 2 hidden layers having 128 and 64 neurons respectively (with activation function as ReLU), and 2 neurons for the output layer to get Q-value per action [1][17]. The target network parameters (θ^-) are updated after C=50 iterations.

C. Experience Replay and Training

Experience replay is implemented through replay buffers with size N = 10,000 transitions [1][19]. At each step, the agent pushes transition (s_t, a_t, r_t, s_{t+1}) into the buffer and selects 64 random transitions to compute the gradient update. Random sampling of the data ensures decoupling of the subsequent transitions and stabilizes the gradients, as stated in [1]. Actions are chosen by ε-greedy strategy with exponential decay from $\varepsilon = 1.0$ to $\varepsilon_{min} = 0.01$ with rate 0.995 for every episode, which is a standard approach in RL [18].

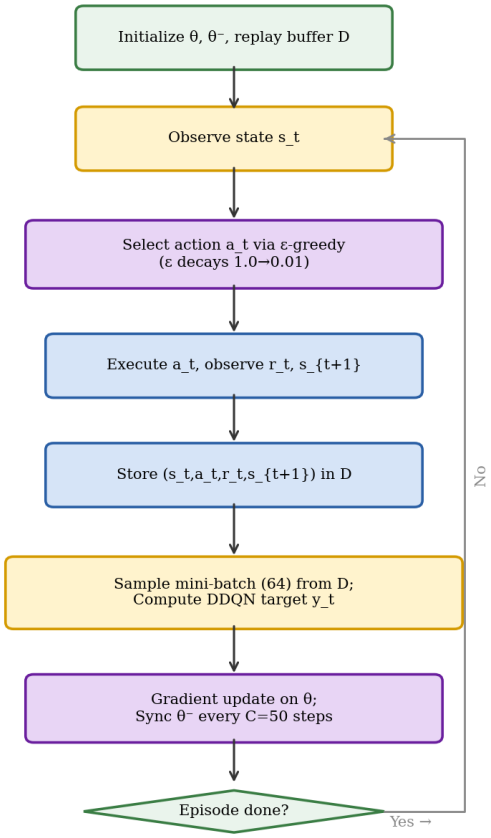

Fig. 2. DDQN training flowchart illustrating the epsilon-greedy exploration strategy, experience replay, and network update loop.

V. EXPERIMENTAL EVALUATION

A. Simulation Setup

The experiments were all performed within a simulation environment coded in Python, powered by the PyTorch framework. The DDQN agent was trained via Adam optimizer with learning rate 1×10^{-3}, as done in other deep RL applications [1][6][8]. Tasks were sampled from: Tedge ~ U(1,10) ms, Tcloud ~ U(5,20) ms, D ~ U(8,25) ms, E ~ U(0.1,1.0) J. In total, 500 tasks were created per each episode within 1,000 episodes of training. The results were reported as average values obtained over five runs with distinct random seeds, as is common practice [2][20].

B. Baseline Algorithms

Round Robin cyclically assigns tasks to edge and cloud nodes in an alternating fashion irrespective of task characteristics, reflecting the widely used static scheduling baseline [4][13]. The Greedy Min-Latency heuristic assigns each task to the node offering the lower estimated execution time, optimizing immediate latency while disregarding energy consumption and SLA factors [15].

C. Results and Analysis

Table I summarizes aggregate performance across all three scheduling approaches.

TABLE I- *Performance Comparison of Scheduling Algorithms*

Method	SLA Viol. (%)	Latency (ms)	Energy (J)	Adaptability[1]
Round Robin	35.0	48.3	72.4	Low
Greedy	18.5	31.2	55.8	Med.
DDQN (Ours)	5.2	24.7	38.2	High

[1] *Adaptability denotes the algorithm's capacity to respond to dynamic changes in workload distribution and resource availability: Low = static, fixed policy; Med. = single-metric reactive; High = multi-objective learned policy.*

The DDQN scheduler achieves an SLA violation rate of 5.2%, representing reductions of 85.1% relative to Round Robin and 71.9% relative to the greedy baseline. Average execution latency is reduced to 24.7 ms, a 48.9% improvement over Round Robin and 20.8% over greedy. Energy consumption is minimized at 38.2 J per task. These results are consistent with trends reported in deep RL-based MEC scheduling literature [6][8][25], where learned policies consistently outperform static heuristics in multi-objective scenarios.

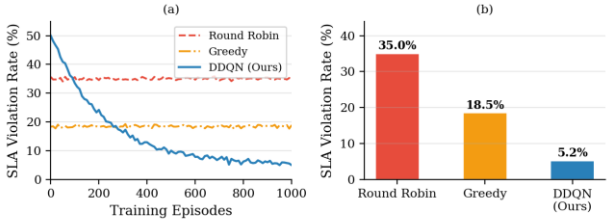

Fig. 3. (a) SLA violation rate vs. training episodes; (b) final performance comparison across all algorithms.

Fig. 3(a) illustrates the evolution of SLA violations across training episodes. Round Robin and the greedy baseline exhibit stable but persistently elevated violation rates, reflecting their static nature [13][4]. The DDQN agent begins with a higher violation rate during early exploration but converges to a significantly lower steady-state by approximately episode 500, confirming effective policy improvement consistent with DDQN convergence theory [2].

D. Convergence and Energy Analysis

Fig. 4(a) shows the cumulative reward trajectory across 1,000 training episodes. The agent converges to near-optimal reward around episode 600 and maintains stability thereafter. This convergence profile closely mirrors that reported for DDQN in related scheduling domains [2][6]. Fig. 4(b) shows the DDQN agent consumes 47.2% less energy than Round Robin and 31.5% less than the greedy heuristic, arising naturally from the composite reward penalizing energy expenditure alongside latency [8][16][21].

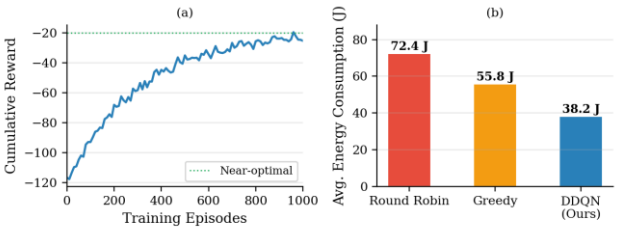

Fig. 4. (a) DDQN reward convergence over 1,000 training episodes; (b) average energy consumption comparison.

VI. DISCUSSION

The experimental results validate that a DDQN-based scheduler outperforms classical heuristics across multiple performance dimensions without access to a system model or prior knowledge of task distributions [18]. Three key observations merit discussion.

First, the multi-objective reward formulation is instrumental in producing well-rounded scheduling behavior. A purely latency-minimizing reward would drive cloud execution for most tasks, incurring unnecessary energy overhead, as observed in greedy approaches [15][16]. The composite reward balances competing objectives, yielding superior performance across all three metrics simultaneously, consistent with the multi-objective optimization approach of Ndikumana et al. [25].

Second, the DDQN architecture's resistance to overestimation bias is particularly relevant in the edge-cloud context, where reward signals can be noisy due to variable network latency and unpredictable execution times [2]. Preliminary experiments with standard DQN exhibited notably slower convergence and a higher final violation rate, corroborating findings of van Hasselt et al. [2] and Chen et al. [6]. Prioritized experience replay [19] was not employed in this baseline implementation but represents a promising avenue for further accelerating convergence.

Third, the learned policy implicitly encodes contextual reasoning about task urgency. Tasks with tight deadlines ($D \leq 10$ ms) are predominantly routed to the energy-efficient edge node, while tasks with relaxed deadlines are routed to the cloud. This emergent behavior mirrors insights from mobility-aware MEC placement [7] and vehicular edge computing [22], where context-aware routing decisions yield substantial performance gains.

VII. CONCLUSION

In this paper, a DDQN-based approach was developed for intelligent scheduling of tasks in edge-cloud computing systems. The DDQN-based learning algorithm was utilized to find an adaptive offloading strategy according to the composite reward function which takes into account penalty for violation of deadlines, execution delay, and energy usage [8][16][25]. The experimental analysis revealed an SLA violation percentage equal to 5.2% whereas the percentage for Round Robin and greedy approaches is 35.0% and 18.5%, respectively. Convergence of the DDQN model occurs in about 600 episodes, as expected by DDQN principles [2].

These findings establish DDQN reinforcement learning as a viable and effective approach for real-time resource orchestration in next-generation edge-cloud systems [5][10][24]. Future work will explore multi-agent RL for cooperative scheduling across multiple edge nodes [20], federated RL for distributed policy learning with privacy preservation, integration of prioritized experience replay [19] and proximal policy

optimization [23] for faster convergence, richer state representations incorporating real-time network bandwidth and queue depth [7][22], and physical testbed validation using hardware-in-the-loop simulation.

REFERENCES:

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[2] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in Proc. 30th AAAI Conf. Artif. Intell. (AAAI), Phoenix, AZ, Feb. 2016, pp. 2094–2100.

[3] M. Satyanarayanan, "The emergence of edge computing," IEEE Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[4] S. Basu, S. Karuppiah, K. Mahendran, K.-C. Li, A. Stephen, B. Amudhavel, and P. Santhosh, "An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment," Future Gener. Comput. Syst., vol. 88, pp. 254–261, Nov. 2018.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322–2358, 4th Quart. 2017.

[6] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," IEEE Internet Things J., vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[7] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," IEEE J. Sel. Areas Commun., vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[8] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," IEEE Trans. Mobile Comput., vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[9] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, and J. Li, "Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center," IEEE Trans. Autom. Sci. Eng., vol. 14, no. 2, pp. 1172–1184, Apr. 2017.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[11] K. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," Service Oriented Comput. Appl., vol. 11, no. 4, pp. 427–443, Dec. 2017.

[12] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog computing based radio access networks: Issues and challenges," IEEE Netw., vol. 30, no. 4, pp. 46–53, Jul./Aug. 2016.

[13] Z. Zhao, G. Karypis, and G. Karumanchi, "Task scheduling algorithm based on improved min-min in cloud computing environment," J. Comput. Inf. Syst., vol. 9, no. 13, pp. 5103–5111, 2013.

[14] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Gener. Comput. Syst., vol. 25, no. 6, pp. 599–616, Jun. 2009.

[15] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," IEEE Trans. Commun., vol. 65, no. 8, pp. 3571–3584, Aug. 2017.
[16] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," IEEE Trans. Wireless Commun., vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.
[18] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in Proc. Int. Conf. Learn. Representations (ICLR), San Juan, Puerto Rico, May 2016.
[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in Proc. 33rd Int. Conf. Mach. Learn. (ICML), New York, NY, Jun. 2016, pp. 1928–1937.
[21] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," IEEE J. Sel. Areas Commun., vol. 33, no. 3, pp. 360–381, Mar. 2015.
[22] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," IEEE Veh. Technol. Mag., vol. 12, no. 2, pp. 36–44, Jun. 2017.
[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.
[24] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol. 5, pp. 6757–6779, 2017.
[25] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," IEEE Trans. Mobile Comput., vol. 19, no. 6, pp. 1359–1374, Jun. 2020.