Goodput Maximization for Large Language Model Edge Inference: A Two-Phase Maskable PPO Approach

Xiaojing Chen, Qi Zhang, Wei Ni, *Fellow, IEEE*, Shunqing Zhang, *Senior Member, IEEE*, and Yanzan Sun

***Abstract*—This paper presents a novel two-phase maskable proximal policy optimization (TP-MPPO) algorithm, which maximizes the system goodput counting request throughput with strict service level objective (SLO) compliance for large language model (LLM) inference services in wireless edge networks. In the first phase of TP-MPPO, we optimize the task offloading decisions by MPPO with action masking mechanism, effectively avoiding exploring invalid actions and reducing the action space. In the second phase, closed-form solutions are derived for uplink bandwidth allocation; a greedy algorithm is designed for downlink bandwidth allocation to provide immediate rewards for the MPPO in the next round. The two stages alternate till convergence. Simulation results demonstrate that TP-MPPO can improve the system reward by 33.3%–87.5% compared to its benchmarks and achieve the highest goodput.**

***Index Terms*—Large language model, edge inference, resource allocation, task offloading.**

I. Introduction

The rapid proliferation of large language models (LLMs) has surged inference demands, straining traditional cloud-based LLM inference services that struggle with throughput and latency [1]. Edge computing supplements cloud-based LLM inference by leveraging distributed resources to scale service capacity. Furthermore, deployment on nearby edge nodes effectively mitigates propagation latency.

Unlike conventional edge scenarios, LLM services are constrained by intensive workloads and GPU scarcity. At the model and system level, efficient inference techniques such as KV cache compression [2] and speculative decoding [3] reduce per-request compute overhead, while selective batching [4] improves server-side utilization. These techniques are complementary to network-level resource management, which requires jointly optimizing task offloading and bandwidth allocation across heterogeneous users and wireless edge nodes [5], [6]. In [5], LLM task offloading, computational, bandwidth and graphics memory resource allocation were optimized to minimize the average latency of all LLM tasks. In [6], LLM task distribution, computational and communication resource allocation were optimized to maximize the system utility of 6G networks. However, the studies in [5] and [6] overlooked throughput, a critical metric for LLM service systems.

Although some studies have targeted aggregate throughput [1], they have overlooked the fact that downstream applications have different latency requirements for user experience, leading to dramatically different service level objectives (SLOs) that need to be satisfied. The most widely used SLOs in LLM services are Time-to-First Token (TTFT), Time per Output Token (TPOT) and End-to-End (E2E) latency. Goodput [7], which captures request throughput under SLO attainment reflecting both cost and service quality, can be an effective metric for LLM service systems. To fill the void in goodput-oriented optimization, this paper presents a two-phase maskable proximal policy optimization (TP-MPPO) algorithm, which determines offloading and bandwidth allocation for edge LLM inference with GPU-constrained users, while preventing out-of-memory (OOM) failures caused by excessive batch sizes. The contributions of this paper are collated below.

- We model a wireless edge LLM inference system capturing full E2E latency pipeline and GPU memory constraints, enabling a realistic formulation of joint offloading and resource allocation.
- We formulate a mixed-integer nonlinear programming (MINLP) problem to maximize goodput, bridging the gap between raw throughput and service quality. The problem jointly optimizes task offloading and bandwidth allocation under SLO and memory capacity constraints.
- We propose TP-MPPO, which decomposes the MINLP into an MPPO-based offloading phase and an analytical bandwidth allocation phase with closed-form uplink and greedy downlink solutions. This algorithm provides immediate rewards that drive stable policy convergence, e.g., within 200 episodes.

Experiment results demonstrate that TP-MPPO improves the system reward by 33.3%–87.5% compared to its benchmarks and achieves the best goodput performance under diverse configurations of user and node numbers.

The rest of this paper is organized as follows. Section II provides the system model. Section III formulates the considered problem. The TP-MPPO algorithm is proposed in Section IV and evaluated numerically in Section V, followed by concluding remarks in Section VI.

II. System Model

We consider an LLM inference system comprising a service controller co-located with an access point (AP), J GPU-

†Work in the paper was supported by Shanghai Municipal Science and Technology Commission Foundation grants 25DP1500300 and 24DP1501001. *(Corresponding author: Yanzan Sun.)*

X. Chen, Q. Zhang, S. Zhang and Y. Sun are with the Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200444, China. Emails: {jodiechen, zhangqi_kawhi2, shunqing, yanzansun}@shu.edu.cn.

W. Ni is with the School of Engineering, Edith Cowan University, Perth, WA 6027, Australia. Email: wei.ni@ieee.org.

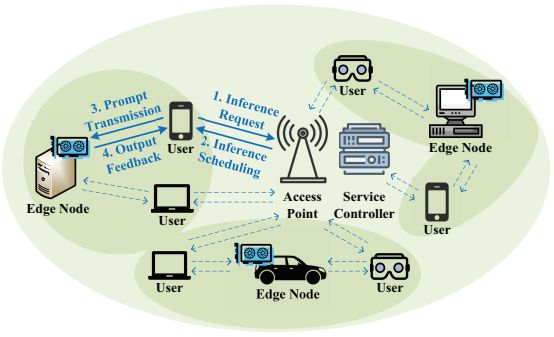


Fig. 1. Illustration of the system supporting LLM inference services.

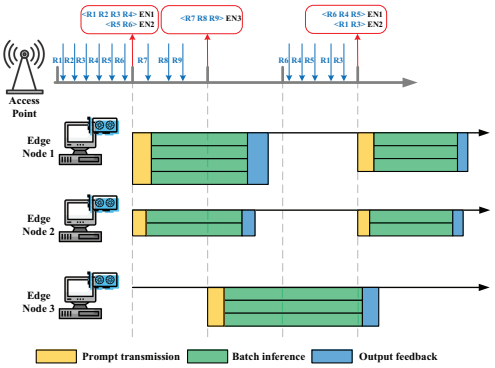


Fig. 2. Workflow of LLM inference system: Requests R1–R6 arrive in the first slot; the AP assigns R1–R4 to Node 1 and R5–R6 to Node 2. Subsequently, R7–R9 are allocated to Node 3, since Nodes 1 and 2 remain occupied with ongoing inference.

enabled edge nodes ($\mathcal{J} = \{1, \ldots, J\}$), and N users ($\mathcal{N} = \{1, \ldots, N\}$); see Fig. 1. These users generate LLM inference requests but cannot perform local inference due to lack of GPU resources; they need to offload these tasks to the edge nodes. The AP acts as a centralized scheduler and controller, collecting requests and making offloading decisions [1].

As depicted in Fig. 2, the AP schedules requests on the basis of time slots $\mathcal{T} = \{1, \ldots, K\}$. The slot length is denoted as τ, which acts as a macroscopic scheduling window rather than an orthogonal frequency division multiplexing (OFDM) symbol duration. Requests arriving in slot t are buffered at the AP and scheduled at the end of this slot. Each request from user n is denoted by $q_n^t = \left\langle s_n^{\text{in},t}, s_n^{\text{out},t}, \varsigma_n^t \right\rangle$, where $s_n^{\text{in},t}$ is the input prompt length, $s_n^{\text{out},t}$ is the maximum output sequence length, and ς_n^t defines the E2E delay bound. The AP decides whether to offload a request to node j or reject it. The offloading decision of request q_n^t is an integer variable $\alpha_n^t \in \mathcal{J} \cup \{0\}$, and $\alpha_n^t = 0$ indicates that the request is rejected and failed. After scheduling, users transmit their inference prompts to the designated edge nodes, which aggregate the prompts into a single batch, perform inference, and return the output results.

We assume each user initiates a new request only after receiving the previous response [4], with "thinking time" following an exponential distribution. The LLM inference process comprises two stages [8]: the *prefill stage*, which processes inputs in parallel to initialize the Key-Value (KV) cache and generate the first token; and the *decoding stage*, which utilizes the KV cache to auto-regressively generate subsequent tokens. Both stages rely on the Transformer architecture [9], with the key distinction that prefill computes self-attention across all input tokens concurrently, whereas decoding restricts attention to the new token and cached history. Let $o_n^t \leq s_n^{\text{out},t}$ denote the total number of output tokens generated for request q_n^t.

A. Communication Model

The communication model focuses on prompt transmission and output feedback, with negligible overhead from inference request and scheduling. We consider OFDM, with each edge node operating on non-overlapping frequency bands to avoid interference. Let $\beta_{nj}^{\text{up},t}, \beta_{nj}^{\text{down},t} \in [0, 1]$ denote the bandwidth fractions assigned by edge node j to user n ($\alpha_n^t = j$) for uplink and downlink transmissions, respectively. Per time slot t, the uplink transmit rate of user n is given by

$$R_{nj}^{\text{up},t} = \beta_{nj}^{\text{up},t} B_j^{\text{up}} \log_2\left(1 + \frac{p_{nj}^{\text{up},t} h_{nj}^{\text{up},t}}{\psi}\right), \tag{1}$$

where B_j^{up} is the uplink bandwidth of edge node j; ψ is the background noise power at the edge node; $p_{nj}^{\text{up},t}$ denotes the uplink transmit power of user n; and $h_{nj}^{\text{up},t}$ is the channel gain from user n to edge node j. The wireless channel consists of large-scale path loss and small-scale fading: $h_{nj}^{\text{up},t} = \xi_{nj}^{\text{up},t} \bar{h}_{nj}^{\text{up}}$, where $\xi_{nj}^{\text{up},t}$ follows an exponential distribution with unit mean corresponding to Rayleigh fading, and $\bar{h}_{nj}^{\text{up}} = (A_d \frac{3\times 10^8}{4\pi f_c d_{nj}})^{d_e}$ is the distance-dependent average channel gain, with A_d the antenna gain, f_c the carrier frequency, d_e the path loss exponent, and d_{nj} the distance between user n and edge node j. The channel gain $h_{nj}^{\text{up},t}$ is treated as slot-level effective average value under a quasi-static fading assumption, as widely adopted in wireless edge computing models [1], [5], [6]. Likewise, the downlink transmit rate is $R_{nj}^{\text{down},t} = \beta_{nj}^{\text{down},t} B_j^{\text{down}} \log_2(1 + \frac{p_{nj}^{\text{down},t} h_{nj}^{\text{down},t}}{\psi'})$, where B_j^{down} is the downlink bandwidth of edge node j; ψ' is the background noise power at user n; $p_{nj}^{\text{down},t}$ denotes the downlink transmit power of edge node j; and $h_{nj}^{\text{down},t}$ is the channel gain from edge node j to user n.

B. Latency and Memory Analysis

For request q_n^t (if user n generates a request during slot t), let $T_n^{\text{up},t}$ denote its uplink latency, and $D_n^{\text{in},t}$ denote the data size of its input prompt. We have $T_n^{\text{up},t} = D_n^{\text{in},t} / R_{nj}^{\text{up},t}$. Likewise, the downlink latency of q_n^t is $T_n^{\text{down},t} = D_n^{\text{out},t} / R_{nj}^{\text{down},t}$, where $D_n^{\text{out},t}$ is the data size of its output. Each uplink and downlink transmission is assumed to complete within one scheduling slot, over which the channel gain is treated as quasi-static. Extending the framework to account for transmissions spanning multiple slots is left for future work.

The E2E latency of user n's task is composed of five components. Since requests arriving at the AP are scheduled at the end of each time slot, a waiting time for scheduling $T_n^{\text{wait},t}$ is incurred at the AP. Upon finishing the prompt transmission which takes $T_n^{\text{up},t}$, user n may need to wait for other users assigned to the same node to complete their prompt transmissions before conducting batch inference, resulting in a batching wait time $T_n^{\text{stay},t}$ at the edge node. Let $\mathcal{N}_j^t = \{n | \alpha_n^t = j\}$ denote the set of requests assigned to edge node j, with batch size $b_j^t = |\mathcal{N}_j^t|$. During batch inference at edge

node j, all requests assigned to the same batch are processed together, and the batch completion latency is given by $T_j^{\mathrm{inf},t} = T_j^{\mathrm{P},t} + T_j^{\mathrm{D},t}$, where $T_j^{\mathrm{P},t}$ and $T_j^{\mathrm{D},t}$ denote the prefill and decoding latency, respectively. Following the analytical latency model for Transformer inference in [1], the prefill latency is $T_j^{\mathrm{P},t} = Lb_j^t(6s_j^t h^2 + (4(s_j^t)^2 h + 2s_j^t h^2) + 4s_j^t h\mathfrak{h})/F_j$ and the decoding latency is $T_j^{\mathrm{D},t} = Lb_j^t(o_j^t - 1)(6h^2 + (4(s_j^t + \frac{o_j^t}{2})h + 2h^2) + 4h\mathfrak{h})/F_j$. Here, s_j^t is the padded input length, $o_j^t = \max_{n\in\mathcal{N}_j^t} o_n^t$ is the batch-level output length, L is the number of Transformer layers, h is the Transformer hidden dimension, $\mathfrak{h}$ is the feed-forward network (FFN) hidden dimension, and F_j is the computational capacity (in FLOPs/s) of edge node j. The inference latency model captures the latency trends observed in practical inference systems, where larger batch sizes, longer sequences, and larger models increase computation latency, while stronger GPU resources reduce latency. Finally, the output feedback consumes $T_n^{\mathrm{down},t}$. The total E2E latency of user n is $T_n^{\mathrm{E2E},t} = T_n^{\mathrm{wait},t} + T_n^{\mathrm{up},t} + T_n^{\mathrm{stay},t} + T_j^{\mathrm{inf},t} + T_n^{\mathrm{down},t}$.

For batch inference, the memory footprint of edge node j is $m_j^t = m^{\mathrm{wts}} + m_j^{\mathrm{kv},t}$, which consists of the memory for model weights, m^{wts}, and the KV cache footprint, $m_j^{\mathrm{kv},t}$. The KV cache footprint $m_j^{\mathrm{kv},t} = 4hLb_j^t(s_j^t + o_j^t)$ [1].

III. Problem Formulation

Let a binary variable $\mathfrak{r}_n^t \in \{0,1\}$ denote the completion status of request q_n^t, where $\mathfrak{r}_n^t = 1$ indicates that request q_n^t is successfully completed, and $\mathfrak{r}_n^t = 0$ indicates otherwise. Request q_n^t is considered successfully completed ($\mathfrak{r}_n^t = 1$) if it is assigned to an edge node (i.e., not rejected, $\alpha_n^t \neq 0$), the batch inference finishes without an OOM failure, and the output is returned to the user. The system throughput can be given as $Y = \sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}} \mathfrak{r}_n^t$. Larger batch sizes enhance throughput; conversely, oversized batch sizes can trigger memory exhaustion, causing the inference to fail. Let M_j denote the memory capacity of edge node j, which depends on its GPU resources.

We define *goodput* as the number of completed requests satisfying the E2E latency SLO [7]. For request q_n^t, let $\mathfrak{g}_n^t \in \{0,1\}$ denote whether request q_n^t contributes to the goodput. $\mathfrak{g}_n^t = 1$ only if $\mathfrak{r}_n^t = 1$ and $T_n^{\mathrm{E2E},t} \leq \varsigma_n^t$. Requests that are rejected, fail due to OOM, or are completed but violate the SLO are not counted. The system goodput is defined as $Y' = \sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}} \mathfrak{g}_n^t$, a dimensionless metric representing the total number of SLO-compliant completed requests over the entire simulation horizon (i.e., all K time slots).

Our goal is to maximize the system goodput by optimizing offloading decisions and bandwidth allocation, accounting for the unique challenges of LLM inference driven by dynamic KV cache footprint and heterogeneous SLO requirements. Let m_j^t represent the memory footprint of edge node j for batch inference. The optimization problem is formulated as:

$$\mathcal{P}1: \quad \max_{\{\boldsymbol{\alpha}^t, \boldsymbol{\beta}^{\mathrm{up},t}, \boldsymbol{\beta}^{\mathrm{down},t}\}_t} \quad Y' \tag{2a}$$

$$\text{s.t.} \quad \sum_{n\in\mathcal{N}_j^t} \beta_{nj}^{\mathrm{up},t} \leq 1, \ \sum_{n\in\mathcal{N}_j^t} \beta_{nj}^{\mathrm{down},t} \leq 1, \quad \forall j,t, \tag{2b}$$

$$m_j^t \leq M_j, \quad \forall j,t. \tag{2c}$$

Here, $\boldsymbol{\alpha}^t = \{\alpha_n^t, \forall n\}$, $\boldsymbol{\beta}^{\mathrm{up},t} = \{\beta_{nj}^{\mathrm{up},t}, \forall n,j\}$, and $\boldsymbol{\beta}^{\mathrm{down},t} = \{\beta_{nj}^{\mathrm{down},t}, \forall n,j\}$. Constraint (2b) limits uplink and downlink bandwidth allocation, while Constraint (2c) ensures that each edge node's memory footprint remains within its capacity. Solving $\mathcal{P}1$ is challenging because it is an NP-hard MINLP problem involving non-convex and discontinuous binary decisions. Moreover, LLM batch inference creates strong coupling among users, where the latency and memory consumption of a batch depend on the specific combination of assigned tasks.

IV. Proposed TP-MPPO Scheme

We present TP-MPPO to address $\mathcal{P}1$. In the first phase, MPPO determines the offloading decisions. In the second phase, the remaining problem is decomposed into two subproblems. Closed-form solutions are derived for uplink bandwidth allocation, while downlink bandwidth allocation is solved using a greedy-based algorithm.

A. PPO Framework with Invalid Action Masking

Upon inspecting $\mathcal{P}1$, optimizing $\boldsymbol{\alpha}^t$ first makes the remaining problem over $\{\boldsymbol{\beta}^{\mathrm{up},t}, \boldsymbol{\beta}^{\mathrm{down},t}\}$ more tractable. We propose MPPO to tackle offloading decisions $\boldsymbol{\alpha}^t$. MPPO extends PPO with invalid action masking [10], enabling agents to avoid infeasible actions under state-dependent constraints and complex action spaces. The MPPO components are designed as follows.

*1) **State**:* The state space is $\boldsymbol{s}^t = \{q_n^t, h_{nj}^{\mathrm{up},t}, h_{nj}^{\mathrm{down},t}, I_j^t, \forall n,j\}$, where $I_j^t \in \{0,1\}$ indicates whether edge node j is idle, taking 1 if idle and 0 otherwise.

*2) **Action**:* To prevent invalid node selection, we introduce a binary mask vector $\mathbf{v}_n^t = \{v_{n,j}^t, \forall j\}$ for each user. Each element is dynamically generated based on the current environment: $v_{n,j}^t = 1$ when request q_n^t exists and node j is available at slot t (i.e., $I_j^t = 1$); otherwise, $v_{n,j}^t = 0$.

During actor network forward propagation, logits of invalid actions ($v_{n,j}^t = 0$) are set to $-\infty$. After the Softmax, their selection probability becomes zero. This Action Masking dynamically prunes the action space, ensuring $\boldsymbol{\alpha}^t$ is always executable. The resulting action space is $\boldsymbol{a}^t = \{\boldsymbol{\alpha}^t\}$.

*3) **Reward**:* The objective is to maximize the system goodput while penalizing memory overflow failures. The immediate reward r^t is formulated as a normalized utility given by $r^t = \frac{\nu_{\mathrm{good}}\sum_{n\in\mathcal{N}} \mathfrak{g}_n^t - \nu_{\mathrm{oom}} N^{\mathrm{oom},t}}{N^{\mathrm{req},t}}$, where $N^{\mathrm{oom},t}$ is the number of requests that failed due to memory overflow, and $N^{\mathrm{req},t}$ is the total number of requests generated during slot t. ν_{good} and ν_{oom} are weight coefficients.

MPPO selects the action $\boldsymbol{a}^t$ by pruning invalid choices via the mask $\mathbf{v}^t = \{\mathbf{v}_n^t\}_{n=1}^N$. For a given $\boldsymbol{\alpha}^t$, the subproblems are solved to determine $\boldsymbol{\beta}^{\mathrm{up},t}$ and $\boldsymbol{\beta}^{\mathrm{down},t}$. The complete action set $\{\boldsymbol{\alpha}^t, \boldsymbol{\beta}^{\mathrm{up},t}, \boldsymbol{\beta}^{\mathrm{down},t}\}$ then updates the environment, triggering the update of $\{I_j^t\}$ and the state transition from $\boldsymbol{s}^t$ to $\boldsymbol{s}^{t+1}$.

Specifically, the behavior actor network (with parameters $\boldsymbol{\theta}_{\mathrm{old}}$) generates action $\boldsymbol{a}^t$ under the masked policy $\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(\boldsymbol{a}^t|\boldsymbol{s}^t, \mathbf{v}^t)$. The critic network (with parameters $\boldsymbol{\omega}$) estimates the state value $V_{\boldsymbol{\omega}}(\boldsymbol{s}^t)$. Collected trajectories $(\boldsymbol{s}^t, \mathbf{v}^t, \boldsymbol{a}^t, r^t, \boldsymbol{s}^{t+1})$ are sampled from the replay buffer to update the target actor network $\boldsymbol{\theta}$ by minimizing $L_{\mathrm{CLIP}}^t(\boldsymbol{\theta}) =$

Algorithm 1: Optimal Solution to $\mathcal{P}3$

Initialize satisfied set $\mathcal{S}_j^t = \varnothing$, and cumulative bandwidth usage $\beta_{j,\text{sum}}^{\text{down},t} = 0$.
for *each user* $n \in \mathcal{N}_j^t$ **do**
  Calculate the minimum required bandwidth fraction $\beta_{nj,\min}^{\text{down},t}$.
end
Sort users in $\mathcal{N}_j^t$ based on $\beta_{nj,\min}^{\text{down},t}$ in ascending order to obtain the sorted index list $\mathcal{K} = \{k_1, \ldots, k_{|\mathcal{N}_j^t|}\}$.
for $i = 1$ *to* $|\mathcal{N}_j^t|$ **do**
  if $\beta_{j,\text{sum}}^{\text{down},t} + \beta_{k_i j,\min}^{\text{down},t} \le 1$ **then**
    Add user k_i to satisfied set: $\mathcal{S}_j^t \leftarrow \mathcal{S}_j^t \cup \{k_i\}$.
    Allocate minimum requirement: $\beta_{k_i j}^{\text{down},t} = \beta_{k_i j,\min}^{\text{down},t}$.
    Update usage: $\beta_{j,\text{sum}}^{\text{down},t} \leftarrow \beta_{j,\text{sum}}^{\text{down},t} + \beta_{k_i j,\min}^{\text{down},t}$.
  end
end
Obtain the optimal $\{\beta_{nj}^{\text{down},t}\}_{n\in\mathcal{N}_j^t}$.

$\hat{\mathbb{E}}_t[\min(g^t(\boldsymbol{\theta})\hat{A}^t, \text{clip}(g^t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}^t)]$, where $g^t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}^t|\boldsymbol{s}^t,\mathbf{v}^t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{a}^t|\boldsymbol{s}^t,\mathbf{v}^t)}$ is the masked-policy probability ratio, and ϵ is the clipping factor. $\hat{A}^t = \sum_{i=0}^{K-t}(\gamma)^i\delta^{t+i}$ is the advantage function calculated via generalized advantage estimation [11], where γ is the discount factor and $\delta^t = r^t + \gamma V_{\boldsymbol{\omega}}(\boldsymbol{s}^{t+1}) - V_{\boldsymbol{\omega}}(\boldsymbol{s}^t)$ represents the temporal-difference error. The critic network is updated by minimizing $L^t(\boldsymbol{\omega}) = \mathbb{E}_t[(\delta^t)^2]$.

B. Closed-Form Solutions to Uplink Bandwidth Allocation

Given $\boldsymbol{\alpha}^t$, the subset of users offloaded to edge node j is determined. For user $n \in \mathcal{N}_j^t$, the prompt transmission time $T_n^{\text{up},t}$ and waiting time $T_n^{\text{stay},t}$ are dictated by the slowest user in the batch, as batch inference requires all prompts uploaded before padding. Thus, the uplink bandwidth allocation problem at node j is to minimize the maximum uplink latency among its assigned users, expressed as $T_j^{\text{up},t} = \max_{n\in\mathcal{N}_j^t} T_n^{\text{up},t}$. The subproblem of uplink bandwidth allocation is given by

$$\mathcal{P}2: \min_{\{\beta_{nj}^{\text{up},t}\}_{n\in\mathcal{N}_j^t}} \quad T_j^{\text{up},t} \qquad \text{s.t.} \quad \sum_{n\in\mathcal{N}_j^t} \beta_{nj}^{\text{up},t} \le 1, \ \forall j,t. \tag{3}$$

According to the min-max fairness principle, the optimal strategy equalizes uplink latencies across all users in the batch. Using the auxiliary variable $\mathfrak{R}_{nj}^{\text{up},t} = B_j^{\text{up}} \log_2(1 + \frac{p_{nj}^{\text{up},t} h_{nj}^{\text{up},t}}{\psi})$ and equating $\frac{D_n^{\text{in},t}}{\beta_{nj}^{\text{up},t}\mathfrak{R}_{nj}^{\text{up},t}}$ for all users $n \in \mathcal{N}_j^t$, the optimal allocation is $(\beta_{nj}^{\text{up},t})^* = \frac{D_n^{\text{in},t}/\mathfrak{R}_{nj}^{\text{up},t}}{\sum_{i\in\mathcal{N}_j^t}(D_i^{\text{in},t}/\mathfrak{R}_{ij}^{\text{up},t})}$.

C. Optimal Solution to Downlink Bandwidth Allocation

After batch inference, user n must satisfy the SLO, with downlink latency $T_n^{\text{down},t}$ constrained by the residual margin $T_n^{\text{re},t} = \varsigma_n^t - T_n^{\text{wait},t} - T_n^{\text{up},t} - T_n^{\text{stay},t} - T_j^{\text{inf},t}$. The downlink bandwidth allocation problem at node j seeks to maximize the number of requests meeting the SLO, aligning with system goodput. Let $\mathcal{N}_j^{\text{good},t} = \{n | n \in \mathcal{N}_j^t, T_n^{\text{down},t} \le T_n^{\text{re},t}\}$ be the set of requests contributing to goodput. The subproblem of downlink bandwidth allocation is then formulated as

$$\mathcal{P}3: \max_{\{\beta_{nj}^{\text{down},t}\}_{n\in\mathcal{N}_j^t}} \quad |\mathcal{N}_j^{\text{good},t}| \qquad \text{s.t.} \quad \sum_{n\in\mathcal{N}_j^t} \beta_{nj}^{\text{down},t} \le 1, \quad \forall j,t. \tag{4}$$

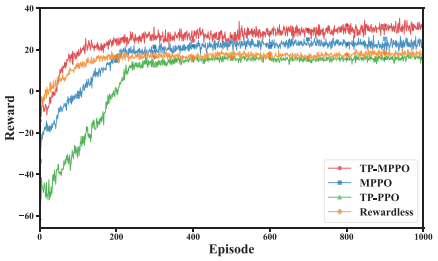


Fig. 3. Reward as the number of episode grows.

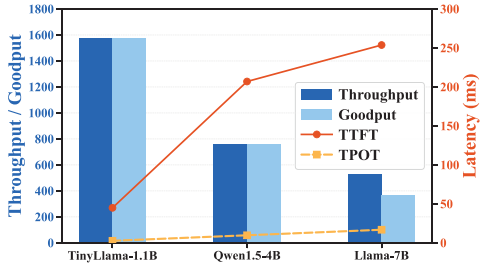


Fig. 4. Comparisons of throughput, goodput, TTFT, and TPOT.

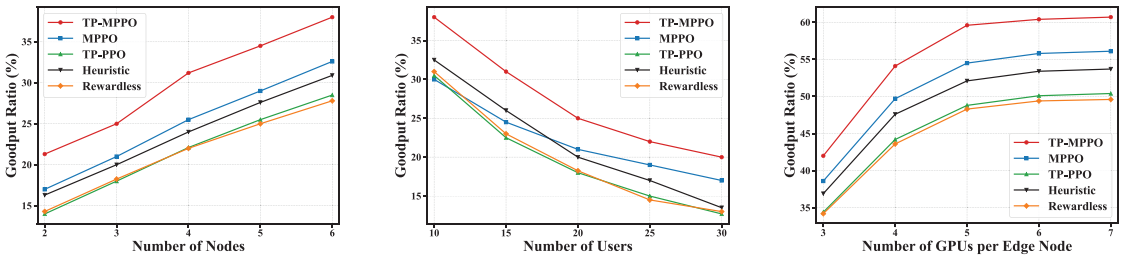
Based on the expressions of $R_{nj}^{\text{down},t}$ and $T_n^{\text{down},t}$, we calculate the minimum required bandwidth fraction (subject to $T_n^{\text{down},t} \le T_n^{\text{re},t}$) for user n by setting $T_n^{\text{down},t} = T_n^{\text{re},t}$:

$$\beta_{nj,\min}^{\text{down},t} \triangleq \frac{D_n^{\text{out},t}}{T_n^{\text{re},t} B_j^{\text{down}} \log_2(1 + \frac{p_{nj}^{\text{down},t} h_{nj}^{\text{down},t}}{\psi'})}. \tag{5}$$

By introducing $\beta_{nj,\min}^{\text{down},t}$, we can reformulate problem $\mathcal{P}3$ as finding the largest subset $\mathcal{S}_j^t \subseteq \mathcal{N}_j^t$ that satisfies the constraint $\sum_{n\in\mathcal{S}_j^t} \beta_{nj,\min}^{\text{down},t} \le 1$. This problem has a special equal-value structure, where all selected requests contribute equally (i.e., one unit) to the goodput. Hence, Algorithm 1 achieves the optimal solution by prioritizing requests with the smallest bandwidth requirements, attaining zero optimality gap. The complexity of Algorithm 1 is dominated by the sorting operation, which is $\mathcal{O}(N \log N)$.

V. Numerical Results

Consider a system where 3 edge nodes and 15 users are randomly deployed within a 300 m radius circular area. The input sequence length of inference request is randomly generated from $\{128, 256, 512\}$ tokens, with corresponding SLO targets of $\{3, 5, 9\}$ s. The maximum output sequence length $s_n^{\text{out},t}$ is randomly generated from $\{128, 256, 512, 1024\}$ tokens, and the realized output length is set as $o_n^t = \lambda_n^t s_n^{\text{out},t}$, where $\lambda_n^t \in \{0.5, 0.75, 1\}$ [1], [8]. All these task parameters are independently sampled uniformly from their respective discrete sets. For communication modeling, we assume 32 bits per token, consistent with the INT32 representation commonly used for token IDs. To ensure the simulated edge nodes reflect a realistic deployment, the memory capacity and computational capacity are set according to the specifications of an NVIDIA L2 PCIe GPU with 24 GB memory. We adopt the native Llama-7B hyperparameters: $h = 4096$ and $L = 32$. We also set $f_c = 3.5$ GHz and $d_e = 3$, representing sub-6 GHz fifth-generation (5G) edge access and urban/suburban non-line-of-sight (NLOS) propagation, respectively, together



(a) (b) (c)

Fig. 5. Goodput ratio with different numbers of nodes and users, and different edge-node GPU provisioning levels (Llama-30B).

with $\tau = 1$ s, $\psi = 10^{-14}$ W, $\psi' = 2 \times 10^{-14}$ W, $\epsilon = 0.2$, $\nu_{\text{good}} = 1$, and $\nu_{\text{oom}} = 0.5$.

We compare TP-MPPO with four benchmarks: 1) MPPO, which learns all variables solely via MPPO. 2) TP-PPO [12], which adopts PPO without action masking for offloading decisions. 3) Rewardless [6], which adopts classical active inference for offloading without reward guidance. 4) Heuristic [13], which employs a weighted scoring mechanism based on channel conditions and memory load for offloading decisions.

Fig. 3 shows the rewards of the considered algorithms. TP-MPPO yields the highest reward and converges within 200 episodes. The reward of TP-MPPO is 33.3%, 87.5% and 71.4% higher than those of MPPO, TP-PPO and Rewardless, respectively. Action masking in TP-MPPO and MPPO accelerates convergence by eliminating invalid action explorations. TP-PPO struggles with early-stage cold starts, while Rewardless converges to a low reward, indicating its free-energy-based guidance is insufficient to avoid OOM states.

Fig. 4 shows the impact of model size on the throughput–goodput gap and latency. Lighter models achieve lower latency and higher throughput/goodput with a smaller gap, as their lower resource demands reduce SLO violations and OOM failures. In contrast, Llama-7B incurs a larger gap due to its higher memory and computational demands, highlighting the importance of goodput-oriented resource allocation and the trade-off between model capacity and service reliability.

Figs. 5(a) and (b) demonstrate how number of edge nodes and users affect the ratio of goodput to total requests. In Fig. 5(a) (with 20 users), TP-MPPO surpasses all benchmarks. With 6 nodes, the goodput ratio of TP-MPPO exceeds MPPO, TP-PPO, Heuristic and Rewardless by 5.4%, 9.5%, 7.1% and 10.2%, respectively. In Fig. 5(b) (with 3 nodes), as the user count scales to 30, the goodput ratio of TP-MPPO exceeds MPPO, TP-PPO, Heuristic and Rewardless by 3%, 7.3%, 6.5% and 7%, respectively. The relatively low goodput ratios in Figs. 5(a) and (b) is caused by the constrained edge deployment scenario, where limited GPU resources and strict SLOs reduce the number of successfully served requests. Fig. 5(c) further evaluates the goodput ratio with the Llama-30B model under varying GPU provisioning levels. As GPU resources increase, TP-MPPO consistently outperforms the baselines, improving the goodput ratio from 42.0% to 60.7% by reducing OOM failures and inference latency.

VI. Conclusions

This paper presented the TP-MPPO framework for edge LLM inference services, where the task offloading decisions were learned by MPPO, uplink bandwidth allocation was given in closed form, and downlink bandwidth allocation was decided by the greedy algorithm. Simulation results have demonstrated that TP-MPPO improves the reward by 33.3%–87.5% and outperforms the benchmarks significantly in goodput performance across various network configurations.

References

[1] X. Zhang, J. Nie, Y. Huang, G. Xie, Z. Xiong, J. Liu, D. Niyato, and X. Shen, "Beyond the cloud: Edge inference for generative large language models in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 24, no. 1, pp. 643–658, 2025.

[2] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu, "KIVI: A tuning-free asymmetric 2bit quantization for KV cache," in *International Conference on Machine Learning (ICML)*, 2024.

[3] D. Xu, W. Yin, H. Zhang, X. Jin, Y. Zhang, S. Wei, M. Xu, and X. Liu, "EdgeLLM: Fast on-device LLM inference with speculative decoding," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 3256–3273, 2025.

[4] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A distributed serving system for transformer-based generative models," in *Proc. 16th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2022, pp. 521–538.

[5] Y. He, J. Fang, F. R. Yu, and V. C. Leung, "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach," *IEEE Trans. Mob. Comput.*, vol. 23, no. 12, pp. 11 253–11 264, 2024.

[6] X. He, Y. Jiang, X. Xu, H. Cui, Y. Liu, M. Chen, Y. Hong, and J. Zhang, "Large language model offloading using active inference in 6G symbiotic IoT," *IEEE Internet Things J.*, pp. 1–1, 2025.

[7] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in *Proc. 18th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2024, pp. 193–210.

[8] H. Zhou, C. Hu, D. Yuan, Y. Yuan, D. Wu, X. Liu, Z. Han, and J. Zhang, "Generative AI as a service in 6G edge-cloud: Generation task offloading by in-context learning," *IEEE Wireless Commun. Lett.*, vol. 14, no. 3, pp. 711–715, 2025.

[9] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[10] H. Han, Y. Xu, Z. Feng, X. Wang, Y. Xu, W. Li, and F. Zhou, "Efficient dynamic spectrum anti-jamming access with large action space: An action space decomposition-based approach," *IEEE Wireless Commun. Lett.*, vol. 13, no. 10, pp. 2667–2671, 2024.

[11] X. Ding, Y. Zhang, B. Chen, D. Ying, T. Zhang, J. Chen, L. Zhang, A. Cerpa, and W. Du, "Scalable and efficient reinforcement learning for virtual machine rescheduling in cloud data centers," *IEEE Trans. Parallel Distrib. Syst.*, pp. 1–18, 2026.

[12] X. Ye, Y. Sun, D. Wen, G. Pan, and S. Zhang, "End-to-end delay minimization based on joint optimization of DNN partitioning and resource allocation for cooperative edge inference," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, 2023, pp. 1–7.

[13] Z. Liu, Q. Lan, and K. Huang, "Resource allocation for multiuser edge inference with batching and early exiting," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1186–1200, 2023.