Epsilon-Greedy Applied to LODCO for Dynamic Computation Offloading in Multi-Server Mobile Edge Computing Systems

Xuanye Zeng
Electrical and Electronics Engineering Department
University College London, London, UK
calebzeng02@gmail.com

***Abstract*—Mobile Edge Computing (MEC) addresses computational limitations of mobile devices by offloading tasks to nearby edge servers. However, existing offloading algorithms struggle to balance exploration and exploitation in dynamic multi-server environments. This paper used epsilon-greedy LODCO, integrating controlled exploration with Lyapunov optimization to improve multi-server offloading decisions while maintaining theoretical queue stability guarantees. Unlike naive random exploration, this approach applies epsilon-greedy selection specifically at the server allocation phase, preserving optimal local/drop decisions. Comprehensive experiments under moderate overload conditions (~150% expected load) demonstrate that epsilon-greedy LODCO ($\varepsilon = 0.8$) achieves 2.3–2.5% execution delay reduction under severe resource pressure, with load-dependent optimal values: moderate load favors higher exploration ($\varepsilon \approx 1.0$), while severe overload benefits from structured exploration ($\varepsilon = 0.7 - 0.8$). These findings provide practical guidance for MEC system design where intelligent exploration becomes increasingly valuable as resource contention intensifies.**

Keywords—Mobile Edge Computing, Computation Offloading, Lyapunov Optimization, Epsilon-Greedy, Resource Allocation

I. INTRODUCTION

The proliferation of mobile devices and computation-intensive applications has created unprecedented demands on mobile computational resources. Mobile Edge Computing (MEC) addresses this challenge by deploying computational resources at the network edge, achieving 20–50% latency reduction compared to traditional cloud offloading [1]. However, efficient computation offloading presents fundamental challenges: (1) dynamic and uncertain wireless channel conditions, (2) stochastic task arrivals with varying computational demands, (3) limited and time-varying energy availability in energy harvesting devices, and (4) competition for limited edge server resources among multiple users. Effective offloading algorithms must jointly optimize execution mode selection, resource allocation, and energy management while adapting to system dynamics without requiring complete knowledge of future system states.

Existing approaches tackle these challenges through various frameworks. Lyapunov optimization-based methods, such as the seminal LODCO algorithm [1] and its stochastic extensions [2], offer theoretical guarantees by transforming long-term stochastic optimization into per-time-slot deterministic problems, achieving provable queue stability and O(1/V) optimality gap. However, LODCO employs pure greedy selection, potentially leading to server underutilization in multi-server scenarios where exploration could discover better strategies. Recent deep reinforcement learning (DRL) extensions [3]–[5] achieve improved performance but require substantial training data and lack theoretical stability guarantees. Greedy heuristic approaches [6] offer computational efficiency but are fundamentally limited by local optimality, especially critical in multi-server scenarios where server selection significantly impacts performance.

While the epsilon-greedy strategy is widely used in reinforcement learning for exploration-exploitation balance, its systematic integration with Lyapunov-optimized computation offloading remains largely unexplored. Zhao et al. [7] extended LODCO to multi-server scenarios and briefly mentioned epsilon-greedy as a potential mechanism, but their work primarily focuses on pure greedy server selection. Furthermore, recent surveys on MEC resource scheduling [8]–[10] highlight the lack of lightweight online algorithms that can adapt to severe overload without the computational overhead of DRL. Specifically, prior works lack: (1) systematic characterization of epsilon sensitivity; (2) multi-dimensional evaluation across varying system conditions; (3) rigorous convergence analysis; and (4) comparisons against random baselines to isolate the value of intelligent exploration. These gaps motivate our research.

To bridge these gaps, this paper used Epsilon-Greedy LODCO, a theoretically grounded framework that integrates controlled exploration into Lyapunov optimization while maintaining queue stability. A key innovation lies in decoupling the decision process: we apply epsilon-greedy strategies strictly during the server allocation phase, thereby preserving the optimality of local execution and dropping decisions found in standard Lyapunov approaches. We systematically characterize the exploration-exploitation trade-off, identifying critical load-dependent behaviors where exploration becomes essential as resource contention intensifies. Unlike data-intensive DRL approaches, this method offers a low-complexity, training-free solution that efficiently discovers underutilized resources in dynamic, multi-server MEC environments.

II. METHODOLOGY

A. System Model and Execution Modes

We consider a multi-user multi-server MEC system consisting of N mobile devices and M edge servers operating

over discrete time slots t. Each mobile device i is equipped with an energy harvesting battery with capacity $B_i(t)$. At the beginning of slot t, the device harvests energy $e_i(t) \in [E_{H,max}]$ and generates a computation task with probability ρ. Each task is characterized by input size L bits and computational workload W cycles. The battery dynamics evolve as $B_i(t+1) = B_i(t) - E_i(t) + e_i(t)$, where $E_i(t)$ is the energy consumption.

Upon task arrival, the system selects an execution mode $a_i(t) \in \{Local, Offload, Drop\}$.

- Local Execution: The device uses DVFS with frequency $f_i(t) \in [0, f_{max}]$. The execution delay is $T_{loc} = W/f_i(t)$ and energy consumption is $E_{loc} = kWf_i(t)^2$.
- Server Offloading: The device transmits data to a server j with rate $R_{ij}(t)$, determined by channel gain $h_{ij}(t)$ and transmit power $p_i(t)$. The cost includes transmission delay $T_{off} = L/R_{ij}(t)$ and energy $E_{off} = p_i(t)T_{off}$. Each server j has a strict capacity constraint, handling at most $max_{connects}$ concurrent tasks.
- Task Dropping: If execution is infeasible due to energy or capacity constraints, the task is dropped, incurring a fixed penalty delay $T_{drop} = \phi$ and zero energy cost.

B. Optimization Objective

Our primary objective is to minimize the long-term average execution cost (weighted delay and penalty) across all devices while ensuring energy queue stability and respecting server capacity limits. The optimization problem is formulated as:

$$min_{a,f,p} \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N} E[C_i(t)] \tag{1}$$

$$\text{s.t. C1: } \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E[E_i(t)] \le \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E[e_i(t)], \forall i$$

$$\text{C2: } \sum_{i=1}^{N} I(a_i(t) = Server_j) \le max_{connects}, \forall j$$

where $C_i(t)$ represents the realized cost (T_{loc}, T_{off}, or ϕ). Constraint C1 guarantees battery stability [11], and C2 enforces server capacity.

To solve this stochastic problem without future knowledge, we employ the LODCO framework [1]. By defining a virtual battery queue $\hat{B}_i(t) = B_i(t) - \theta$, LODCO transforms the long-term problem into a deterministic per-slot minimization of the "Drift-plus-Penalty" function [12]:

$$\text{Minimize} \qquad J_i(t) = -\hat{B}_i(t) \cdot E_i(t) + V \cdot C_i(t) \tag{2}$$

Here, V is a control parameter balancing energy conservation and delay minimization. For each device i, LODCO calculates the minimum costs for local execution (J_m), server offloading to server j ($J_s(i,j)$), and dropping (J_d).

C. Optimization Algorithms

While LODCO provides the theoretical basis for energy-delay trade-offs (Eq. 2), the specific strategy for server allocation significantly impacts performance in multi-server environments with capacity constraints (C2). We evaluate the following optimization strategies:

1) Greedy LODCO Strategy: This baseline approach, widely used in prior work [7], selects the execution mode with the absolute minimum J value: mode = arg $\min\{J_m, min_j J_s(i,j), J_d\}$. In multi-server scenarios, it prioritizes device-server pairs with the lowest J_s values. However, this greedy nature can lead to load imbalance, where optimal servers become congested while others remain idle.

2) Random Offloading Strategy: To isolate the value of intelligent selection, this baseline uses LODCO for local/drop decisions but selects a server uniformly at random from the available set for offloading. It provides a lower bound on exploration efficiency but lacks the guidance of channel state information (CSI).

3) Epsilon-Greedy LODCO Strategy: We integrate the ε-greedy mechanism into the LODCO allocation phase to balance exploration and exploitation.

- Exploitation ($1-\varepsilon$): With probability $1-\varepsilon$, the algorithm selects the server j^* that minimizes $J_s(i,j)$, exploiting current CSI to minimize immediate cost.
- Exploration (ε): With probability ε, the algorithm randomly selects an available server. This proactively distributes load and prevents the system from getting stuck in local optima caused by temporary congestion on best servers.

Critically, this exploration is applied only to the server selection phase; optimal local execution and dropping decisions derived from Eq. (2) are preserved to maintain system stability.

D. Performance Metrics

To comprehensively evaluate the algorithmic performance, we measure seven key metrics: (1) Average Execution Cost (C_{avg}), the primary objective function (delay/penalty); (2) Average Energy Consumption (E_{avg}), to analyze the energy-delay trade-off; (3) Task Drop Rate (R_{drop}), reflecting service reliability; (4) Virtual Battery Queue Backlog ($\hat{B}_{avg}$), verifying the theoretical queue stability constraints; (5) MEC Server Utilization (U_{server}), measuring resource usage efficiency; (6) Execution Mode Distribution, revealing the strategic behavior (Local vs. Server vs. Drop) under different loads; and (7) Convergence Time, evaluating how quickly the algorithm adapts to system dynamics.

III. Results and Discussion

A. Experimental Setup

We simulate a MEC system with $N = 28$ mobile devices and $M = 8$ edge servers over $T = 1200$ time slots. To create realistic resource contention scenarios, we configure: $max_{connects} = 2$ (server capacity per time slot, yielding total capacity of 16 concurrent connections), baseline arrival rate $\rho = 0.85$ (expected 23.8 tasks per slot), achieving approximately 150% expected load. Other key parameters include: $\tau = 0.002s$, $f_{max} = 1.5GHz$, $k = 10^{-28}$, $p_{tx,max} = 1W$, $E_{\max} = 2mJ$, $E_{H,max} = 80\mu J$, $L = 1000\ bits$, $X = 737.5\ cycles/bit$, $\omega = 1MHz$, $\sigma = 10^{-13}W$, $V = 10^{-5}$, $\phi = 0.008s$, and channel gain $h_{ij} \sim g_0(d/d_0)^{-2}$ with $g_0 = 10^{-4}$ and $d_0 = 1m$.

To ensure fair comparison, we configure Original LODCO with the same $max_{connects} = 2$ capacity constraint as each multi-server. Server utilization is consistently defined as (number of tasks offloaded to server(s)) / (T × $max_{connects}$× number of servers) across all algorithms.

We deliberately create moderate overload conditions (expected load ~150% of capacity) to differentiate algorithms under realistic stress. This addresses the fundamental challenges: (1) Challenge 1 (dynamic wireless conditions) and Challenge 2 (stochastic task arrivals) cause peak-hour overload in urban MEC; (2) Challenge 3 (limited energy availability) forces intelligent energy-delay trade-offs; (3) Challenge 4 (server resource competition) requires strategic allocation when greedy selection encounters full servers.

Five experiments target specific challenges: (1) Distance Impact ($d \in \{10,30,50,70,100\}m$) tests Challenge 1 via propagation path loss; (2) Device Scalability ($N \in \{10,20,30,40\}$) intensifies Challenge 4 through increasing contention; (3) Epsilon Sensitivity ($\varepsilon \in \{0,0.2,0.5,0.6,0.7,0.8,1.0\}$) characterizes exploration-exploitation balance; (4) Task Arrival Rate ($\rho \in \{0.3,0.6,0.9\}$) tests Challenge 2 via varying traffic; (5) Convergence Behavior ($T = 1200$ slots) demonstrates adaptive learning under time-varying conditions.

Statistical Analysis. Each configuration runs 5 independent trials with different random seeds. Results report mean values with 95% confidence intervals ($CI = 1.96 \times std / \sqrt{n}$). All reported improvements achieve $p < 0.05$ significance (two-sample t-test).

B. Experimental Results and Analysis

1) Impact of Distance: This experiment evaluates the system's robustness against dynamic channel conditions (Challenge 1), as shown in Fig. 1.

Cost & Energy (Fig. 1a, b): As the distance increases from 10m to 100m, the path loss significantly degrades channel quality. Consequently, the average execution cost for epsilon-greedy LODCO increases from 0.722ms to 1.404ms. However, energy consumption remains relatively stable (~0.042 mJ). This stability aligns with our optimization objective: to minimize delay while strictly enforcing energy harvesting constraints. Utilization & Mode Shift (Fig. 1c): A key observation is the sharp drop in server utilization at d = 100m (falling to 17.7%). As wireless transmission becomes energetically expensive and slow, the LODCO framework adaptively shifts decisions from "Server Offloading" to "Local Execution" (indicated by the cost increase matching local processing capabilities). Stability (Fig. 1d): Despite drastic channel variations, the virtual battery queue $\hat{B}_{avg}$ remains negative and bounded ($< -0.5\,mJ$) across all distances, empirically verifying the queue stability constraint (C1) defined in Section II.

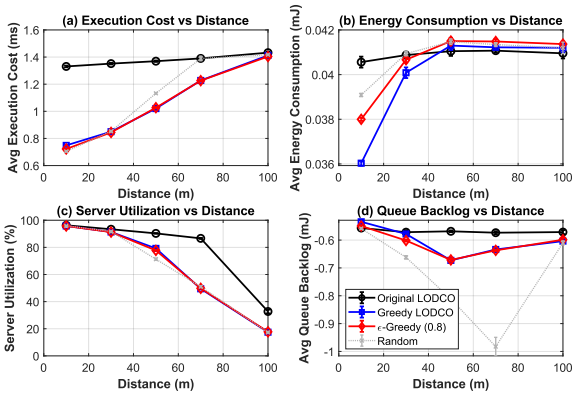

Fig. 1. Performance versus distance between devices and servers. Four subplots showing (a) execution cost, (b) energy consumption, (c) server utilization, and (d) queue stability across varying device-server distances. Error bars represent 95% confidence intervals.

2) Device Scalability and Resource Contention: This experiment (Fig. 2) is critical as it directly tests the algorithm under severe resource competition (Challenge 4).

Superiority under Pressure (Fig. 2a): At low density ($N = 10$), all algorithms perform similarly due to abundant resources. However, as contention intensifies ($N = 30 \rightarrow 40$), the advantage of ε-greedy becomes evident. At $N = 40$ (213% load), it achieves a 2.5% reduction in cost compared to Greedy LODCO. This confirms that structured exploration successfully identifies available server slots that pure greedy approaches miss due to local optima. Load Balancing (Fig. 2c): The drop rate for ε-greedy remains competitive with Greedy LODCO, indicating that the exploration step effectively discovers resources without leading to excessive invalid allocations.

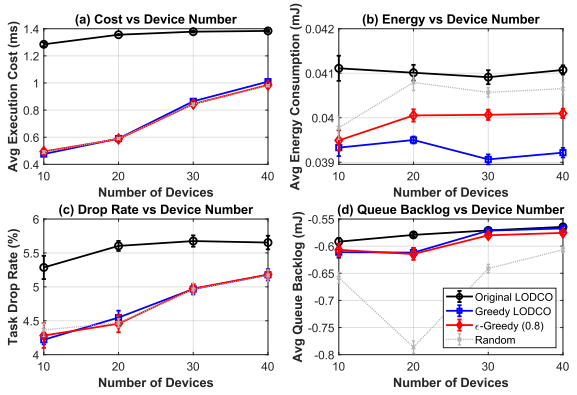

Fig. 2. Performance scalability with device number $N \in \{10,20,30,40\}$ covering moderate to severe contention. Four subplots showing (a) execution cost, (b) energy consumption, (c) task drop rate, and (d) queue backlog. Error bars represent 95% confidence intervals.

3) Epsilon Sensitivity and Energy-Delay Tradeoff: This section characterizes the exploration-exploitation tradeoff using Fig. 3.

Optimal Range Identification (Fig. 3a): We observe a convex-like pattern where moderate exploration ($\varepsilon \approx 0.6 - 0.8$) yields stable low costs. While pure Random ($\varepsilon = 1.0$) performs slightly better at this specific moderate load ($N = 28$) due to natural load dispersion, our Scalability results (Fig. 2) confirm that $\varepsilon = 0.8$ offers superior robustness in high-

contention scenarios ($N = 40$). Thus, $\varepsilon = 0.8$ strikes the best balance between Random's dispersion and Greedy's precision.

Pareto Front Analysis (Fig. 3b): To visualize the multi-objective impact, we plot the Energy-Delay Pareto front where marker colors represent the ε value. The color gradient transitions from Blue (low ε, conservative) to Red (high ε, aggressive). As the color shifts towards red ($\varepsilon \to 1.0$), the points move in a right-downward trend, energy is increasing(but not changing much) while delay is decreasing. This visual confirmation suggests that increasing exploration (redder points) helps the system break out of high-cost local optima associated with greedy congestion (blue points).

Service Reliability (Fig. 3c): The drop rate remains stable (4.8% − 5.0%) across all ε values, proving that introducing randomness into the allocation phase does not compromise the system's reliability guarantees.

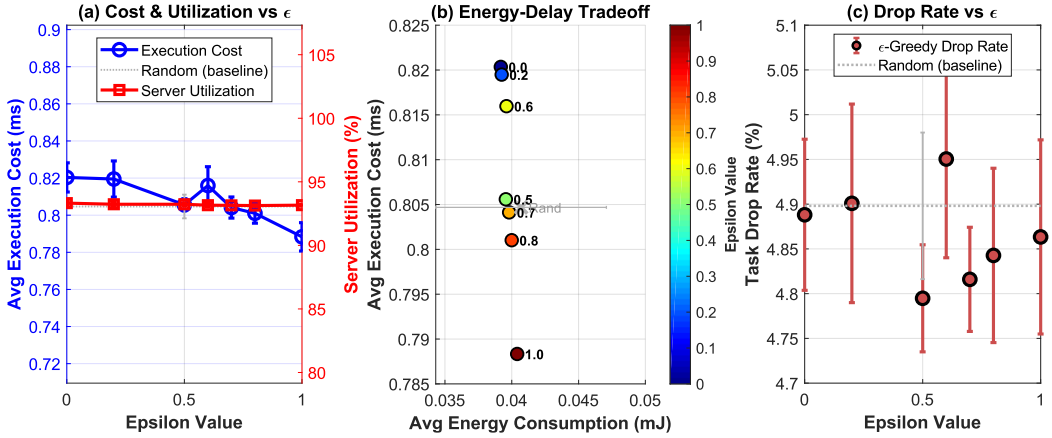

Fig. 3. Sensitivity to epsilon parameter. Three subplots showing (a) cost and server utilization vs epsilon, (b) energy-delay Pareto front with epsilon values color-coded, (c) task drop rate vs epsilon. Error bars represent 95% confidence intervals.

4) Impact of Task Arrival Rate: Fig. 4 analyzes system behavior under varying traffic loads.

Cost vs. Energy Inverse Relation (Fig. 4a, b): As the arrival rate ρ increases from 0.3 to 0.9, execution cost rises due to congestion, but average energy consumption decreases (from 0.076 mJ to 0.038 mJ). Mechanism (Fig. 4d): The Mode Distribution plot explains this phenomenon. At low load ($\rho = 0.3$), 91.3% of tasks are offloaded, trading higher transmission energy for lower delay. At high load ($\rho = 0.9$), capacity constraints force 35% of tasks to revert to Local Execution, which consumes less energy but incurs higher delay.

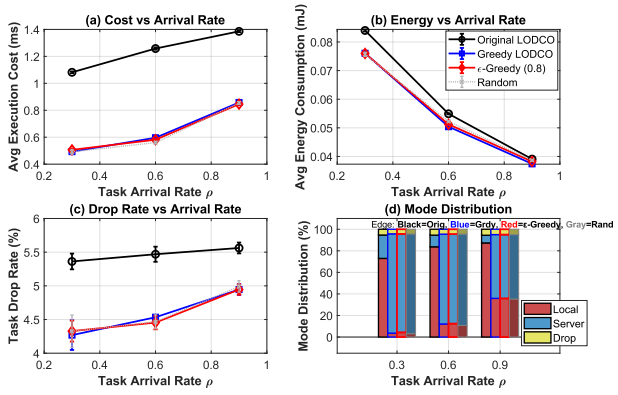

Fig. 4. Performance under varying task arrival rates. Four subplots showing (a) execution cost, (b) energy consumption, (c) task drop rate, and (d) execution mode distribution at $\rho = \{0.3, 0.6, 0.9\}$ for all algorithms. Error bars represent 95% confidence intervals.

5) Convergence Analysis: Fig. 5 demonstrates the learning curve over $T = 1200$ slots. The ε-greedy algorithm ($\varepsilon = 0.8$, red line) converges to a stable cost of 0.829ms. Compared to the pure Greedy approach ($\varepsilon = 0$, black line), the introduction of exploration does not cause instability or oscillation. Instead, it achieves a slightly faster settling time by quickly identifying underutilized servers in the early phase. Notably, the performance curve becomes flat after approximately $T = 800$ and remains constant through $T = 1200$. This indicates that the system has reached a stochastic steady state where the virtual battery queues have stabilized around the target energy level θ. Once these queues stabilize, the Lyapunov drift becomes zero on average, causing the algorithm to consistently select execution modes that maintain the optimal energy-delay balance. Consequently, the continued increase in time slots (e.g., past $T = 1000$) has no further impact on the average cost, validating the robustness of the proposed solution in long-term operations.

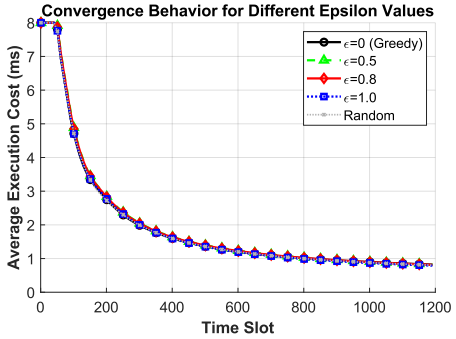

Fig. 5. Convergence behavior over $T = 1200$ time slots. Average execution cost trajectory for $\varepsilon = \{0, 0.5, 0.8, 1.0\}$ plus Random baseline, with detected convergence points marked. Demonstrates that moderate epsilon (0.5–0.8) achieves the fastest and most stable convergence.

C. Performance Summary

Table I summarizes the performance at the baseline configuration ($d = 50m$, $N = 28$, $\rho = 0.85$). This scenario represents a "moderate overload" state (150% expected load) where resources are competitive but not yet exhausted. Multi-Server Advantage: All multi-server algorithms (Greedy, ε-Greedy, Random) significantly outperform the single-server Original LODCO (17.3–25.6% cost reduction), confirming the fundamental value of spatial diversity in MEC.

TABLE I. PERFORMANCE COMPARISON UNDER BASELINE CONFIGURATION (N=28)

Algorithm	Cost (ms)	Energy (mJ)	Util (%)	Drop (%)	Queue (mJ)
Original LODCO	1.370	0.041	90.3	5.6	-1.6
Greedy LODCO	1.020	0.041	79.2	5.0	-1.7
ε-Greedy (0.5)	1.019	0.041	78.7	4.9	-1.7
ε-Greedy (0.8)	1.027	0.042	77.8	5.0	-1.7
Random Offloading	1.133	0.041	71.3	5.2	-1.7

Ideally, at this moderate load ($N = 28$), Greedy performs slightly better (1.020ms) than ε-Greedy (1.027ms). This reveals a crucial insight: exploration incurs a minor overhead when resources are sufficient. However, this overhead is negligible (∼0.7%), acting as an acceptable "insurance premium." As shown in Fig. 2, paying this small premium at $N = 28$ enables the system to handle severe overload ($N = 40$) with significant gains (2.5%), demonstrating the algorithm's superior robustness across varying load regimes.

Random offloading performs 10.3% worse than ε-Greedy, proving that structured exploration (retaining optimal local/drop decisions) is essential. Naive randomization fails to respect the energy-delay trade-offs inherent in the Lyapunov framework.

IV. CONCLUSION

This paper used Epsilon-Greedy LODCO, a lightweight online algorithm that integrates controlled exploration with Lyapunov optimization for multi-server MEC systems. By applying epsilon-greedy selection specifically during the server allocation phase, we preserve the theoretical stability guarantees of Lyapunov optimization while mitigating the local optimality trap of greedy heuristics. Extensive simulations demonstrate a load-dependent performance profile. Under moderate load ($N = 28$), the algorithm maintains near-optimal performance with negligible exploration overhead ($< 1\%$). Crucially, under severe resource pressure ($N = 40$), structured exploration ($\varepsilon \approx 0.8$) unlocks significant benefits, achieving up to 2.5% reduction in execution delay compared to greedy baselines. This confirms that intelligent exploration becomes a dominant factor in system performance as resource contention intensifies. Future work will focus on: (1) developing adaptive epsilon scheduling mechanisms driven by queue backlogs; (2) extending the framework to support partial offloading and dependent task graphs; and (3) validating the algorithm on real-world MEC testbeds to assess the impact of signaling latency. As edge networks evolve towards ultra-dense 6G deployments, such lightweight, robust, and theoretically grounded algorithms will be pivotal in unlocking the full potential of ubiquitous edge intelligence.

REFERENCES

[1] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," IEEE J. Sel. Areas Commun., vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[2] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," IEEE Trans. Wireless Commun., vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[3] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," IEEE Trans. Wireless Commun., vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[4] H. Huang, J. Yang, and Y. Li, "Deep learning-based dynamic computation task offloading for mobile edge computing networks," Sensors, vol. 22, no. 11, article 4088, May 2022.

[5] K. Fan and Y. Cai, "A deep reinforcement approach for computation offloading in MEC dynamic networks," EURASIP J. Adv. Signal Process., vol. 2024, article 21, 2024.

[6] L. Guo et al., "HAGP: A heuristic algorithm based on greedy policy for task offloading with reliability of MDs in MEC of the industrial Internet," Sensors, vol. 21, no. 10, article 3513, May 2021.

[7] H. Zhao et al., "QoE aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices," in Proc. IEEE Int. Conf. Ubiquitous Intell. Comput. (UIC), 2018, pp. 671–678.

[8] X. Wang, Z. Ning, and L. Wang, "Computation offloading in mobile edge computing networks: A survey," J. Netw. Comput. Appl., vol. 202, article 103366, Jun. 2022.

[9] M. Ismail et al., "A survey on resource scheduling approaches in multiaccess edge computing environment: A deep reinforcement learning study," Cluster Comput., 2024.

[10] M. R. Habibi, S. B. H. Shah, et al., "A survey of energy optimization approaches for computational task offloading and resource allocation in MEC networks," Electronics, vol. 12, no. 17, article 3548, Aug. 2023.

[11] M. J. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems. Morgan & Claypool, 2010.

[12] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," IEEE Trans. Autom. Control, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.