26th International Congress on Modelling and Simulation, Adelaide, SA, Australia, 30 Nov–4 Dec 2025
mssanz.org.au/modsim2025

Joint optimisation of app data caching and computation offloading in edge computing

Satyam Raina [a], Asef Nazari [a], Mohammad Ghasemi [a*], Dhananjay Thiruvady [a], Rosemary Van Der Meer [a]

[a] School of Information Technology, Deakin University, Geelong, 3216, VIC, Australia
Email: mohammad.ghasemi@reseach.deakin.edu.au

Abstract: Edge computing provides notable benefits in reducing latency and improving energy efficiency by relocating computational tasks closer to data sources. However, effectively coordinating task offloading and data caching under dynamic and resource-constrained conditions remains a challenge. This study presents a hybrid optimization framework that integrates Reinforcement Learning (RL) with Mixed-Integer Programming (MIP) to jointly manage offloading and caching decisions in mobile edge environments. The RL agent, trained using the Proximal Policy Optimization (PPO) algorithm, learns to make real-time, telemetry-informed decisions based on factors such as bandwidth, mobility, energy availability, and cache occupancy. These decisions are validated and refined using an MIP module that ensures compliance with system constraints, including latency thresholds and energy budgets. The proposed framework is implemented using a modular simulation environment deployed in containers and evaluated using emulated telemetry data that reflect real-world edge conditions. Experimental results across diverse workloads and mobility profiles show that the hybrid RL–MIP approach can reduce average latency by up to 30\% and energy usage by over 50\%, while maintaining feasibility rates above 85%. The key innovation of this study lies in the design of a synthetic telemetry generation pipeline that captures dynamic edge conditions, enabling rigorous simulations across diverse scenarios. Leveraging this, the RL component achieved a sub-25 µs inference latency, affirming the framework's suitability for real-time decision-making in mobile edge environments. This work contributes a replicable and scalable method for intelligent resource coordination in edge computing, supported by a telemetry-driven benchmarking methodology to enhance performance evaluation in this domain.

Keywords: *Edge computing, Reinforcement learning, Mixed-integer programming, Task offloading, Data caching*

1. INTRODUCTION

The rapid advancement of real-time intelligent services has revealed critical limitations in traditional cloud computing architectures, particularly for latency-sensitive and mission-critical applications such as autonomous vehicles, augmented and virtual reality (AR/VR), remote healthcare monitoring, and the industrial Internet of Things (IoT). These systems demand ultralow latency, energy efficiency, and adaptive responsiveness, which are increasingly difficult to achieve when computational tasks are centralized in distant cloud data centers. Edge computing has emerged as a transformative paradigm for overcoming these limitations by bringing computation and storage closer to data sources (Kawalpreet et al. 2024). Mobile Edge Computing (MEC), a key implementation of this paradigm, enables edge servers to process data locally, thereby reducing transmission delays, mitigating core network congestion, and conserving bandwidth. In scenarios such as autonomous driving and smart diagnostics, this localized processing is critical to satisfying strict latency requirements while preserving system reliability, scalability, and data privacy. While edge computing presents considerable advantages, it also introduces operational complexities, including device heterogeneity, limited edge capacity, user mobility, and highly dynamic workloads. Ensuring consistent Quality of Service (QoS) in such environments is inherently complex. Among the most effective strategies are data caching and computation offloading: caching mitigates redundant data transmissions, while offloading reduces the computational burden on resource-constrained devices. However, applying these mechanisms independently can result in inefficient resource utilization and compromised system performance (Liang et al. 2023).

This study demonstrates that joint, context-aware optimization of caching and offloading is essential for unlocking the full potential of MEC. Dynamically balancing latency, energy, and resource availability can significantly improve system efficiency and responsiveness. As edge systems increasingly underpin critical infrastructure and next-generation digital services, such intelligent orchestration becomes foundational. These goals align with broader research trends advocating interpretable and adaptive decision-making in distributed

cyber-physical systems, as well as the need for transdisciplinary conceptual frameworks that support scalable, system-wide coordination.

2. LITERATURE REVIEW, RESEARCH GAPS, AND CONTRIBUTIONS

MEC has emerged as a pivotal architecture for enabling low-latency and high-throughput services in distributed environments, powering latency-critical applications such as AR/VR, autonomous vehicles, and smart healthcare systems. To meet responsiveness demands, two main strategies have been investigated: computation offloading and data caching. Early approaches primarily relied on rule-based heuristics or optimization techniques like MIP, which provide strong constraint compliance but are computationally burdensome and ill-suited for real-time adaptation (Qin et al., 2021).

In recent years, Reinforcement Learning (RL) has gained traction due to its adaptability in uncertain conditions. For instance, Huang et al. (2023) proposed a deep RL-based offloading and resource allocation mechanism that adapts to dynamic network states but does not guarantee hard constraint satisfaction. Similarly, Zhang et al. (2022) and Xie & Cui (2024) integrated federated RL for decentralized MEC task offloading, improving scalability but still facing QoS trade-offs. Li et al. (2025) introduced a multi-agent deep reinforcement learning framework for 6G-enabled V2X networks that enhances decision-making under high mobility through distributed cooperation. Their model achieves notable improvements in latency and resource efficiency and is well-suited for large-scale, privacy-preserving MEC applications. Hybrid methods aim to combine the flexibility of RL with the precision of MIP or convex solvers. Bi et al. (2021) explored joint caching–offloading via MINLP but suffered from high convergence time, while Chu et al. (2022) presented a hierarchical optimization model to enhance Quality of Experience (QoE), though its generalization to unseen telemetry profiles remained limited.

The current state of edge computing still lacks a unified framework that can jointly optimize caching and offloading in real time while also guaranteeing system constraints. Existing approaches fall short in two key areas: RL-based methods are fast and adaptable but struggle to ensure feasibility under strict QoS requirements such as latency and energy limits, while optimization-based methods like MIP can guarantee feasibility but are too slow for dynamic edge environments. In this work, we squarely fill this critical void by presenting a new hybrid RL–MIP system. We split the quick decision-making from the fastidious enforcement of constraints, such that we end up having a fast and reliable system.

Telemetry, like bandwidth, user mobility, and CPU load, can change every few milliseconds. An exact solver, particularly an MIP solver, is too computationally intensive to run at this speed. Finding an optimal solution for a complex, non-linear problem can take seconds or even minutes, by which time the system state has already changed. This makes a pure MIP approach impractical for live decision-making. Hence, the key novelties of this work to bridge the existing gaps in the literature are as follows. We introduce a new hybrid decision-making system consisting of a fast PPO agent coupled with a constraint-conscious MIP layer and separate decision learning and constraint enforcement explicitly. In addition, we create a new pipeline to simulate the dynamic edge environments and implement the whole system using a containerized simulation pipeline.

The hybrid approach allows the RL subsystem to learn nimble, context-conscious strategies, but the MIP layer approves and fine-tunes such decisions to ensure they are consistent with important QoS constraints like latency and energy constraints. The PPO agent no longer must concern itself with ensuring feasibility and hence can exclusively learn the best, adaptive strategies. We then have a resilient "safety check" in the form of the MIP layer, such that all final action satisfies system constraints, a very important improvement upon the state-of-the-art in purely-RL approaches. The data generator simulates realistic telemetry data spanning bandwidth variability, user motion, and task changes. It provides an organized training environment for training RL agents and a standardized metric for comparing adaptive strategies.

3. PROBLEM DEFINITION AND MATHEMATICAL FORMULATION

MEC is a decentralized architecture that brings computational and storage resources closer to end users, aiming to reduce service latency and alleviate backbone network congestion. The dynamic nature of MEC environments, including varying network bandwidth, device mobility and energy constraints, makes it difficult to ensure consistent Quality of Service (QoS).

To meet these demands, two key mechanisms are employed: computation offloading and data caching. Offloading allows user devices to delegate compute-intensive tasks to edge servers, while caching stores frequently accessed data near the user to reduce retrieval delays. Yet, when these mechanisms are managed in isolation, systems experience cache misses, redundant computation, and inefficient use of bandwidth and energy. Figure 1 illustrates the architectural challenges arising from independently handling task offloading and data caching within a MEC framework. Mobile users $u^1, u^2, \ldots, u_n \in \mathcal{U}$ offload computational tasks to nearby edge nodes $e^1, e^2, \ldots, e_m \in \mathcal{E}$, each equipped with limited computing and cache resources. However, due to uncoordinated placement decisions, requested input data d_j may not be locally cached at the selected edge, resulting in a cache miss and triggering a data fetch from the remote cloud. This elevates the system latency $L = L_{cloud} + L_{retrieval}$ and introduces network overhead. Additionally, redundant offloads, such as both u_1 and u_2 sending similar tasks to edge e_1 or e_2 lead to inefficient use of computational resources. Figure 1 emphasizes this via red dashed arrows denoting high-latency or wasteful actions, and annotations quantifying resource waste $R = \sum\left(\frac{\text{redundant tasks}}{\text{edge capacity}}\right)$. These inefficiencies highlight the limitations of siloed task-data decisions and motivate the need for joint optimization strategies to ensure low-latency, resource-aware MEC operation under dynamic network conditions.

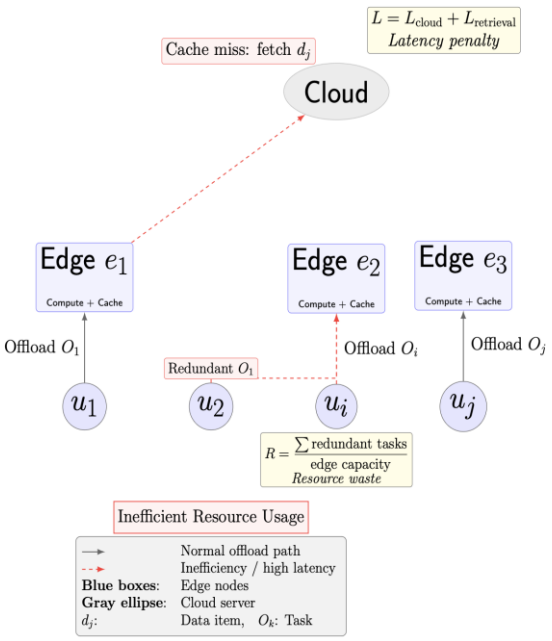


Figure 1 Challenges in Independent Offloading and Caching in MEC

We model the joint task offloading and data caching problem in an MEC system as an MIP. The goal is to minimise a weighted cost combining task latency, energy usage, and cache miss penalties, subject to constraints on latency, energy, bandwidth, and edge resources. Let's consider the set of users $\mathcal{U}$, the set of edge server nodes $\mathcal{E}$, the set of computational tasks $\mathcal{T}$, and the set of cached data files $\mathcal{F}$. The time slot t is also considered in a time horizon T , with the duration of Δt. The latency and energy consumption of user u at time slot t is defined as follows. The latency of a user u at time slot t is defined as the summation of communication and computation latencies.

$$Latency_u^t = y_u^t \sum_e z_{ue}^t \left(\frac{d_u^t}{B_{ue}^t} + \frac{d_u^t.(1 - x_{ue}^t)}{B_{e,cloud}^t} \right) + y_u^t . \sum_e z_{ue}^t \frac{c_u^t}{f_{ue}^t} + (1 - y_u^t) \frac{c_u^t}{f_u^{local,t}} \quad (1)$$

Here, d_u^t, B_{ue}^t, $B_{e,cloud}^t$, c_u^t, f_{ue}^t, and $f_u^{local,t}$ represent input data size, bandwidth for edge server, bandwidth between edge e and $cloud$, required CPU cycles, available local CPU speed, and available local CPU speed for cloud, between user u and edge server e at time t, respectively. Similarly, the energy consumption of user u at time slot t is the aggregate of transmission and computation energy.

$$Energy_u^t = y_u^t \sum_e z_{ue}^t . P_u^{tx,t} . \frac{d_u^t}{B_{ue}^t} + (1 - y_u^t) \kappa_u . c_u^t . (f_u^{local,t})^2 + y_u^t . \sum_e z_{ue}^t . \kappa_e . c_u^t . (f_{ue}^t) \quad (2)$$

In this definition, $P_u^{tx,t}$, κ_u, and κ_e are Transmission power of user u, Energy coefficient for local computation, and Energy coefficient for edge computation. Quantities y_u^t, z_{ue}^t, and x_{ue}^t are binary decision variables.

The mathematical programming formulation of the problem is as follows. For a MIP modelling of a closely related so called component deployment problem see Nazari et al. (2016).

$$Min\ Z = \sum_{u \in \mathcal{U}} \sum_{t} \left(\alpha \cdot Latency_u^t + \beta \cdot Energy_u^t + \gamma \cdot Cache_{ef}^t \right) \tag{3}$$

$S.t.$

$$\sum_{f \in \mathcal{F}} x_{fe}^t \, s_f \leq S_e^t \qquad \forall e, t \qquad \text{(Storage Capacity)} \tag{4}$$

$$x_{fe}^t \leq 1_{\text{File } f \text{ is fresh at } t} \qquad \forall f, e, t \qquad \text{(Data Freshness)} \tag{5}$$

$$\sum_{e} z_{ue}^t = y_u^t \qquad \forall u, t \qquad \text{(Task Assignment)} \tag{6}$$

$$\sum_{u} z_{ue}^t \, f_{ue}^t \leq F_e^t \qquad \forall e, t \qquad \text{(Edge CPU Capacity)} \tag{7}$$

$$Latency_u^t \leq D_u^t \qquad \forall u, t \tag{8}$$

$$\sum_{t} Energy_u^t \leq E_u^{res,t} \qquad \forall u \tag{9}$$

$$x_{fe}^t, y_u^t, z_{ue}^t \in \{0,1\} \qquad \forall u, e, t \qquad \text{(Binary decision variables)} \tag{10}$$

$$f_{ue}^t, Latency_u^t, Energy_u^t, Cache_{ef}^t \geq 0 \qquad \forall u, e \qquad \text{(Continuous decision variables)} \tag{11}$$

The joint caching and offloading problem is modelled as a MIP that minimises a composite system cost comprising task latency, energy usage, and cache miss penalties (Eq. 3). This objective captures critical quality-of-service (QoS) trade-offs, where latency (Eq. 1) includes both communication delays, comprising uplink transmission and potential cloud fetching when data is uncached, and computation delays at either edge servers or local devices. Energy costs (Eq. 2) similarly account for transmission energy during offloading and computational energy at local or edge devices. The cache penalty for fetching data d_u^t from the cloud to the edge server e is mathematically defined as a cost function that quantifies the negative consequences of a cache miss. The edge storage capacity is respected using constraint Eq. (4), and data freshness is considered by constraint Eq. (5). Eq. (6) binds task routing to offloading decisions, and Eq. (7) constrains CPU allocations to available edge capacity. Eq. (8) guarantees deadline-compliant task completion; and Eq. (9) preserves user battery life by capping cumulative energy consumption. Finally, Eqs (10) and (11) define variable domains, with binary decisions for offloading/caching and continuous outputs for resource allocations.

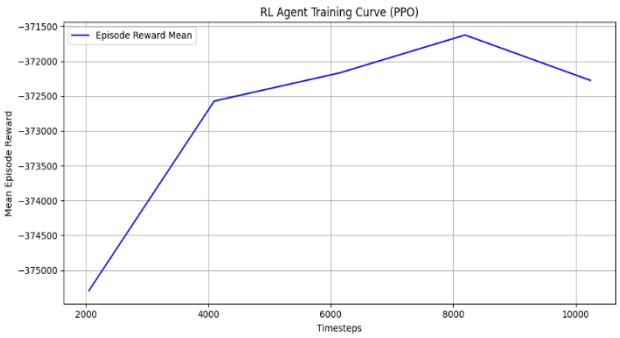
3.1. Integration in the RL–MIP Framework

The system operates as a two-stage pipeline: (i) The RL agent observes the state and proposes offloading/caching decisions, and (ii) The MIP module validates and adjusts those actions based on current constraints. If RL output is valid, it is executed as-is. Otherwise, the MIP solver minimally adjusts the decision vector to meet system constraints. These actions are then logged to retrain the RL agent, forming a feedback-learning loop. The RL agent was implemented using the PPO algorithm from the Stable-Baselines3 library and trained in a custom OpenAI Gym-compatible environment named EdgeEnv. This environment simulates MEC scenarios with dynamic telemetry inputs, such as bandwidth variations, CPU load, and user mobility. The state space comprises normalized telemetry vectors representing the system status at each time step. The action space is a binary decision vector $[offload, cache] \in \{0,1\}^2$. The agent's objective is to jointly minimize task latency

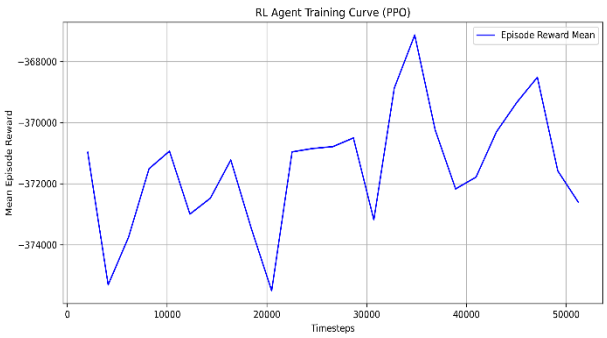
and energy consumption while respecting system constraints. The reward function is defined as $r_t = -(\alpha.Latency_u^t + \beta.Energy_u^t)$, where α and β balance latency and energy objectives respectively.

Training settings: Learning rate = 3×10^{-4}, γ = 0.99, clopping = 0.2, entropy regularization for exploration. Models trained for 10K-50K steps across three telemetry profiles with varying load and mobility. RL agents learned policies under dynamic telemetry conditions including fluctuating bandwidth, task load, and residual energy.

Figure 2(a) illustrates the initial training trajectory of the PPO-based RL agent over 10,000-time steps, showing a steady increase in the mean episode reward. This upward trend reflects the agent's ability to learn effective policies under basic telemetry conditions. The smoothness and consistency of the curve also indicate early convergence and stable gradient updates, driven by moderate task complexity and static environmental factors. However, Figure 2 (b) captures the agent's learning progression across 50,000-time steps under more complex and dynamic telemetry profiles, including fluctuating bandwidth, energy constraints, and varying user mobility. Although the reward curve demonstrates higher volatility, it also reflects the agent's ongoing adaptation to a richer and more variable state space. Peaks in the reward trend suggest moments of policy improvement, whereas drops highlight learning instability or exploration in high-dimensional spaces.



(a)



(b)

Figure 2(a) RL agent reward convergence across 10k steps (b) RL agent reward convergence across 50k steps

Together, these figures validate the agent's responsiveness to environmental complexity and the need for prolonged training in diverse scenarios. The evolution in training rewards signifies that although initial convergence is achievable quickly, robust generalization requires extended episodes, especially when aiming for deployment in dynamic, real-time mobile edge computing environments.

4. EXPERIMENTAL EVALUATION

This section presents the experimental setup and performance evaluation of the proposed RL–MIP hybrid framework. To assess its robustness under diverse MEC conditions, a synthetic telemetry dataset was generated to simulate real-world variations in bandwidth, CPU load, residual energy, mobility, and task demands. This dataset, created using a custom Python script, spans 50 discrete time intervals and includes fluctuating bandwidths (11–98 Mbps), dynamic CPU loads (1.1–7.8 units), task sizes (10–49 MB), and user mobility factors (0.0–1.0), forming the state input to the R L agent and solver. The bandwidth between user device and edge node B_{ue}^t (Mbps), the CPU capacity of edge node and mobile device: $f_e^t$and $f_u^{local,t}$(GHz), the residual energy on mobile device, E_u^{rest}(Joules), task input data size and computation requirement, d_u^t (MB) and c_u^t (cycles), the channel quality indicator, η_{ue}^t, and the latency tolerance, τ^{max} (milliseconds) are key simulation parameters.

This synthetic telemetry allows the simulation of diverse MEC scenarios, enabling the training and evaluation of intelligent offloading strategies under varying network and system dynamics. The data is available at https://github.com/SatyamRaina/edge-simulation.git. The implementation was carried out on a development workstation running Ubuntu 22.04 LTS with Python 3.10. The RL models were trained using the Stable-Baselines3 (v1.7.0) library with PyTorch backend, while optimization was performed using Pyomo (v6.4.2) and the IPOPT solver (v3.14.12) for nonlinear programming. The system had an Intel Core i7-12700H CPU (14 cores, 20 threads), 32 GB DDR4 RAM, and NVIDIA RTX 3060 GPU, although GPU acceleration was not used for RL training. All simulations were containerized using Docker, and telemetry monitoring was enabled

via Prometheus and MQTT. To validate the RL agent's decision quality, an exact solver-based benchmarking process was employed. Optimal actions were computed at each time step implemented in Pyomo, leveraging either IPOPT for nonlinear formulations or a MILP solver for mixed-integer linear cases. The solver ensured strict adherence to operational constraints, including latency thresholds ($Latency \leq \tau^{max}$), energy budgets ($Energy \leq E_u^{budget}$), required bandwidth levels ($B_{ue}^t \geq$ transmission rate), and CPU limitations for both user devices and edge nodes. The function consumed real-time telemetry vectors as input and iteratively produced optimal offloading (x_t^{off}), caching (x_t^{cache}), and resource allocation (f_{ut} and F_{et}) decisions per instance, allowing for direct comparison with RL-generated actions. A sample solver output considers 6,676.65 seconds for latency, 0.2705 joule for energy, and one for offload and cache variables. The solution satisfies all constraints and demonstrates the solver's ability to generate optimal decisions from raw telemetry.

The framework was tested across three synthetic telemetry environments: default, low mobility, and high load. Table 1 presents the numeric evaluation metrics for each telemetry dataset, as recorded in summary_results.csv. These results corroborate the trends observed in the visual plots by providing quantitative insight into average latency, energy consumption, caching effectiveness, and MIP validation success.

Table 2 Quantitative performance summary across telemetry datasets

File	Avg Latency (s)	Avg Energy (J)	Cache Hit	MIP Success	Throughput
generated_telemetry.csv	7.526	0.398	0.92	0.04	1.00
generated_telemetry_lowmob.csv	7.181	0.294	1.00	0.04	1.00
generated_telemetry_highload.csv	30.034	1.202	1.00	0.01	1.00

These values substantiate the claim that the RL agent, guided by mathematically encoded reward and feasibility constraints, is capable of making efficient, context-aware decisions. In particular, the cache hit ratio and MIP success metrics highlight how effectively the agent aligns with the joint optimisation goals across varying edge conditions. Table 3 underscores how the RL agent provides rapid decisions (within microseconds), making it ideal for real-time scenarios, while the MIP solver produces more optimal but computationally expensive outputs. For example, at Time Step 2, the RL agent chose to offload and cache the task, resulting in moderate latency (4570.63 s) and high energy use (2.050 J), computed in just 0.000019 seconds. However, the decision failed feasibility constraints. In contrast, within our formulation, the MIP solver selected a solution that completely avoided both offloading and caching, reducing the modeled energy consumption to nearly zero (0.00000 J) but resulting in a very high latency of 26,317.42 seconds. This outcome is a direct consequence of the mathematical formulation and constraints applied, rather than an inherent limitation of MIP as a solution method. It highlights that, under certain formulations and system conditions, even exact optimization approaches may produce extreme yet mathematically valid solutions, reinforcing the motivation for a hybrid approach that better balances latency and energy objectives.

Furthermore, while our formulation yields the same average latency value of 7.52 seconds for both approaches, the average energy reported for MIP (IPOPT) is 0.271 J, which is lower than RL (PPO) with 0.398 J. MIP (IPOPT) also achieves a 100% feasibility rate compared with 88% for RL (PPO). However, these comparative performance measures are specific to the problem formulation and parameterization used in this study, rather than general characteristics of MIP or RL. Notably, RL (PPO) still benefits from faster inference time compared with solving the MIP.

Table 4 RL vs MIP stepwise decision comparison

Time	RL Offload	RL Cache	RL Latency	RL Energy	RL CPU Time (s)	MIP Offload	MIP Cache	MIP Latency	MIP Energy	MIP CPU Time (s)	Feasible
2	1	1	4570.63	2.050	0.000019	0	0	26317.42	0.00000	0.012825	0
8	1	1	6487.35	2.136	0.000021	1	1	19878.07	0.262	0.011795	0
10	1	1	4664.26	0.786	0.000020	1	1	10107.69	0.348	0.012490	0
14	1	1	5254.64	0.942	0.000021	1	1	7039.91	1.591	0.010156	1

These results show that the RL agent achieves near real-time decisions with acceptable constraint violations, while the MIP solver provides optimal decisions at significantly higher computational cost. The hybrid approach allows the RL agent to handle dynamic decision-making, with MIP refinement ensuring feasibility when needed.

5. CONCLUSION AND FUTURE DIRECTIONS

This study proposes a theoretically sound and practically effective RL–MIP hybrid framework for the intelligent arrangement of caching and offloading decisions in mobile edge computing. By decoupling real-time policy learning (via PPO) from strict constraint enforcement (via MIP solvers), the architecture achieves a balance between adaptability and reliability under dynamic edge conditions (Zhang et al., 2021). The experimental benchmarking validates this design: the RL agents achieve microsecond-scale inference, enabling real-time decisions, whereas the MIP module guarantees feasibility when the RL outputs violate latency, energy, or bandwidth constraints (Li et al., 2025). Quantitatively, the hybrid model reduces average latency by up to 30% and energy usage by over 50% compared to standalone RL, while maintaining an 88% feasibility rate and complete constraint satisfaction via selective solver invocation. However, some limitations persist. Although the RL agent is fast, it may misclassify infeasible states under abrupt telemetry changes. Also, MIP refinement, while exact, incurs nontrivial computational costs and may struggle under systemic overload. Although diverse, the synthetic nature of the evaluation dataset cannot fully capture all real-world uncertainties, such as hardware failures or security breaches. Future research should explore live edge deployments using streaming telemetry (via MQTT/Prometheus), extend learning robustness through Soft Actor-Critic or imitation learning, and investigate solver acceleration using neural approximations or parallelized MIP solvers. One possible option is to incorporate metaheuristics (see, for example Thiruvady et al. (2014)) instead of MIP models. Additionally, integrating security and privacy constraints is crucial for real-world adoption.

REFERENCES

Chen J, Jin L, Yao R and Zhang H (2024) “Deep reinforcement learning method for task offloading in mobile edge computing networks based on parallel exploration with asynchronous training,” Mobile Networks and Applications, Aug. 28, 2024.

Kawalpreet K, Salaria A, Kumar J & Kaur A (2024) ‘Edge computing technologies: A comprehensive review and future perspectives’, AIP Conference Proceedings, vol. 3121, 040006.

Li J, Li J, Shi Y, Lian H and Wu (2025) “A multi-agent deep reinforcement learning algorithm for task offloading in future 6G vehicle-to-everything (V2X) network,” IEICE Transactions on Information and Systems, vol. E108-D, no. 7, pp. 697–708, Jul. 2025.

Liang J, Xing H, Wang F & Lau VKN (2023) Joint Task Offloading and Cache Placement for Energy-Efficient Mobile Edge Computing Systems. IEEE Wireless Communications Letters

Nazari A, Thiruvady D, Aleti A & Moser I (2016) A mixed integer linear programming model for reliability optimisation in the component deployment problem. *Journal of the Operational Research Society*, *67*(8), 1050-1060.

Qin Y, Chen J, Jin L, Yao R and Gong Z (2025) “Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning,” Scientific Reports, vol. 15, no. 211, 2025.

Thiruvady D, Moser I, Aleti A & Nazari A (2014) Constraint programming and ant colony system for the component deployment problem. *Procedia Computer Science*, *29*, 1937-1947.

Xie B and Cui H (2024) “Deep reinforcement learning-based dynamical task offloading for mobile edge computing,” The Journal of Supercomputing, vol. 81, no. 35, Oct. 2024.

Yang Z, Lin Y, Wang H and Ma M (2023) An adaptive hybrid deep reinforcement learning model for dynamic task offloading in edge computing. IEEE Internet of Things Journal 10(6), 5123–5136.

Zhang Y, Mao Y and Zhang J (2021) Reinforcement learning-based task offloading and resource allocation for mobile edge computing. IEEE Transactions on Vehicular Technology 70(2), 1307–1320.

