Towards Efficient Resource Management in MEC: A Hybrid TRPO-SAC Solution for Computing, Caching, and Pushing

1st Divyanshu Bathla
Department of CSE, IIT Guwahati
b.divyanshu@iitg.ac.in

2nd Innocent Dengkhw Mochahari
Department of CSE, IIT Guwahati
innocent@iitg.ac.in

3rd Evanstar Field War
Department of CSE, IIT Guwahati
evanstar@iitg.ac.in

4th Harsh Kumar Modi
Department of CSE, IIT Guwahati
harsh.modi@iitg.ac.in

***Abstract*—In Mobile Edge Computing (MEC) networks, optimising computing, caching, and data pushing is critical to meet the low-latency and high-reliability demands of modern mobile applications. Recent advancements leverage reinforcement learning (RL) techniques, such as the Soft Actor-Critic (SAC) algorithm, to effectively manage these resources through continuous learning. However, SAC's performance is limited by exploration inefficiencies in high-dimensional and highly constrained environments like MEC. In this study, we propose an enhanced RL approach that integrates Trust Region Policy Optimization (TRPO) with SAC to jointly optimize computing, pushing, and caching in MEC networks. By incorporating TRPO's constraint-aware updates, our method improves policy stability and enhances the accuracy of resource allocation. Experimental evaluations demonstrate that our hybrid TRPO-SAC approach achieves superior performance in terms of latency reduction, cache hit rate, and computational efficiency compared to standalone SAC and conventional heuristic methods. This improvement validates the efficacy of our approach for adaptive and robust MEC resource management, offering a promising solution for future large-scale, dynamic network environments.**

***Index Terms*—TRPO-SAC, Mobile Edge Computing (MEC), Joint Optimization**

I. Introduction

The proliferation of smart mobile devices has catalyzed the emergence of data-intensive applications such as virtual reality (VR), augmented reality (AR), online gaming, and autonomous driving. These applications impose stringent requirements on communication latency and computational resources, creating significant challenges for mobile network operators who must balance resource efficiency with user-perceived quality of experience (QoE). To address these challenges, mobile edge computing (MEC) has emerged as a promising solution, enabling the deployment of computational and caching resources at the network edge—closer to end-users via access points, base stations (BSs), and mobile devices. By leveraging MEC, networks can reduce latency, offload traffic from core infrastructure, and better serve the demanding requirements of modern mobile applications.

A. Prior Art in Joint Optimization for MEC

In MEC systems, caching, pushing, and computing are crucial components for minimizing transmission costs and reducing access delays. Caching can enhance bandwidth utilization by placing popular content closer to users for repeated access, with strategies ranging from static caching, based on content popularity, to dynamic caching, which adapts in real-time to changing user demands. Despite these advances, standalone caching techniques lack the foresight to proactively push content based on anticipated demand, limiting their effectiveness in high-traffic scenarios. Proactive caching combined with pushing has shown potential to improve system performance, as in scenarios where content is distributed during off-peak periods to prepare for future demand surges. However, most existing studies focus solely on caching and pushing, overlooking the computational demands of emerging applications, particularly those requiring real-time processing such as VR and autonomous driving.

The integration of computing capabilities with caching and pushing—often referred to as "3C" (computing, caching, and communication) optimization—has been a growing area of research in MEC. Studies have shown that combining caching and computation at MEC servers can reduce latency and energy consumption. However, many traditional 3C designs still rely on static caching and overlook the advantages of dynamic caching and pushing, which could further improve efficiency and adaptability in MEC networks.

B. Deep Reinforcement Learning in MEC Optimization

In recent years, deep reinforcement learning (DRL) has emerged as a powerful tool for solving complex optimization problems in MEC, particularly those involving large action spaces and dynamic environments. Techniques like Soft Actor-Critic (SAC) [1] have gained popularity for their effectiveness in continuous control tasks, as SAC's maximum entropy framework allows for balanced exploration, crucial in dynamic MEC systems. SAC has been used to tackle various MEC optimization challenges, such as joint caching and computation offloading, with notable success in improving resource allocation. However, the curse of dimensionality and the need for stable policy updates limit the efficiency of standard DRL methods in large-scale MEC settings.

To address these challenges, we propose an advanced hybrid approach that combines Trust Region Policy Optimization (TRPO) with SAC [1]. TRPO's constraint-aware policy updates enhance the stability of SAC, enabling more accurate and efficient resource allocation in MEC. By integrating TRPO with SAC, we aim to address both the exploration limitations and policy instability issues inherent in high-dimensional, constraint-sensitive MEC tasks. Our hybrid approach seeks to provide an adaptive solution that dynamically manages caching, pushing, and computing resources in real-time, with improved accuracy and performance over standalone methods.

C. Contributions

This work proposes a novel hybrid approach that combines Trust Region Policy Optimization (TRPO) with Soft Actor-Critic (SAC) [1] to enhance resource management in Mobile Edge Computing (MEC) networks through the joint optimization of computing, caching, and pushing tasks. Our key contributions are as follows:

- We develop a hybrid reinforcement learning (RL) model that integrates TRPO's stable policy updates with SAC's entropy-driven exploration. This synergy addresses exploration inefficiencies and policy instability in dynamic, high-dimensional MEC environments, enabling more accurate adaptation to fluctuating user demands.
- Our model provides a unified approach to optimize caching, computation, and data pushing resources, improving latency and bandwidth efficiency for modern mobile applications. This joint optimization directly enhances user-perceived quality of experience (QoE) by better managing MEC resources.
- Through extensive experiments, we demonstrate that the TRPO-SAC hybrid significantly reduces latency and enhances cache hit rates and computational efficiency over both standalone SAC and traditional heuristic approaches, underscoring its effectiveness in meeting MEC network performance targets.

II. Literature Survey

In this section, we investigate relevant research on adaptive video streaming without and with super-resolution.

A. Adaptive Video Streaming without Super-Resolution

Dynamic Adaptive Streaming over HTTP (DASH) has become a key technology for improving the Quality of Experience (QoE) of viewers by selecting the optimal bitrate based on network and device conditions. In [2], a buffer-based algorithm is proposed that determines bitrates solely based on the playback buffer occupancy at the video player. Yin et al. [3] introduce a Model Predictive Control (MPC)-based algorithm, which aims to maximize QoE over a horizon of five future chunks, factoring in both throughput and playback buffer information. The authors in [4] propose Pensieve, a neural network-based model that selects bitrates for future chunks using observations collected by video players.

These ABR algorithms enhance the viewer's QoE by adjusting the source transmission bitrate according to network conditions. However, they remain highly dependent on network quality and typically offer low-resolution videos when network conditions are poor. To address this limitation, this paper introduces Video Super-Resolution (VSR) into DASH, jointly optimizing both Source Transmission Resolution (STR) and VSR-Reconstructed Resolution (VRR) to reduce the reliance on network conditions.

B. Adaptive Video Streaming with Super-Resolution

The VSR technique enables the recovery of high-resolution video frames from low-resolution counterparts. In previous works such as NAS [5], SRAVS [6], and LiDeR [7], VSR has been incorporated into adaptive video streaming on the client side. However, VSR can be computationally intensive. Given the limited computational resources available on user devices, SRAVS adopts SRCNN, a lightweight super-resolution model with just three convolutional layers, while [7] proposes a lightweight VSR network specifically tailored for user devices, named LiDeR. Despite these efforts, the performance of VSR is still constrained by the computing capabilities of user devices.

To overcome this limitation, VISCA [8] presents an edge-assisted video delivery framework that integrates VSR with video caching at the edge, coupled with a DRL-based ABR algorithm to enhance video quality. Cai et al. [9] propose a VSR-based adaptive video streaming scheme in satellite backhaul wireless networks, while the authors in [10] introduce LiveSR, a framework for super-resolution live streaming.

Many of the existing methods that combine VSR with DASH at the edge server fail to address the mismatch between the available computational resources and the VSR reconstruction load, focusing primarily on regulating the VSR-reconstructed resolution without accounting for these computational limitations.

Wenshu Huang et al. [11] introduce **SuperABR**, a VSR-assisted adaptive video streaming method aimed at enhancing Quality of Experience (QoE) under limited network conditions. Conventional adaptive bitrate (ABR) strategies struggle in low-bandwidth environments, often compromising video quality. SuperABR, however, integrates a deep reinforcement learning (DRL) algorithm to optimize factors like video quality, rebuffering time, and quality consistency. This system dynamically adjusts source transmission resolution (STR) and VSR-reconstructed resolution (VRR) at the edge, considering historical VSR delays and available computing resources. Queue-learning techniques enhance cache management, improving streaming performance significantly over conventional ABR approaches. Experiments show that SuperABR boosts QoE by 20%-76% compared to baseline algorithms, though it introduces minor increases in jitter. Future work targets system adaptation for 5G networks to further improve streaming performance.

The work by Kevin Nassiside et al. [12] focus on adaptive streaming algorithms that optimize buffering and video quality

by incorporating dynamic buffering management and user behavior analytics. Recognizing buffering as a persistent challenge in live event streaming, even with modern technologies like MPEG-DASH and HLS, the authors propose a model that adjusts based on real-time simulation of viewing behaviors and network conditions. Metrics such as play rate, pause duration, buffer health, and quality adaptation were examined, demonstrating that strategic buffer management and behavioral insights can significantly enhance user experience in live streaming.

A comprehensive review of reinforcement learning (RL) in wireless communications, emphasizing single-agent and cooperative multi-agent reinforcement learning (MARL) frameworks was provided by Amal Feriani et al. [13]. The review addresses the application of these techniques to various wireless communication challenges, including Multi-Access Edge Computing (MEC), UAV control, and MIMO systems. MARL's potential in enabling adaptive, cooperative systems for next-generation networks is explored, highlighting future research needs in areas like safety, privacy, and algorithmic efficiency. The authors aim to bridge theoretical advancements with real-world implementations to support robust, scalable wireless systems.

Wonjun Kim et al. [14] explore the application of deep learning (DL) to wireless channel estimation, a critical aspect of modern communications. They review challenges in adapting DL for channel state feedback and emphasize the importance of careful model selection, data generation, and neural network design, particularly in high-frequency bands like millimeter-wave and terahertz for 6G technologies. The authors present experimental results demonstrating improvements in channel prediction through DL-based methods and discuss the necessity of future DL education and training to meet 6G demands.

Andreas Presas et al. [15] propose a Q-learning-based solution for **collision management** in vehicle networks using the IEEE 802.11p protocol. The method optimizes vehicle communication behavior, enhancing data packet exchange and minimizing bandwidth waste. By incorporating a collective contention estimation mechanism, this approach achieves faster convergence, better throughput, and improved fairness.

S. Aradi et al. [16] review state-of-the-art strategies for **motion control and trajectory planning** in autonomous driving, including car-following, lane-keeping, and dense traffic navigation. They classify solutions by task and level of autonomy, discussing methods like grid-based and camera-based models.

R F. Atallah et al. [17] develop **PEARL** (Protocol for Energy-efficient Adaptive Scheduling using Reinforcement Learning), which optimizes downlink traffic scheduling in vehicular networks. PEARL enables RSUs to create energy-efficient scheduling policies that maximize service requests during discharge phases. Simulation results indicate that PEARL significantly improves both QoE and throughput compared to heuristic methods, demonstrating its potential for effective network management.

In the recent work of Xiangyu et. al [1], the authors formulate the joint optimization problem as an *infinite-horizon discounted-cost Markov decision process (MDP)*. This structure allows for dynamic decision-making to optimize costs over time. They propose a *deep reinforcement learning (DRL)* framework based on *Soft Actor-Critic (SAC)* learning to handle the high-dimensional action space, which is a challenge in MEC due to the multiple functions (computing, caching, pushing) that must be jointly managed. To address dimensionality, they relax the action space from discrete to continuous and apply vector quantization to manage discrete decisions. SAC is chosen for its efficiency in continuous-action settings, its stability, and its ability to handle high-dimensional spaces through entropy maximization. The SAC model allows the MEC network to predict future user demands and proactively push or cache content based on these predictions, improving system efficiency.

III. System Model

To illustrate the system, we consider a straightforward mobile-edge network with a single MEC server and one mobile device (see Fig. 1). This model can extend to multiuser settings by summing objective functions across users and accounting for overall communication and computing resource limits. The MEC server has ample cache space, capable of pre-storing input and output data for all tasks requested by the mobile device, while the mobile device's cache is limited, with a maximum capacity denoted as C (in bits). The mobile device is equipped with multiple computing cores, each operating at a frequency of f_D (in cycles/s), with a total of M cores available. The system runs over an infinite time horizon, divided into discrete time slots indexed by $t = 0, 1, 2, \ldots$, each lasting τ seconds. At the beginning of each slot, the mobile device submits one task request that is time-sensitive and must be fulfilled within the slot. These tasks are classified as delay-sensitive to maintain optimal user experience, especially for applications like AR/VR, real-time communication, and streaming, which are highly delay-sensitive. Due to device mobility, the link quality between the mobile device and the server varies, modeled using the signal-to-noise ratio (SNR) based on Shannon theory. This system is optimized to jointly handle pushing, caching, and computing functions to minimize the network's computation and transmission costs, while ensuring timely and efficient task execution.

A. Task Model

The mobile device may request up to F tasks, represented by the task set $F = \{1, 2, \ldots, f, \ldots, F\}$. Each task $f \in F$ is defined by a 4-item tuple $\{I_f$ (input data size in bits), O_f (output data size in bits), w_f (computation cycles per bit), and τ (maximum latency in seconds)$\}$. Here, I_f represents the size of the input data retrievable from the Internet, which can be cached. O_f represents the size of the output data after computation. The parameters w_f and τ represent the computational requirement per bit and the maximum allowable delay, respectively.

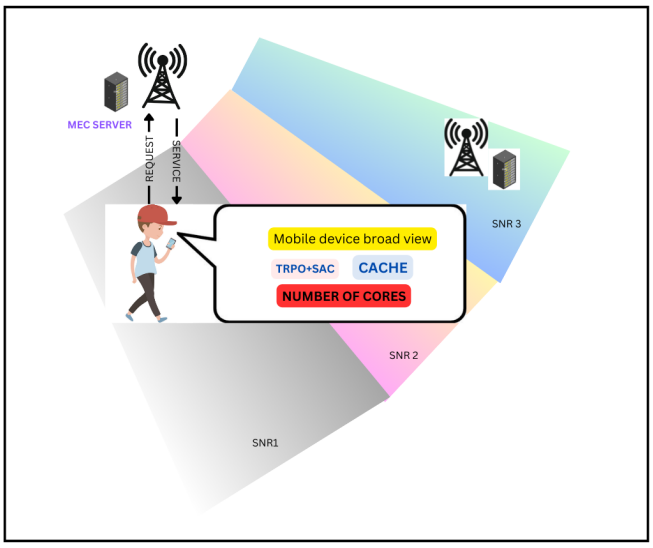


Fig. 1. Figure Depicts a mobile-edge computing (MEC) network with one MEC server and a single mobile device, the model assumes that the mobile device is moving slowly, such as an individual carrying a smartphone. The quality of the communication link between the device and the MEC server is represented by the signal-to-noise ratio (SNR), which may vary over time due to the device's movement. [1]

B. System State

Request State: During each time slot t, the mobile device requests one task, represented by the state $A(t) \in F$, where $A(t) = f$ indicates that task f in set F is being requested. The size of F is F. Task requests and transitions are modeled using a first-order F-state Markov chain [18] [19] $A(t) : t = 0, 1, 2, \ldots$. In this model, each state corresponds to a unique task, and transitions depend only on the current state. The probability of transitioning to state $j \in F$ at time $t+1$, given that the current state is $i \in F$, is $\Pr[A(t+1) = j \mid A(t) = i]$. The matrix $Q = (q_{i,j})_{i \in F, j \in F}$ represents these transition probabilities, with $q_{i,j} = \Pr[A(t+1) = j \mid A(t) = i]$. For flexibility, an irreducible Markov chain is assumed, so any state is reachable from any other with a non-zero probability. The steady-state distribution of $A(t)$, denoted as $p = (p_f)_{f \in F}$, satisfies $p_f = \lim_{t \to \infty} \Pr[A(t) = f]$ and $p_f = \sum_{i \in F} p_i q_{i,f}$ for all $f \in F$.

Cache State: Define $S_I^f(t) \in \{0, 1\}$ as the cache state indicator for task f's input data in the mobile device, where $S_I^f(t) = 1$ if the input data for task f is cached, and $S_I^f(t) = 0$ otherwise. Similarly, $S_O^f(t) \in \{0, 1\}$ represents the cache state for task f's output data, where $S_O^f(t) = 1$ if cached and $S_O^f(t) = 0$ otherwise. The mobile device's cache capacity C (in bits) imposes the constraint:

$$\sum_{f=1}^{F} I_f S_I^f(t) + O_f S_O^f(t) \leq C \tag{1}$$

This ensures the combined size of all cached input and output data does not exceed the device's cache capacity. Let the cache state at time t be $S(t) = (S_I^f(t), S_O^f(t))_{f \in F} \in S$, with S representing the cache state space of the device, bounded by $((FN_{\min}), (FN_{\max}))$, where $N_{\min} = \left\lfloor \frac{C}{\max(I_f, O_f)} \right\rfloor$ and $N_{\max} = \left\lfloor \frac{C}{\min(I_f, O_f)} \right\rfloor$.

System State: The complete system state at each time slot t comprises the task request and cache state, represented as $X(t) = (A(t), S(t)) \in F \times S$.

C. System Actions

Reactive Computation Action: In each time slot t, the system incurs reactive transmission bandwidth cost $B_R(t)$ and computation energy cost $E_R(t)$. Based on the state $X(t) = (A(t), S(t))$, task $A(t)$ is served as follows:

1) If $S_O^{A(t)}(t) = 1$, the output for task $A(t)$ is cached locally, so the task can be accessed without transmission or computation, resulting in zero delay and cost.
2) If $S_I^{A(t)}(t) = 1$ and $S_O^{A(t)}(t) = 0$, the task can be computed using locally cached input data. Define $c_{R,f}(t)$ as the number of cores used to process task f reactively. For task $A(t)$, completion within τ requires $\frac{I_{A(t)} w_{A(t)}}{\tau} \leq c_{R,A(t)}(t) f_D$. The computation cost $E_R(t)$ is $\mu c_{R,A(t)}^2(t) f_D^2 I_{A(t)} w_{A(t)}$, where μ is a chip architecture constant.
3) If both $S_I^{A(t)}(t) = 0$ and $S_O^{A(t)}(t) = 0$, the device downloads input data before computation. Let $\text{SNR}(t)$ be the SNR for data transmission at time t. The latency is $\frac{I_{A(t)}}{B_R(t) \log_2(1+\text{SNR}(t))} + \frac{I_{A(t)} w_{A(t)}}{c_{R,A(t)}(t) f_D}$. To meet the latency constraint, the minimum bandwidth cost is $B_R(t) = \frac{I_{A(t)}}{(\tau - \frac{I_{A(t)} w_{A(t)}}{c_{R,A(t)}(t) f_D}) \log_2(1+\text{SNR}(t))}$, and the computation cost is $E_R(t) = \mu c_{R,A(t)}^2(t) f_D^2 I_{A(t)} w_{A(t)}$.

Proactive Transmission (Pushing): Let $b_f(t) \in \{0, 1\}$ be the decision to push input data for task f to the mobile device, where $b_f(t) = 1$ indicates pushing. The proactive transmission cost $B_P(t)$ satisfies the constraint:

$$B_P(t) = \sum_{f=1}^{F} \frac{I_f b_f(t)}{\tau \log_2(1 + \text{SNR}(t))} \tag{2}$$

Cache Update: The cache update action for each task f is defined by $\Delta s_I^f(t) \in \{-1, 0, 1\}$ for input data and $\Delta s_O^f(t) \in \{-1, 0, 1\}$ for output data, with rules to ensure valid cache additions and removals without exceeding cache capacity.

System Action: At each time slot, the complete action set comprises reactive computation, pushing, and cache updates, denoted as $(c_R, b, \Delta s) \in A(X)$.

D. System Cost

The overall cost at each time slot combines transmission and computation costs. The total transmission cost $B(t)$ includes both proactive and reactive transmission costs:

$$B(t) = B_R(t) + B_P(t) \tag{3}$$

while the computation cost $E(t)$ is given by

$$E(t) = E_R(t) \tag{4}$$

The combined system cost is

$$B(t) + \lambda E(t) \tag{5}$$

where λ balances the two cost components.

IV. TRPO with Soft Actor-Critic Learning

A. TRPO-SAC System State and Action

The system state x of the TRPO-SAC algorithm is defined to align with the system state X in the formulated problem, where $x = X = (A(t), S(t))$ and has a vector size of $2F+1$.

Since SAC primarily targets continuous-action problems, we adjust the action space $(c_R, b, \Delta s)$ of the formulated problem, which includes discrete components, into a continuous representation. To achieve this, we define the TRPO-SAC system action as a continuous analog of the original system action space:

$$a = (\bar{c}_R, \bar{b}, \bar{\Delta s}) \in \Pi(\bar{X}) \triangleq \Pi_{\bar{R}_C}(X) \times [0,1]^F \times \Pi_{\bar{\Delta s}}(X).$$

Here, $\Pi_{\bar{R}_C}(X) \triangleq \{(c_{R,f})_{f\in F} \in [0,M]^F : (2)\}$, and

$$\Pi_{\bar{\Delta s}}(X) \triangleq \left\{ \begin{array}{c} \left(\Delta s_f^I, \Delta s_f^O\right)_{f\in F} \in [-1,1]^F \times [-1,1]^F : \\ (9),\ (10),\ (11) \end{array} \right\}. \tag{6}$$

Since $\bar{c}_R = \{(\bar{c}_{R,f})_{f\in F}\}$ must be zero for $f \in F \setminus A(t)$, the action space simplifies by excluding cores for non-requested tasks. Therefore, the action a can be expressed as:

$$a = \left(\bar{c}_{A(t)}, \bar{b}, \bar{\Delta s}\right),$$

with a vector size of $3F$.

B. TRPO-SAC Learning

TRPO-SAC incorporates SAC's entropy maximization with TRPO's trust region updates for stable learning. TRPO constrains policy updates via a KL-divergence limit, ensuring gradual policy shifts to avoid instability. The objective function maximized by TRPO is given by:

$$J_{\text{TRPO}}(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(x_t,a_t)\sim\rho_\pi}\left[r(x_t,a_t)\right], \tag{7}$$

subject to $D_{\text{KL}}(\pi_{\text{old}} \| \pi) < \delta$, with δ as the trust region threshold.

The SAC algorithm is designed to maximize the entropy-regularized reward:

$$J_{\text{SAC}}(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(x_t,a_t)\sim\rho_\pi}\left[r(x_t,a_t) + \alpha H(\pi(\cdot|x_t))\right], \tag{8}$$

where the entropy term $H(\pi(\cdot|x_t)) = \mathbb{E}_{a_t}[-\log\pi(a_t|x_t)]$ encourages exploration.

C. Policy and Value Function Updates

The soft Q-function $Q_\theta(x_t,a_t)$ and policy $\pi_\phi(a_t|x_t)$ are parameterized by neural networks. The Q-function parameters θ are updated by minimizing the soft Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(x_t,a_t)\sim\mathcal{D}}\Bigg[\frac{1}{2}\Big(Q_\theta(x_t,a_t) - \Big(r(x_t,a_t) + \gamma\mathbb{E}_{x_{t+1}\sim p}[V_{\bar{\theta}}(x_{t+1})]\Big)\Big)^2\Bigg]. \tag{9}$$

where $\mathcal{D}$ is the replay buffer. The soft value function $V_{\bar{\theta}}(x_t)$ is defined as:

$$V_{\bar{\theta}}(x_t) = \mathbb{E}_{a_t\sim\pi}\left[Q_{\bar{\theta}}(x_t,a_t) - \alpha\log\pi(a_t|x_t)\right], \tag{10}$$

with $\bar{\theta}$ representing the target Q-function parameters updated by an exponentially moving average of θ to stabilize training.

The policy parameters ϕ are updated by minimizing the expected KL-divergence:

$$J_\pi(\phi) = \mathbb{E}_{x_t\sim\mathcal{D}}\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha\log\pi_\phi(a_t|x_t) - Q_\theta(x_t,a_t)\right]. \tag{11}$$

Using a Gaussian distribution, the policy π_ϕ is parameterized as:

$$a_t = f_\phi(\epsilon_t; x_t), \tag{12}$$

where ϵ_t is sampled from a Gaussian distribution. The gradient of $J_\pi(\phi)$ is approximated by:

$$\begin{aligned}\nabla_\phi J_\pi(\phi) = {} & \nabla_\phi\alpha\log(\pi_\phi(a_t|x_t)) \\ & + \nabla_{a_t}\Big[\alpha\log(\pi_\phi(a_t|x_t)) \\ & - Q_\theta(x_t,a_t)\Big]\nabla_\phi f_\phi(\epsilon_t;x_t).\end{aligned} \tag{13}$$

Through alternating updates of policy evaluation and improvement, TRPO-SAC aims to optimize the maximum-entropy objective, combining stable trust-region updates and entropy-regularized exploration.

V. Evaluation Matrices

The proposed system is built on the proactive transmission and dynamic-computing-frequency reactive service with cache, referred to as PTDFC. For comparison, we have selected the following baselines:

- **Most-recently-used proactive transmission and least-recently-used cache replacement (MRU-LRU)**: This is a heuristic algorithm [20], [21], where, at each time slot, the requested task is reactively served, and the input data of the most-recently-used task is proactively cached. When the cache is full, the input data cache of the least-recently-used task is replaced. We choose to cache only the input data, excluding the output data (post-calculation), due to the common scenario where output data tends to be larger in size than input data. This size difference makes caching output data less efficient in the heuristic design for the purpose of reducing overall costs.

The number of computing cores being used is fixed at $0.75M$.

- **Most-frequently-used proactive transmission and least-frequently-used cache replacement (MFU-LFU)**: This algorithm is similar to the MRU-LRU algorithm, except that the most/least recently used task is replaced with the most/least frequently used task [20], [21].
- **Dynamic-computing-frequency reactive service with no cache (DFNC)**: This algorithm provides reactive service to the requested task, where the mobile device first downloads the input data from the MEC server and then computes it to obtain the output data.
- **Dynamic-computing-frequency reactive service with cache (DFC)**: This algorithm provides reactive service to the requested task, with the option of caching the input and output data into the limited capacity.
- **Proactive Transmission (PT)**: This is the previous implementation of the system, which utilizes SAC implementation for proactive transmission techniques and dynamic-computing-frequency reactive service. As a baseline, it will be compared against the new TRPO-SAC-based approach to evaluate performance improvements.

It is important to note that the DFC, DFNC, and PTDFC algorithms are all implemented with the SAC algorithm. As a result, we refer to them as "SAC-enabled algorithms" in the following analysis.

VI. Results and Discussion

In this section, we present the results of comparing the TRPO-SAC-based approach with the baseline algorithms, including the previous **Proactive Transmission** approach and the SAC approach. We focus on key metrics such as transmission cost, computational cost, and overall system performance.

A. Transmission and Computational Costs

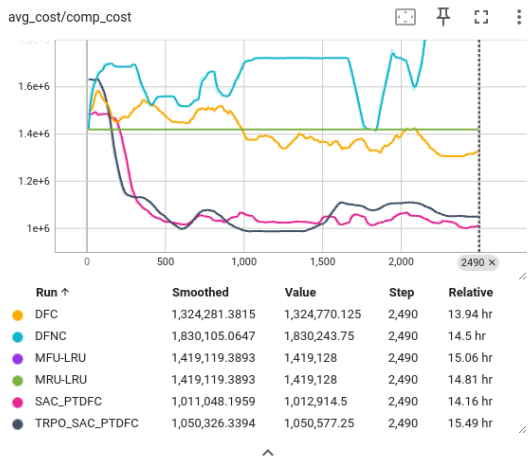
The comparison between TRPO-SAC and SAC is based on the reduction in both transmission and computational costs. As shown in Fig. 2, the implementation of TRPO-SAC leads to a slight reduction in both transmission cost and computational cost compared to using only SAC.

- **Transmission Cost:** The use of TRPO-SAC shows a modest improvement in reducing transmission costs compared to SAC. By leveraging the trust region and entropy-based exploration offered by TRPO, the system is able to make more informed decisions regarding the transmission of data, leading to a slight reduction in the number of transmission events required to complete the task. This efficiency is particularly noticeable in systems with a high volume of requests or complex task processing.
- **Computational Cost:** TRPO-SAC also results in a slight reduction in computational cost over SAC alone. The combined benefits of TRPO's stable policy updates and SAC's exploration capabilities enable the system to converge to an optimal solution more efficiently. The result is that fewer computational resources are consumed, while still maintaining high task processing accuracy and performance.

While both costs are slightly reduced, the difference between TRPO-SAC and SAC is not drastic, but it indicates that the combination of TRPO's stability with SAC's entropy-based exploration leads to a more efficient trade-off between transmission and computation. This is especially advantageous in systems with limited resources where balancing both aspects is crucial.

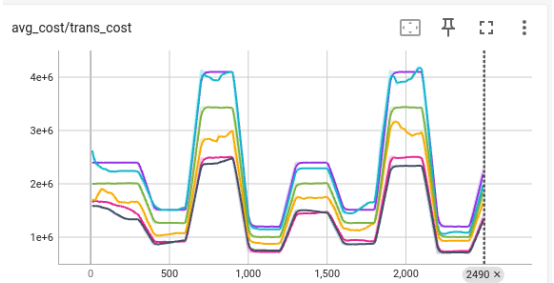
B. Overall System Performance

When considering the overall system performance, TRPO-SAC exhibits a higher level of efficiency, as it outperforms both SAC and Proactive Transmission in the long-term learning efficiency and the adaptability of the system to varying network and computation load conditions. As shown in Fig. 2 and Fig. 3, TRPO-SAC consistently achieves better task completion times while maintaining a lower operational cost in terms of both transmission and computation.



Run	Smoothed	Value	Step	Relative
DFC	1,324,281.3815	1,324,770.125	2,490	13.94 hr
DFNC	1,830,105.0647	1,830,243.75	2,490	14.5 hr
MFU-LRU	1,419,119.3893	1,419,128	2,490	15.06 hr
MRU-LRU	1,419,119.3893	1,419,128	2,490	14.81 hr
SAC_PTDFC	1,011,048.1959	1,012,914.5	2,490	14.16 hr
TRPO_SAC_PTDFC	1,050,326.3394	1,050,577.25	2,490	15.49 hr

Fig. 2. Comparison of Computation Cost of TRPO-SAC vs. Other Baselines.



Run	Smoothed	Value	Step	Relative
DFC	1,639,338.3488	1,702,153.125	2,490	13.94 hr
DFNC	1,975,592.2767	2,057,173	2,490	14.5 hr
MFU-LRU	2,185,683.591	2,275,681	2,490	15.06 hr
MRU-LRU	1,836,863.2729	1,911,952.125	2,490	14.81 hr
SAC_PTDFC	1,340,551.4347	1,394,826.75	2,490	14.16 hr
TRPO_SAC_PTDFC	1,302,818.7961	1,356,886.25	2,490	15.49 hr

Fig. 3. Comparison of Transmission cost of TRPO-SAC vs. Other Baselines.

The results demonstrate that the TRPO-SAC method provides a more balanced and efficient solution compared to SAC, validating the advantages of incorporating the trust region and entropy maximization techniques.

C. Discussion

Although the performance improvements are slight, the TRPO-SAC-based approach shows significant advantages in systems requiring long-term decision-making with fluctuating network and computation resources. The combination of TRPO and SAC addresses the stability-exploration trade-off, leading to more efficient resource utilization without compromising performance. This is a critical factor in real-time applications where maintaining a balance between transmission and computation is essential for optimizing operational costs.

REFERENCES

[1] Xiangyu Gao, Yaping Sun, Hao Chen, Xiaodong Xu, and Shuguang Cui. Joint computing, pushing, and caching optimization for mobile edge computing networks via soft actor-critic learning. *IEEE Internet of Things Journal*, 2023.

[2] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[3] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.

[4] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.

[5] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 645–661, 2018.

[6] Yinjie Zhang, Yuanxing Zhang, Yi Wu, Yu Tao, Kaigui Bian, Pan Zhou, Lingyang Song, and Hu Tuo. Improving quality of experience by adaptive video streaming with super-resolution. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1957–1966. IEEE, 2020.

[7] Ekrem Çetinkaya, Minh Nguyen, and Christian Timmerer. Lider: Lightweight dense residual network for video super-resolution on mobile devices. In *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5. IEEE, 2022.

[8] Aoyang Zhang, Qing Li, Ying Chen, Xiaoteng Ma, Longhao Zou, Yong Jiang, Zhimin Xu, and Gabriel-Miro Muntean. Video super-resolution and caching—an edge-assisted adaptive video streaming solution. *IEEE Transactions on Broadcasting*, 67(4):799–812, 2021.

[9] Haoyuan Cai, Wenpeng Jing, Xiangming Wen, Zhaoming Lu, and Zhifei Wang. Mec-based qoe optimization for adaptive video streaming via satellite backhaul. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–7. IEEE, 2021.

[10] João Da M Liborio Filho, Maiara de Souza Coelho, and Cesar AV Melo. Super-resolution on edge computing for improved adaptive http live streaming delivery. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pages 104–110. IEEE, 2021.

[11] Wenshu Huang, Yongyi Ran, Jie Rao, Jiangtao Luo, and Shuangwu Chen. Queue-learning-based qoe optimization for super-resolution-assisted adaptive video streaming. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pages 140–145. IEEE, 2023.

[12] Kevin Nassisid, Teef David, and Kassi Muhammad. Adaptive video streaming over 6g networks: Buffer control and user behavior analysis. *arXiv preprint arXiv:2407.05436*, 2024.

[13] Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 23(2):1226–1252, 2021.

[14] Wonjun Kim, Yongjun Ahn, Jinhong Kim, and Byonghyo Shim. Towards deep learning-aided wireless channel estimation and channel state information feedback for 6g. *Journal of Communications and Networks*, 25(1):61–75, 2023.

[15] Andreas Pressas, Zhengguo Sheng, Falah Ali, and Daxin Tian. A q-learning approach with collective contention estimation for bandwidth-efficient and fair access control in ieee 802.11 p vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(9):9136–9150, 2019.

[16] Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):740–759, 2020.

[17] Ribal F Atallah, Chadi M Assi, and Jia Yuan Yu. A reinforcement learning technique for optimizing downlink scheduling in an energy-limited vehicular network. *IEEE Transactions on Vehicular Technology*, 66(6):4592–4601, 2016.

[18] Yichen Qian, Rui Wang, Jun Wu, Bin Tan, and Haoqi Ren. Reinforcement learning-based optimal computing and caching in mobile edge network. *IEEE Journal on Selected Areas in Communications*, 38(10):2343–2355, 2020.

[19] Konstantinos Psounis, An Zhu, Balaji Prabhakar, and Rajeev Motwani. Modeling correlations in web traces and implications for designing replacement policies. *Computer Networks*, 45(4):379–398, 2004.

[20] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[21] Yaping Sun, Ying Cui, and Hui Liu. Joint pushing and caching for bandwidth utilization maximization in wireless networks. *IEEE Transactions on Communications*, 67(1):391–404, 2018.