Technical Report

Reward-Oriented Task Offloading Under Limited Edge Server Power for Multiaccess Edge Computing

Submitted by

Parthu Reddy, CB.SC.U4CSE23639
Pratyush Yadav, CB.SC.U4CSE23641
Ravindran, CB.SC.U4CSE23647
Adarsh K.G, CB.SC.U4CSE23658

in partial fulfilment of the requirements for the course of
23CSE362 - EDGE COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
AMRITA SCHOOL OF COMPUTING
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112

List of Tables

Table 1:

TABLE I
SUMMARY OF THE LITERATURE ON THE HANDLING OF POWER AND COMPUTATION BUDGET CONSTRAINTS UNDER MEC

Ref.	Scheme objectives	Solutions/techniques
[19]	Edge server deployment to minimize energy consumption subject to distance constraints between BSs and ESs	Algorithm for minimizing the number of active servers based on particle swarm optimization
[20]	Reducing the total energy consumption of edge servers	Turning off edge servers which are not processing workload
[48]	Minimizing the average delay cost subject to a long-term energy budget	Selective BS sleep determination algorithm based on Lyapunov optimization
[49]	Minimizing the number of active servers subject to users' stochastic delay distribution constraints	Distributed selective server activation algorithm based on the Theory of Minority Games
[50]	Minimizing energy consumption subject to server capacity budget in small-cell BSs	Dynamic server switching algorithm in which under-utilized edge servers are selectively turned off
[52]	Minimizing maximum task execution time under storage and computational budget constraints	Greedy heuristic algorithm based on the cost and time of each task execution
[53]	Minimizing response latency subject to cost constraints	Multi-slot service deployment algorithm based on Lyapunov optimization
[54]	Minimizing application execution delay subject to resource constraints	Edge resource rental algorithm based on Multi-armed Bandit optimization
[55]	Minimizing task execution time subject to task dependency constraints	Greedy offloading algorithm based on edge-cloud or edge-edge cooperation
[56]	Maximizing profit subject to edge server capacity and latency constraints	Deployment of network services to edge systems to match matching the network to service providers
[57]	Minimizing task execution delay subject to the energy constraints of users	User mobility management algorithm based on Lyapunov optimization
[58]	Minimizing delay costs subject to a long-term energy budget in small-cell base stations	Online algorithm to offload tasks to peer base stations based on Lyapunov optimization
[59]	Maximizing the total QoS of the system while managing the expected risk of exceeding the energy budget	Learning-based sample average approximation based on historical data

Table 2:

TABLE II
NOTATIONS USED IN THIS ARTICLE

Notations	Meaning
N^{ES}	Number of ESs
$C_j^{\max}$	Maximum processing capacity of ES j compared to the ES with the highest processing capacity
I^{high}	Index of the ES that has the highest processing capacity
U_i	Normalized usage of task τ_i
N^{task}	Number of total tasks
X_i	Index of the ES to which task τ_i is assigned
U_j^{total}	Total CPU utilization at ES j
Y_j	Maximum allowable CPU utilization for ES j
$P_j(u)$	Amount of power consumed at ES j when its CPU utilization is u, $(0 \leq u \leq 1)$
$H_{i,j}$	Binary constant to indicate whether task τ_i can be accepted by ES j
$R_{i,j}$	Reward received by the EIP for processing task τ_i at ES j
$R_{i,j}^{\mathrm{unit}}$	Reward per unit of processing capacity used at ES j
N_i^{sub}	Number of subtasks for task τ_i
$\tau_{i,m}$	mth subtask in task τ_i
$X_{i,m}$	Index of the ES to which subtask $\tau_{i,m}$ is assigned
$P_{i,j}^{\mathrm{algo}}$	Increase in the power drawn by ES j when a new task τ_i is allocated to it
$V_{i,j}^{\mathrm{algo}}$	$\frac{R_{i,j}}{P_{i,j}^{\mathrm{algo}}}$
$G = (V, E)$, G^{R}	Flow network and its residual network to represent edge allocation problem
$f(a, b)$, $C(a, b)$	Flow and cost between vertices a and b in a flow graph $G = (V, E)$
s, v_i, w_j, t	Source, task, ES and sink vertices in flow network G
$(\gamma^{\mathrm{T}}, \delta^{\mathrm{T}})$	Range of task numbers per IoT device used for simulation
$(\gamma^{\mathrm{R}}, \delta^{\mathrm{R}})$	Range of unit reward values used for simulation
$(\gamma^{\mathrm{U}}, \delta^{\mathrm{U}})$	Ranges of U_i values used for simulation

Table 3:

TABLE III
PARAMETER SETTINGS

Parameters	Range description	Default range	Ranges used in the experiments
Number of tasks per IoT device	$(\gamma^{\mathrm{T}}, \delta^{\mathrm{T}})$	(5,10)	(3,6), (5,10), (7,14), (9,18)
Reward	$(\gamma^{\mathrm{R}}, \delta^{\mathrm{R}})$	(1,15)	(1,5), (1,15), (1,25), (1,35)
Task usage, U_i	$(\gamma^{\mathrm{U}}, \delta^{\mathrm{U}})$	(3,8)	(1,6), (3,8), (5,10), (7,12), (1,12)

Table 4:

TABLE IV
COMPARISON BETWEEN EAA-NTS AND EAA-TS FOR DIFFERENT VALUES OF (γ^{U}, δ^{U}) AND P^{RATIO}

Average percentage difference of total reward.					
P^{ratio}	$(\gamma^{\mathrm{U}}, \delta^{\mathrm{U}})$				
	(1,6)	(1,12)	(3,8)	(5,10)	(7,12)
40%	0.01%	0.07%	2.92%	6.95%	7.01%
60%	0.06%	0.04%	1.84%	3.63%	5.60%
80%	0.02%	0.01%	1.15%	2.92%	4.31%
Maximum percentage difference of total reward.					
P^{ratio}	$(\gamma^{\mathrm{U}}, \delta^{\mathrm{U}})$				
	(1,6)	(1,12)	(3,8)	(5,10)	(7,12)
40%	0.04%	0.25%	3.54%	8.66%	9.07%
60%	0.48%	0.33%	2.12%	4.04%	6.18%
80%	0.07%	0.07%	1.15%	3.2%	5.01%

Table 5:

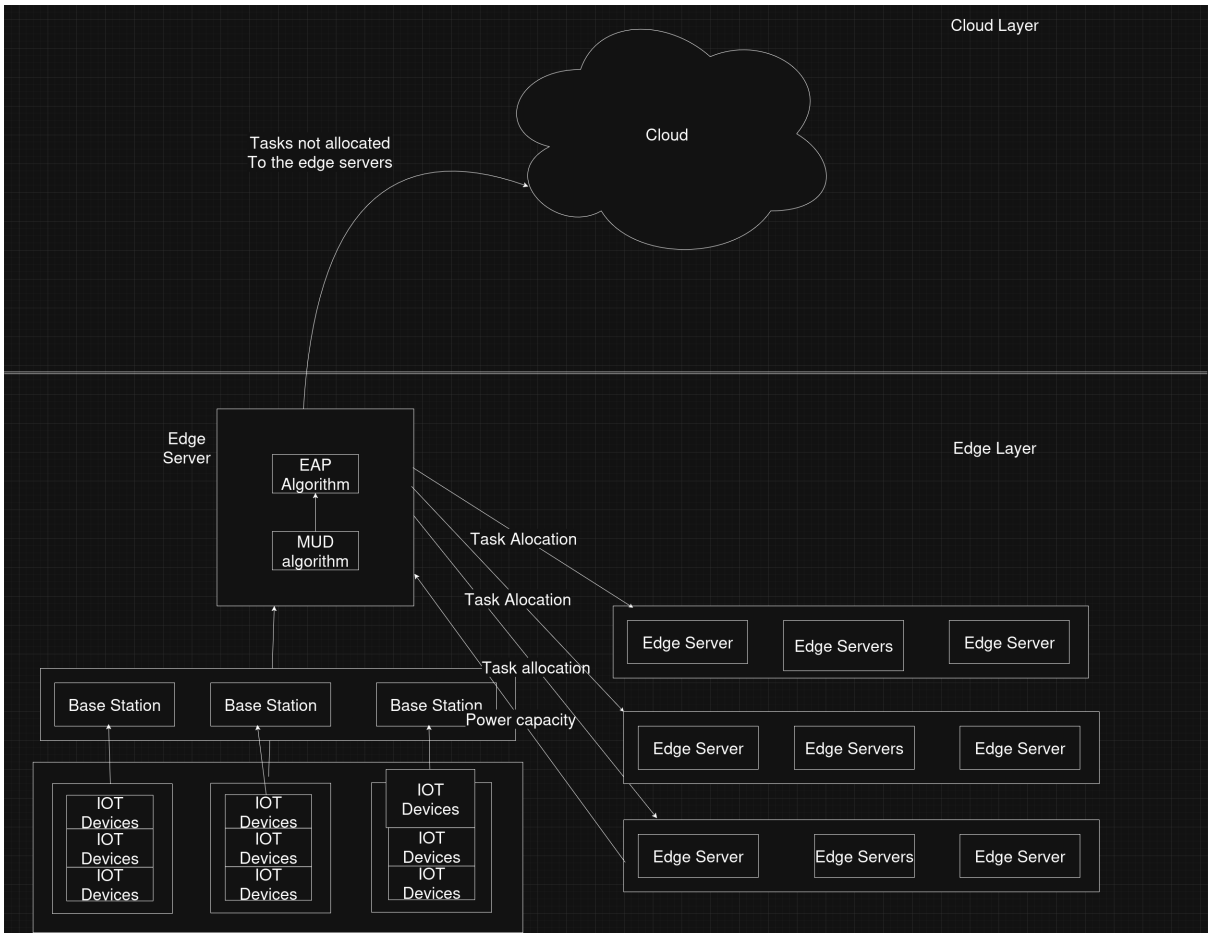
TABLE V

AVERAGE EXECUTE TIME (S) OF THE MUD, EAA-TS, AND EAA-NTS ALGORITHMS AGAINST VALUES OF (γ^T, δ^T)

$(\gamma^{\mathrm{T}}, \delta^{\mathrm{T}})$	(3,6)	(5,10)	(7,14)	(9,18)
N^{task}	3,263	5,720	8,201	10,537
MUD	0.241	0.564	0.921	1.394
EAA-NTS	0.101	0.105	0.109	0.114
EAA-TS	0.493	1.189	1.826	2.905

List of Figures

Architecture Diagram



Contents

List of Abbreviations

MEC: Multiaccess Edge Computing

IoT: Internet of Things

ES: Edge Server

EIP: Edge Infrastructure Provider

QoS: Quality of Service

MCMF: Minimum-Cost-Maximum-Flow

CPU: Central Processing Unit

MKAR: Multiple Knapsack Problem with Assignment Restriction

MCKP: Multiple-Choice Knapsack Problem

MKP: Multiple Knapsack Problem

UDP: Utilization Determination Problem

EAP: Edge Allocation Problem

MUD: Maximum Allowable Utilization Determination

EAA-TS: Edge Allocation Algorithm with Task Splitting

EAA-NTS: Edge Allocation Algorithm without Task Splitting

SPFA: Shortest Path Faster Algorithm

BS: Base Station

WAP: Wireless Access Point

1. Abstract

Edge computing has emerged as a critical technology for reducing latency in IoT applications by processing tasks closer to data sources. However, edge servers operate under significant power constraints, which limits their computational capacity. This report addresses the problem of maximizing total reward (profit) for edge infrastructure providers while maintaining strict power consumption limits across edge servers. We propose a two-phase optimization scheme: (1) Maximum Allowable Utilization Determination (MUD) to establish power-aware workload caps for each server, and (2) Edge Server Allocation (EAP) to optimally assign tasks to servers while maximizing reward. We present two variants of the EAP algorithm—one allowing task splitting (EAA-TS) and one without task splitting (EAA-NTS). Simulations using real server power characteristics demonstrate that our approach achieves (8-80)%. Higher rewards compared to alternative schemes under identical power constraints, with particularly strong performance in low-power scenarios.

Keywords: Edge computing, task offloading, power capping, reward optimization, minimum-cost maximum-flow.

2. Introduction

2.1 Background

The rapid proliferation of Internet-of-Things (IoT) devices has generated massive amounts of data requiring fast computation. Traditional cloud computing architectures introduce significant latency due to the distance between IoT devices and central servers. Edge computing addresses this by deploying computational servers (edge servers) at the network edge, closer to data sources. Multiaccess Edge Computing (MEC) extends this concept by allowing IoT devices within cellular base station coverage to offload tasks to edge servers, significantly reducing processing latency while maintaining service quality. Edge infrastructure providers (EIPs) now offer edge-based services to customers, creating a shared computing environment. Each EIP operates multiple edge servers across different base stations. The revenue model depends on the volume of tasks processed, but the profitability is directly affected by power consumption costs. A typical edge server consumes approximately 10 kW, and thousands of such servers across multiple locations consume megawatts of power—a significant operational concern.

2.2 Motivation

While edge computing has successfully addressed core network congestion and latency issues, two critical challenges remain:

Challenge 1: Limited Processing Capacity - Edge servers have finite computational resources and cannot process all requested tasks, requiring selective task offloading decisions.

Challenge 2: Power Consumption Constraints - Power capping is essential to keep costs manageable and to prevent infrastructure overload during peak demand periods. The relationship between CPU utilization and power consumption is nonlinear and depends on specific server hardware characteristics.

Current approaches typically optimize for either latency minimization or profit maximization independently, without adequately addressing the interplay between power con-

straints and reward optimization. This report proposes an integrated solution that maximizes reward while strictly adhering to power budgets.

2.3 Problem Definition

The core problem—**Edge Server Allocation and Maximum Allowable Utilization Determination (ESA-MAUD)**—can be formally stated as:

Objective: Maximize total reward from offloaded tasks, subject to coverage, capacity, and power constraints. • Each task must be allocated to a server within its coverage area • Total CPU utilization of each server cannot exceed its maximum capacity • Total power consumption across all servers cannot exceed the power limit • Server power consumption depends nonlinearly on CPU utilization This is proven to be NP-hard through reduction from the Multiple Knapsack Problem with Assignment Restriction (MKAR).

Optimization Model:

$$\sum_{j=1}^{N_{ES}} P_j(U_{total_j}) \leq P_{limit} \tag{2.1}$$

3. Literature Survey

3.1 Challenges

- Challenge 1: Computational Complexity - The task allocation problem involves multiple interdependent decisions (which task to offload, to which server, under what constraints). The nonlinear relationship between CPU utilization and power consumption adds further complexity.
- Challenge 2: Power-Aware Resource Management - While many schemes address power consumption by turning servers on/off, this complicates network management by creating connectivity gaps. Instead, this work manages power through CPU utilization control.
- Challenge 3: Conflicting Objectives - Maximizing profit and minimizing power consumption often conflict. Previous work addresses these separately rather than jointly optimizing both objectives.
- Challenge 4: Task-Server Affinity - Coverage constraints mean tasks can only be processed by specific servers, limiting allocation flexibility and requiring constraint-aware algorithms.
- Challenge 5: Real-World Power Characteristics - Server power consumption varies nonlinearly with utilization and depends on specific hardware. Generic models are insufficient; real-world characteristics must be incorporated.

3.2 Research Gap

Previous literature falls into several categories:

- Task Offloading Research: Min et al., Wang et al., and Samanta et al. optimized for latency or QoS compliance but ignored power consumption. Ren et al. considered cloud-edge combinations but not power constraints.

- Server Placement Research: Wang et al., Lahderanta et al., and Xu et al. optimized server locations and workload balancing but did not consider energy consumption.
- Profit-Oriented Research: Anglano et al., Nguyen et al., and Hao et al. maximized revenue/profit but did not jointly optimize power consumption. Huang et al. maximized revenue with limited ES resources but again ignored power constraints.
- Energy-Efficiency Research: Extensive work exists on reducing mobile device energy consumption and server efficiency, but predominantly focuses on binary on/off decisions rather than power-aware utilization control.
- Gap: No prior work integrates reward maximization with strict power constraints by controlling CPU utilization while accounting for real, nonlinear server power characteristics. This work uniquely addresses this gap through MCMF-based task allocation and power-aware utilization management.

4. Problem Formulation

4.1 System Overview

The MEC system consists of the following components:

- **Edge Servers (ESs):** Located at or near base stations (BSs), capable of processing offloaded tasks. Each ES j has maximum processing capacity $C_{\max,j}$ and idle power consumption β_j. Active power consumption $\alpha_j(u)$ depends on CPU utilization u.
- **IoT Devices:** Generate tasks requiring computation. Tasks are characterized by their computational requirement (normalized usage U_i) and associated reward $R_{i,j}$ if processed by ES j.
- **Coverage Relationships:** Each task τ_i can only be processed by ES j if $H_{i,j} = 1$ (indicating the device is within coverage).
- **Reward Model:** EIPs earn revenue proportional to computation performed and incur costs from resource usage. Net profit (reward) is linear in utilization: $R_{\text{unit},i,j} = R_{i,j}/U_i$.
- **Power Model:** Server power consumption combines idle power and utilization-dependent active power:

  $$P_j(u) = \alpha_j(u) \cdot u + \beta_j \cdot (1 - u)$$

  where u is normalized CPU utilization (0 to 1).
- Constraints

  Constraint 1 - Processing Capacity: Total utilization at each ES cannot exceed its maximum allowable utilization Y_j:

  $$\sum_{\forall i, X_i = j} U_i \leq Y_j \quad \forall j$$

Constraint 2 - Coverage: Tasks can only be allocated to servers within coverage:

$$H_{i,X_i} = 1 \quad \forall i$$

Constraint 3 - Power Budget: Total power across all servers cannot exceed limit:

$$\sum_{j=1}^{N_{ES}} P_j(U_{\text{total},j}) \leq P_{\text{limit}}$$

- Optimization Problem

$$\begin{aligned}
&\underset{X_i}{\text{Maximize}} && \sum_{i=1}^{N_{\text{task}}} R_{i,X_i} \\
&\text{Subject to:} && \sum_{\forall i, X_i = j} U_i \leq Y_j, \quad \forall j \\
& && H_{i,X_i} = 1, \quad \forall i \\
& && \sum_{j=1}^{N_{ES}} P_j(U_{\text{total},j}) \leq P_{\text{limit}} \\
& && X_i \in \{0, 1, \ldots, N_{ES}\}, \quad \forall i
\end{aligned}$$

5. Proposed Architecture

The solutions employs a two-phase architecture:

5.1 Phase 1 - Maximum Allowable Utilization Determination (MUD)

- **Input:** Task characteristics (U_i, $R_{i,j}$), server specifications ($C_{\max,j}$, $P_j(u)$), power limit.
- **Process:** Determine Y_j for each server that satisfies the power constraint while maximizing expected reward.
- **Output:** Utilization caps Y_j for each server.
- **Benefit:** Transforms the power constraint into a utilization constraint, enabling an efficient phase 2.

5.2 Phase 2 - Edge Server Allocation (EAP)

- **Input:** Utilization caps Y_j from Phase 1, tasks, coverage matrix $H_{i,j}$.
- **Process:** Allocate each task to the optimal server within utilization constraints.
- **Output:** Task-to-server assignments X_i.
- **Variants:** With task splitting (allows one task to be split across multiple servers) or without task splitting (each task is assigned to a single server).

Centralized Execution Both phases run on a master server (central cloud or a designated ES). The results are then distributed to individual servers and IoT devices.

6. Methodology

6.1 Phase 1: Maximum Allowable Utilization Determination (MUD)

The MUD algorithm (Algorithm 1) determines Y_j values that satisfy the power constraint. It operates as a greedy algorithm based on reward-to-power efficiency ratios.

Key Concept For each potential task-server assignment, calculate the specific reward $V_{\text{algo},i,j} = R_{i,j}/P_{\text{algo},i,j}$, where $P_{\text{algo},i,j}$ is the incremental power increase from assigning task i to server j.

Algorithm Steps

1. Initialize all $X_{\text{tmp},i} = 0$ (all tasks temporarily unassigned), $P_{\text{used}} = 0$.
2. While the power budget allows and tasks remain:
   - For each unassigned task and compatible server, calculate $V_{\text{algo},i,j}$.
   - Find the maximum $V_{\text{algo},i,j}$.
   - If adding this assignment stays within the power limit, assign it.
   - Otherwise, stop.
3. For each server j, $Y_j = \sum U_i$ for all tasks assigned to j.

Time Complexity $O(N_{\text{task}}^2)$ in the worst case.

Output Maximum allowable utilization for each server that meets the power constraint.

6.2 Phase 2: Edge Server Allocation Problem (EAP)

The EAP is formulated as a Minimum-Cost-Maximum-Flow (MCMF) graph problem. The flow network contains:

Vertices

- Source vertex s.
- Task vertices v_i for each task τ_i.
- Server vertices w_j for each server ES_j.
- Sink vertex t.

Edges and Capacities

- $(s \to v_i)$: capacity U_i, cost 0.
- $(v_i \to w_j)$: capacity U_i, cost $-R_{\text{unit},i,j}$ (negative because we maximize by minimizing cost).
- $(w_j \to t)$: capacity Y_j, cost 0.

EAA-TS: With Task Splitting

This variant allows each task to be split across multiple servers. It uses the Shortest Path Faster Algorithm (SPFA) to iteratively find minimum-cost augmenting paths.

Advantages

- Achieves better overall utilization.
- Can accommodate more tasks under tight power constraints.
- Provides optimal or near-optimal solutions.

Process

1. Build a residual graph with forward and reverse edge capacities.
2. While an augmenting path exists from the source to the sink:
   - Find the minimum-cost path using SPFA.
   - Update flows along this path.
3. Assign each unit of flow to the appropriate server.
4. Update $X_{i,m}$ for each subtask.

Time Complexity $O(|V||E| \cdot \sum Y_j)$ in the worst case.

EAA-NTS: Without Task Splitting

Each task must be allocated to a single server. This is a greedy algorithm selecting tasks by the highest reward-per-unit cost.

Process

1. While tasks remain and capacity is available:
   - Find the minimum-cost path (highest reward-per-cost).
   - If the server has capacity, assign the entire task.
   - Otherwise, remove that edge and try the next option.
2. Remove fully-allocated tasks.

Time Complexity $O(N_{\text{task}} \cdot \log N_{\text{task}})$.

7. Results

7.1 Simulation Environment

We conducted experiments using EdgeSimPy, a discrete-event simulator for edge computing. The setup used four commercial server models from the SPECpower_ssj 2008 benchmarks with 86–288 cores and power consumption of 372–1260W at full load.

Network Infrastructure We modeled Melbourne's cellular network data—815 wireless access points and 125 base stations with 450–750m coverage per station. This real-world topology provided realistic conditions for testing.

Our Process We started by preparing a testing subset in Excel to validate parameters and ensure data consistency. This allowed us to verify the simulation configuration before scaling to full experiments. We then configured EdgeSimPy with the server specifications and network topology, implementing our EAA-TS algorithm alongside comparison schemes. Each experiment ran under identical power constraints, and we systematically varied power ratios from 40% to 100% to test performance under different resource availability scenarios.

7.2 Results and Analysis

Performance Advantage EAA-TS outperformed all alternatives by 8–80% under identical power budgets. At a 40% power ratio—the most constrained scenario—our approach achieved 38–80% higher rewards, proving most effective when resources are limited.

Task Splitting Impact Task splitting showed minimal benefit (under 0.48% difference) when minimum task usage was 1. With substantial task requirements, however, splitting improved performance by up to 22%, particularly under tight power constraints where efficient resource packing matters most.

Power Efficiency Competing schemes required 8–65% additional power to match EAA-TS performance. MUD+HUR needed 12–29% more power while LPF+HR demanded 8–25% extra—demonstrating significant energy savings from our approach.

Scalability and Sensitivity Performance gains remained consistent across varying edge server counts. Computation times stayed under 10 seconds for moderate workloads, enabling real-time decisions. The algorithm performed better with more tasks (greater optimization flexibility) and wider reward ranges, while maintaining stability across different task usage distributions.

8. Conclusion

This report presents a comprehensive solution to the reward-oriented task offloading problem under strict power constraints in multi-access edge computing environments. By decomposing the problem into two phases—utilization determination and task allocation—and employing novel algorithms based on greedy optimization and minimum-cost-maximum-flow, we achieve significant improvements over existing approaches.

- **Problem Formulation:** First to formally integrate reward maximization with power-aware CPU utilization control, accounting for nonlinear server power characteristics.
- **Algorithm Innovation:**
  - MUD algorithm for power-aware utilization planning.
  - MCMF-based task allocation with and without task splitting.
  - Greedy reward-to-power ratio optimization.
- **Empirical Validation:** Comprehensive simulations demonstrating 8–80% reward improvements, with particularly strong performance under constrained power scenarios.
- **Practical Impact:** Solutions are computationally feasible for deployment and provide EIPs with significantly higher profit margins under power budgets.

Bibliography

[1] N. Abbas, Y. Zhang, T. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[2] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, Oct. 2020.

[3] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[4] M. Song, Y. Lee, and K. Kim, "Reward-oriented task offloading under limited edge server power for multiaccess edge computing," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13425–13438, Sept. 2021.

[5] C. Anglano, M. Canonico, and M. Guazzone, "Profit-aware resource management for edge computing system," in *Proc. ACM Int. Workshop Edge Syst. Anal. Netw.*, May 2018, pp. 25–30.

[6] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[7] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep se-

quential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[8] A. Samanta, Z. Chang, and Z. Han, "Latency-oblivious distributed task scheduling for mobile edge computing," in *Proc. IEEE Global Commun. Conf.*, Dec. 2018, pp. 1–7.