Efficient Mixture-of-Experts Model Inference at the Edge via Adaptive Expert Merging

Ruirui Zhang, Yifei Zou, *Member, IEEE*, Peng Li, *Senior Member, IEEE*, Fahao Chen, Yupeng Li, Xiuzhen Cheng, *Fellow, IEEE*, Falko Dressler, *Fellow, IEEE*, and Dongxiao Yu, *Senior Member, IEEE*

***Abstract*—This paper studies the Mixture-of-Experts (MoE) model inference problem at the edge via the adaptive expert merging technique. To fill the gap between the large size of MoE models and the limited hardware resources at the edge, existing works primarily focus on static merging policies and overlook the dynamic and heterogeneous nature of edge environments. This paper addresses these gaps by proposing a flexible framework for adaptive expert merging and online inference task offloading on edge servers. Our contributions include (1) a novel approach allowing each edge server to adaptively merge experts based on its resources and task preferences, (2) an online inference task offloading algorithm with a bounded competitive ratio for pre-determined merging policies, and (3) an online algorithm for joint optimization of task offloading and expert merging, which ensures timely model updates and assignment of tasks to specialized servers. Experimental results on five datasets demonstrate that our approach reduces the overall weighted cost by at least 16% compared to baseline methods.**

***Index Terms*—Mixture-of-Experts, Mobile Edge Computing, Edge Network, Task Offloading, Expert Merging.**

I. Introduction

Implementing a Mixture-of-Experts (MoE) based Large Language Model (LLM) with trillions of parameters on the edge side can provide users with specialized, fast, and private inference services [1], [2]. However, the gap between huge MoE size and limited hardware resources at edge presents a critical challenge. Recently, expert merging [3], [4] has emerged as an effective approach to reduce the size of MoEs while maintaining their accuracy. This technique is based on the observation that not all experts contribute equally to the model's performance [5]. By merging less important experts into more significant ones, the number of experts is reduced, thereby decreasing the overall size of the LLM [6]. Meanwhile, this process preserves the knowledge embedded within the experts, ensuring that the model's capabilities remain mostly intact, with only slight degradation in performance [7].

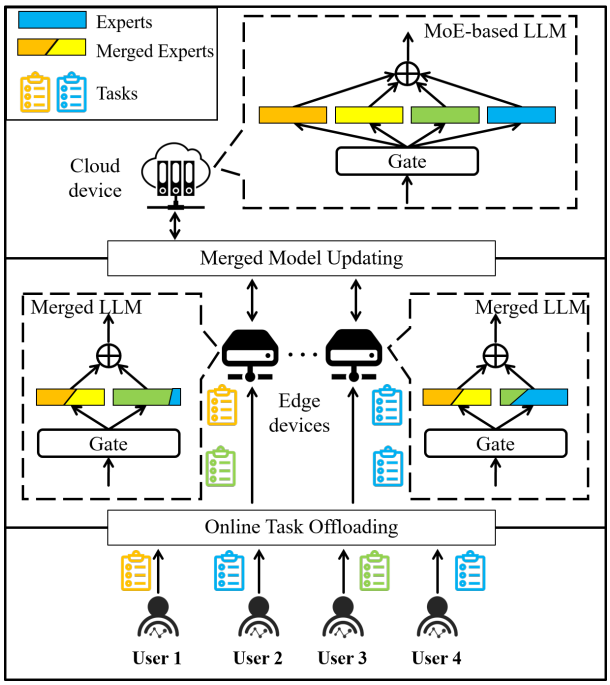


Fig. 1. Online task offloading and merged model updating for MoE-based LLM inference on resource-constrained edge

In this paper, we study the deployment of MoE inference services at the edge using expert merging. As illustrated in Figure 1, we consider multiple edge servers that cannot accommodate a complete MoE model due to limited hardware resources. Instead, each edge server is equipped with a smaller MoE model with merged experts, providing inference services to nearby users. Although expert merging has great potential, existing works primarily originate from the AI community and often rely on strong assumptions about data distribution and hardware resources [5]. These works lack a comprehensive study of the following deployment problems at the edge. **First**, existing research typically considers a single, static expert merging policy (e.g. [3], [7]). However, inference tasks from edge users are often highly diverse and time-varying, which activate different experts and necessitate different expert merging policies. Consequently, each edge server should adopt distinct expert merging policies and dynamically adjust them based on changing task preferences. **Second**, the problem of offloading user tasks to edge servers is closely coupled with the design of expert merging policies. Users generally

Ruirui Zhang, Yifei Zou, Xiuzhen Cheng are with the Department of Computer Science, Shandong University, Qingdao, Shandong, China, 266237 (email: sherryz@mail.sdu.edu.cn; {yfzou, xzcheng}@sdu.edu.cn).

Dongxiao Yu is with the School of Cryptography Science and Engineering, Shandong University, Jinan, Shandong, China, 250101 (email: dxyu@sdu.edu.cn).

Yifei Zou, Xiuzhen Cheng, and Dongxiao Yu are also with the Shandong Provincial Key Laboratory of Computing-Network Integration, Qingdao, Shandong, China, 266237.

Peng Li is with the Department of Computer Science, Xi'an Jiaotong University, Xi'an, China (email: pengli@xjtu.edu.cn).

Fahao Chen is with the School of Artificial Intelligence, Shandong University, Jinan, Shandong, China, 250100 (email: chenfh@ieee.org).

Yupeng Li is with the Department of Interactive Media, Hong Kong Baptist University, Kowloon, Hong Kong (email: ypengl@hkbu.edu.hk).

Falko Dressler is with the School of Electrical Engineering and Computer Science, TU Berlin, Berlin, 10587, Germany (email: dressler@ccs-labs.org).

(Corresponding author: Yifei Zou.)

have connections to multiple edge servers and can choose the one that offers the minimum latency or the highest inference accuracy. Simultaneously, edge servers can adjust their expert merging policies based on the characteristics of incoming tasks. Therefore, **joint optimization is required to enhance overall performance**.

To address these problems, we propose applying adaptive expert merging policies for edge servers with different task distributions and jointly optimizing expert merging and inference task offloading. The first challenge lies in updating the expert merging policies on each edge server. As discussed in Section II, the service quality provided by different merged LLMs can vary significantly for the same tasks. It is essential for each server to update its expert merging policy in a timely manner based on the tasks it receives. However, updating these policies incurs additional switching overhead due to model reloading. Thus, it is important to carefully balance the benefits of policy updates against the resource consumption they entail. The second challenge involves making online decisions without sufficient information about inference tasks and the LLM execution process. Given the generative nature of LLMs, the exact output length and thus the inference time for any given task is uncertain [8]. Therefore, when a task is offloaded to a server, its start and completion times cannot be precisely determined, both of which are crucial for designing an effective offloading strategy.

We design algorithms to address the joint optimization of expert merging policies and inference task offloading. Given the high complexity of this problem, we decompose it into two sub-problems and solve them sequentially. In the first sub-problem, edge servers have distinct but pre-determined expert merging policies. This approach simplifies our algorithm design and is also practical in scenarios where the server daily updates its merged LLM from the cloud. The expert merging policy within a day is pre-determined for each server. To solve this sub-problem, we develop an online inference task offloading algorithm with a bounded competitive ratio. In the second sub-problem, we relax the assumption of pre-determined expert merging policies and allow for their timely updates. We design an online algorithm to jointly optimize task offloading and expert merging. To prevent over-frequent model updates, which could consume excessive computing and communication resources in the edge network, servers update their merged models only when the activation frequency of their experts exceeds a certain threshold.

The contributions of our work are listed as follows.

1) This paper studies the deployment of MoE inference services at the edge using adaptive expert merging. Different from the existing merging technique regardless of the heterogeneity of edge devices and time-varying feature of tasks, our framework allows each server to adaptively merge its experts according to its own resource and task preferences, and offload the tasks to the server with the specialized merged LLM to have a fast and high-accuracy inference. To the best of our knowledge, this is the first paper jointly optimizing the expert merging policies and inference task offloading when LLM inference services are implemented at the edge.
2) For arbitrary pre-determined expert merging policies, an online inference task offloading algorithm is proposed, to minimize the latency of the inference and maximize the accuracy. With theoretical proofs, its competitive ratio to the optimal solution is $\frac{2\mu}{\mu+1}$, in which μ is the hyperparameter for the task inference time range.
3) An online algorithm for joint optimization of task offloading and expert merging is proposed. In our algorithm, the servers adaptively merge their experts according to their assigned tasks, which guarantees that there are always some specialized experts at the edge for task offloading even when the task distribution changes. Then, the task offloading strategy makes sure that the tasks can be assigned to the server with the specialized experts. Comparisons with the baselines from [9]–[14] on five datasets [15]–[19] show that our proposed framework reduces the overall weighted cost by at least 16.0%.

Roadmap. In Sec. 2, we present the background and motivation of our work. Sec. 3 describes the network model and the problem formulation. In Sec. 4, we propose the online task offloading algorithm for any given merged model deployed on the servers and present the theoretical proofs. Furthermore, in Sec. 5, we propose a joint task offloading and merged model updating algorithm, which enables the servers to update their merged models in the task offloading process. The numerical results are reported in Sec. 6. Sec. 7 shows the related work. Finally, Sec. 8 concludes the overall work.

II. Background and Motivation

In this part, we introduce the background knowledge of MoE architecture and expert merging technique, the motivations for adaptive expert merging and inference task offloading, respectively.

1) Background Knowledge of Mixture of Experts (MoE). The MoE [1] model, based on the Transformer architecture, primarily consists of two key components: sparse MoE layers and gating networks or routers. Sparse MoE layers replace the traditional feed-forward network (FFN) layers in Transformer models and are composed of several "experts", each being an independent neural network. Typically, these experts are FFNs, though they can be more complex structures or even other MoE layers, forming a hierarchical MoE structure. The gating network or router determines which tokens are routed to which experts, with some tokens potentially being sent to multiple experts. This routing mechanism is crucial as it is governed by learned parameters that are co-trained with the rest of the network. In summary, an MoE model substitutes each traditional FFN layer in a Transformer model with an MoE layer, consisting of a gating network and multiple experts. Its sparse activation strategy offers enhanced scalability, lower computational overhead, and improved accuracy.

2) Background Knowledge of Expert Merging Technique. Expert merge [3]–[7] is a technique that reduces the overall parameter count of an MoE model by grouping and aggregating individual experts. This approach effectively compresses

the model, making it more resource-efficient while maintaining high accuracy across various datasets. Specifically, after evaluating the importance of each expert (e.g. its activation frequency in inference), the expert merging technique combines multiple experts into a single expert using a weighted average strategy. By doing this, the size of MoE-based LLM gets reduced, while the knowledge behind the experts is preserved according to their importance. With the expert merging technique, the works in [3]–[7] improve model performance, computational efficiency, and deployment feasibility.

3) Motivation for Adaptive Expert Merging. In MoE-based LLM inference experiments, we observe the fact that the task distribution impacts the activation frequencies of experts. The LLM model used in our experiments includes 12 MoE layers with 32 experts per layer. Four distinct tasks (WinoGrande [19], WikiQA [18], SQuAD [17], and COPA [15]) are tested and their expert activation frequencies are presented in Fig. 2. The heatmaps in Fig. 2 show that each kind of task activates different experts in the MoE model. Specifically, the task WinoGrande is for conference resolution, the task WikiQA is typified by answer-selection prompts, and the two tasks, COPA and SQuAD, are characterized by answer-generation prompts. Notably, even for similar tasks like COPA and SQuAD, the distribution of their expert activation frequency differs significantly. Thus, when experts are merged for a server, not only its GPU resources but also its tasks assigned should be considered, ensuring that the merged LLM fits the server's GPU capacity and experts with high activation frequencies on its tasks should have more weights in the merging process, i.e. the expert merging process should be adaptive to the resource of edge devices and tasks of users.

4) Motivation for Online Task offloading. In our experiments, we observe another fact that LLMs merged from different strategies excel at different tasks. Using the approach from [7], we merged four models based on four different datasets, resulting in four merged LLMs: MM 1, MM 2, MM 3, and MM 4. While these models perform well on their respective datasets, they show poor performance on others. For instance, MM 2 was merged based on the WikiQA dataset, but its accuracy on the SQuAD dataset drops dramatically to 0.028, which is a significant decrease compared to MM 3, which was merged using the SQuAD dataset and achieves an accuracy of 0.654. In contrast, MM 1 and MM 4 achieve their highest accuracy on Winogrande (0.696) and COPA (0.670), respectively. Table I presents these results in more detail, showing the performance of each merged model across various downstream tasks. Thus, for multiple servers with adaptive merged LLM, their workload and the specialty of their merged LLM should be considered together in task offloading.

III. System Model and Problem Definition

We consider a classical Cloud-Edge-End network architecture, in which the LLM is deployed on the edge to provide inference services. The network timeline is divided into discrete time slots, denoted as $t \in \{0, 1, 2, ..., \}$. The duration of each time slot varies depending on the task complexity, ranging from several milliseconds to a few minutes.

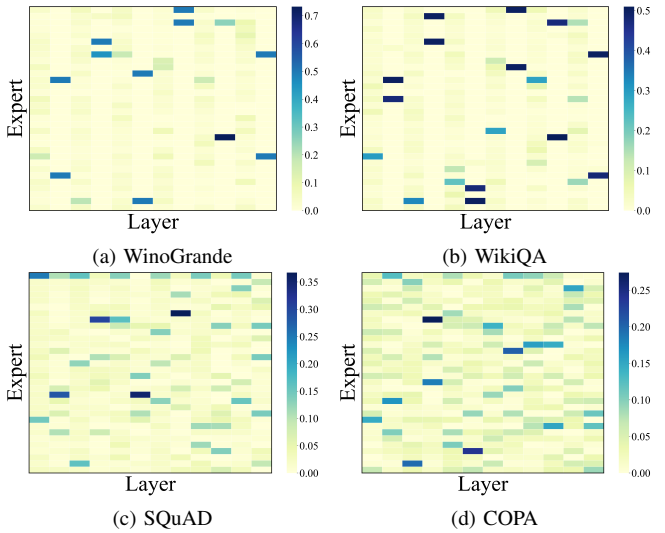


Fig. 2. Distribution of expert activation frequencies. In the heatmap, from left to right is the 12 layers of MoE and from up to down is the 32 experts in each layer.

TABLE I
ACCURACY COMPARISON OF DIFFERENT TASK-APPLIED MERGED MOE MODELS ACROSS VARIOUS DOWNSTREAM TASKS. THE COLUMN HEADERS INDICATE THE TASK USED FOR MERGING THE MODELS (WINOGRANDE, WIKIQA, SQUAD, COPA). MERGED MODEL (MM) 1, 2, 3, AND 4 REPRESENT THE MERGED MODELS OBTAINED AFTER MERGING TASKS FROM WINOGRANDE, WIKIQA, SQUAD, AND COPA, RESPECTIVELY. EACH CELL SHOWS THE ACCURACY OF THE MODEL ON THE CORRESPONDING DATASET.

Dataset	Merged MoE LLM			
	MM 1	MM 2	MM 3	MM 4
Winogrande	0.696	0.436	0.554	0.504
WikiQA	0.487	0.949	0.483	0.544
SQuAD	0.555	0.028	0.654	0.246
COPA	0.288	0.395	0.501	0.670

On the edge side, there are S servers, denoted by the set $\mathcal{S} = \{1, 2, \ldots, S\}$. We address a challenging scenario where the LLM $\mathcal{M}$ on the cloud is too large to be deployed on servers with limited GPU resources. Therefore, a merging technique is employed to combine multiple experts into a single one, thereby reducing the LLM size. Let $\mathcal{M}_s$ be the merged LLM on server s, which can provide services with a quality $q_s(r)$ for the task r. We define q_{min} and q_{max} as the lower bound and upper bound for the service quality provided by all servers.

On the end side, numerous inference tasks r_i are released by the users over time, denoted as $\mathcal{R} = \{r_1, r_2, \ldots\}$. These tasks originate from different applications or users and exhibit varying service quality even when assigned to the same server. Let a_i and c_i be the time slot when task r_i is released by the user and returned by a server s. From a_i to c_i, the task r_i is assigned to server s, waits for some time slots until the server starts processing it, undergoes processing, and is finally returned by the server. Thus, we have $c_i = a_i + t^{tran}_{i,s} + t^{wait}_{i,s} + t^{inf}_{i,s} + t^{tran}_{i,s}$. The transmitting time is defined as $t^{tran}_{i,s}$ of task r_i from its user to server s or from server s to its user. The value of $t^{tran}_{i,s}$ can be known when making the offloading decision because the network bandwidth and latency can be detected.

TABLE II
SYMBOL TABLE

Symbol	Description
t	Discrete time slot
S	Total number of edge servers
$\mathcal{S}$	Set of edge servers, $\mathcal{S} = \{1, 2, \ldots, S\}$
$\mathcal{M}$	Large Language Model (LLM) on the cloud
$\mathcal{M}_s$	Merged LLM on server s
$q_s(r)$	Service quality provided by server s for task r
q_{min}, q_{max}	Lower/Upper bound of Service quality
$\mathcal{R}$	Set of tasks, $\mathcal{R} = \{r_1, r_2, \ldots\}$
r_i	Inference task r_i
a_i, c_i	Time slot when task r_i is released/returned
$t_{i,s}^{tran}$	Transmission time of task r_i to/from server s
$t_{i,s}^{wait}$	Waiting time of task r_i on server s
$t_{i,s}^{inf}$	Inference time of task r_i on server s
$\hat{t}_{i,s}^{inf}$	Predicted inference time of task r_i on server s
t_{min}^{inf}	Minimum inference time
μ	Hyperparameter for inference time range

Furthermore, in edge networks, the transmission time for input and output data is significantly shorter than the inference time of the service. Therefore, in the subsequent algorithms and proofs, we will omit this aspect from our consideration. The waiting time $t_{i,s}^{wait}$ of task r_i on server s occurs while the server processes earlier arriving tasks. The inference time $t_{i,s}^{inf}$ of task r_i on server s is unpredictable due to the generative nature of the models, which makes the exact output length and inference time difficult to determine. For simplicity, we assume that the inference time $t_{i,s}^{inf}$ lies within a range defined by $t_{min}^{inf} \leq t_{i,s}^{inf} \leq \mu t_{min}^{inf}$, where μ is a hyperparameter. Since the inference time $t_{i,s}^{inf}$ is unpredictable, the waiting time $t_{i,s}^{wait}$ is also unpredictable, as it depends on the server's processing of earlier tasks.

Expert Merging Technique. The expert merging technique is used to compress the original LLM on the cloud so that the merged LLM can be deployed on the edge. Firstly, we analyze the activation frequency of each expert in the MoE for the corresponding input data, identifying the dominant experts based on these frequencies. Then, we group the experts around each dominant expert for each layer and merge each group into a single expert. In [7], the merging technique does not target a specific compression size. That is, for the same task, the size of the MoE model merged using the method in [7] remains fixed, and the model sizes resulting from merging different tasks are inherently inconsistent. This limitation makes it unsuitable for scenarios involving multiple servers with heterogeneous storage capacities. In contrast, our work evaluates the available GPU storage on the edge server and sets this as the target size for our merge process.

Problem Formulation. The primary objective of our paper is to provide users with high-quality and rapid LLM inference services on the edge. To achieve this goal, we must design an online task offloading strategy that assigns tasks to the most appropriate server, considering the quality of service and the associated time cost. Additionally, as demonstrated in Sec. 2, various tasks exhibit different activation frequencies among the experts. Consequently, the servers need to update their merged LLM from the cloud based on the assigned tasks.

IV. ONTO - ONLINE TASK OFFLOADING ALGORITHM

In this section, we first analyze a scenario where the merged MoE deployed on each server is predetermined and fixed. The task offloading problem is then formulated within this context. We propose an online algorithm to address the task offloading problem, and a theoretical analysis of the proposed algorithm is provided.

A. Problem 1 (P1): Online Task Offloading

Assuming that each server has already obtained its merged LLM from the cloud, the online task offloading problem focuses on assigning the tasks to the most appropriate server to optimize its service quality and total time cost. For each task r_i arriving at time slot a_i, it is immediately assigned to one of the edge servers. We assume that once a task r_i is assigned to server s, it cannot be migrated to another server. This is because a Key-Value (KV) cache is often used for optimizing generation in autoregressive models, where the model predicts text token by token. Migrating a task to another server during runtime involves transferring the KV cache. However, due to the generative network's nature, different server models may render the KV cache unusable, necessitating a restart of inference from scratch. Consequently, our system prevents preemption or migration to maintain efficiency.

Let $x_{i,s} \in \{0, 1\}$ denote the offloading decision for task r_i on server s at time slot a_i. Specifically, $x_{i,s} = 1$ indicates that task r_i is assigned to server s at time slot a_i. The allocation decision must satisfy the following constraint.

$$\sum_{s \in \mathcal{S}} x_{i,s} = 1, \quad \forall\ r_i \in R$$

which means each task will be assigned to only one server when it arrives on the edge side.

For any set $\mathcal{R}$ with finite tasks online arriving, the objective of our offloading strategy is to complete all tasks as soon as possible and with high quality. Thus, the task offloading optimization problem is defined as follows:

$$\min_{x_{i,s}} \quad c_{max} - \sum_{r_i \in \mathcal{R}} \alpha_i \sum_{s \in \mathcal{S}} x_{i,s} q_s(r_i) \tag{1}$$

$$s.t. \quad \sum_{s \in \mathcal{S}} x_{i,s} = 1, \quad \forall\ r_i \in R \tag{2}$$

$$x_{i,s} \in \{0, 1\}, \forall r_i \in \mathcal{R}, \forall s \in \mathcal{S} \tag{3}$$

In the above equation, $c_{max} = \max_{r_i \in R} c_i$ is the earliest time slot when all the tasks are completed. $\sum_{s \in S} x_{i,s} q_s(r_i)$ is the service quality for task r_i. The hyperparameter $\alpha_i > 0$ is used to balance the service quality for task r_i and its impact on the time cost. In other words, a larger value $\alpha_i > 0$ implies a higher preference for the service quality for task r_i. The term $\sum_{r_i \in \mathcal{R}} \alpha_i \sum_{s \in \mathcal{S}} x_{i,s} q_s(r_i)$ is the sum of the weighted service quality for all the tasks in set R.

Then we prove the **NP-Hardness** of problem P1.

Theorem 1. *The allocation problem **P1** is NP-hard.*

Proof. Let's consider a special scenario where all servers' merged MoE models provide the same quality for all tasks. This means the optimization objective for problem **P1** becomes minimizing the overall system completion time, i.e., the makespan, when inference times are unknown. For this special case, we can treat it as a robust scheduling problem with multiple servers and uncertain processing times. This reduced problem has been proven to be NP-hard in Theorem 1 of the paper [20]. Therefore, problem **P1** is NP-hard. □

B. Online Task Offloading Algorithm

Solving the online task offloading problem, even in an offline manner, is challenging. An optimal solution requires prior knowledge of the inference time for each task. However, since inference time is influenced by the output length of the generated response, accurately predicting this length is difficult given the characteristics of generative models. Let's assume that for each unknown $t_{i,s}^{inf}$, we have a predicted value $\hat{t}_{i,s}^{inf}$, where $\hat{t}_{i,s}^{inf}$ and the true value share the same range: $t_{min}^{inf} \leq \hat{t}_{i,s}^{inf} \leq \mu t_{min}^{inf}$. Notably, this assumption is not unfounded. Since the processing time of generative LLMs is highly dependent on output length, we estimate the inference time by first predicting the generated token length. This can be achieved using lightweight methods, such as historical moving averages over recent queries or a simple Length Predictor head. Following [8], the predicted token length is then mapped to $\hat{t}_{i,s}^{inf}$ through linear scaling calibrated from our observed dataset profiles. Similarly, the predicted service quality $\hat{q}_s(r_i)$ for task r_i on server s can be estimated based on historical performance data or other predictive models. It is crucial to note that the predicted service quality is **task-specific**, just as we demonstrate in Table I; that is, the quality for different tasks r_i assigned to the same server s can vary significantly depending on factors such as task complexity, resource requirements, and other server-specific attributes. Therefore, even if tasks are processed on the same server, the service quality $\hat{q}_s(r_i)$ is likely to differ based on the unique nature of each task r_i. This **task-specific quality** aspect plays a key role in accurately estimating the overall performance and cost for task offloading.

Our online task offloading strategy assigns tasks based on predicted values, following the general process outlined below. For detailed steps, please refer to the pseudocode in Algorithm 1.

For each arriving task, our strategy needs to assign it to a server immediately. Therefore, we focus on the decision-making for task r_i at its arrival time a_i. We calculate its waiting time on each server s using the predicted times of other tasks assigned but still not completed on the server s. Specifically, for each $t_{i,s}^{inf}$, we substitute it with the predicted value $\hat{t}_{i,s}^{inf}$. Thus, the predicted time cost of assigning task r_i on server s is $\hat{c}_i - a_i$, and the predicted weighted service quality is $\alpha_i \hat{q}_s(r_i)$. Then, the predicted cost of task r_i on server s is defined as $\hat{c}_i - a_i - \alpha_i \hat{q}_s(r_i)$. Consequently, we greedily select the server s with the minimum predicted cost for task r_i, setting $x_{i,s} = 1$.

Algorithm 1 ONTO Algorithm

Require: Set of servers $\mathcal{S}$, set of tasks $\mathcal{R}$, arrival time a_i and weights α_i for each task r_i, predicted inference time $\hat{t}_{i,s}^{inf}$ for task r_i on server s

for Task r_i arrived at the edge side at time slot a_i **do**
  for each server $s \in \mathcal{S}$ **do**
    Calculate the waiting time for task r_i on server s using the predicted inference times of other tasks assigned to s
    Predict the quality $\hat{q}_s(r_i)$ of task r_i on server s
    Replace $t_{i,s}^{inf}$ with $\hat{t}_{i,s}^{inf}$
    Compute the predicted cost $\hat{c}_i - a_i - \alpha_i \hat{q}_s(r_i)$ for task r_i on server s
  end for
  Select the server s' with the minimum predicted cost for r_i
  Allocate task r_i to server s' by setting $x_{i,s'}(a_i) = 1$
end for

C. Analysis of ONTO

To facilitate the performance analysis of ONTO, we first establish some assumptions about the problem scenario.

Assumption 1. *The inference time $t_{i,s}^{inf}$ for task r_i on server s is not only bounded by $t_{min}^{inf} \leq t_{i,s}^{inf} \leq \mu t_{min}^{inf}$ but also has a mean value of $\frac{\mu+1}{2} t_{min}^{inf}$.*

This assumption is consistent with our empirical observations on real LLM inference workloads. As discussed in Appendix C-A, inference tasks from the same downstream application usually share similar prompt/output patterns, and their measured inference times tend to concentrate around task-dependent central values within finite bounded ranges. Therefore, the bounded-range and representative-mean characterization in Assumption 1 provides a practical approximation for task groups with similar generation patterns.

Based on this characterization, we can construct both the best-case completion-time lower bound for the offline optimal solution and the worst-case completion-time upper bound for the online algorithm. We present a brief proof in the following lemma to support this point.

Lemma 1. *When the inference time for all tasks $\mathcal{R}$ have an average value of $\mu' t_{min}^{inf}$, the best-case time cost for the offline algorithm is $\frac{R\mu'}{S} t_{min}^{inf}$. The worst-case scenario for Algorithm 1 is at most $\frac{R\mu}{S} t_{min}^{inf}$.*

The proof of Lemma 1 is given in the Appendix.

Definition 1. *The service quality $q_s(r_i)$ of each task r_i provided by each server s has a common upper and lower bound, $q_{min} \leq q_s(r_i) \leq q_{max}$. Furthermore, the bounds q_{min} and q_{max} satisfy the following relationship:*

$$\frac{q_{max}}{q_{min}} = \beta \tag{4}$$

Assumption 2. *For the ratio μ of the upper and lower bounds of task inference time, we make the following assumptions:*

$$S \leq \mu \leq 2\beta S - 1$$

This assumption is easily achievable in practice. We conducted a simple inference time test on the WikiQA dataset [18] using the merged MoE model at the edge network with several servers and end devices. The results showed the maximum inference time was 4 seconds and the minimum was 0.2 seconds. This implies $\mu = 20$, meaning we need to select $S = 20$ to satisfy the above assumption.

With the assumptions above, we then give the competitive ratio of the online algorithm. The proof of Theorem 2 is given in the Appendix.

Theorem 2. *Given the task allocation problem proposed in Problem **P1**, the online allocation algorithm, shown as* `Algorithm` *1, is* $(2-\frac{2}{\mu+1})$*-competitive under Assumption 1 and 2.*

Remark 1. *Impact of the prediction errors regarding* $\hat{t}_{i,s}^{inf}$*. Our theoretical analysis inherently accommodates such deviations as long as both the predicted value* $\hat{t}_{i,s}^{inf}$ *and the true inference time fall within the established bounds* $[t_{min}^{inf}, \mu t_{min}^{inf}]$*. Within this range, the theoretical competitive ratio mathematically absorbs the prediction gap, which is bounded by the parameters* μ *and* t_{min}^{inf}*. However, if predictions deviate significantly and frequently fall outside these bounds, the theoretical guarantees of the competitive ratio would loosen. To mitigate this in highly dynamic edge environments, the adaptive update mechanism in our JTOM framework acts as a system-level safeguard, shown as follows. By periodically recalibrating the merged MoE deployed on the edge server based on the actual, observed task execution frequencies, JTOM can dynamically correct continuous prediction biases, thereby minimizing the long-term impact of extreme prediction deviations.*

V. Joint Task Offloading and Expert Merging

A. Problem 2 (P2): Joint Optimization

Given that the merged MoE has already been deployed on the server, the previously discussed online task offloading algorithm has demonstrated promising theoretical performance. However, for dynamically distributed tasks continuously arriving in the edge network, further optimization is needed. As highlighted in the motivation, different expert merging strategies influence the quality of tasks. Thus, we incorporate the expert merging strategy into our problem to enhance dynamic task offloading performance, leading to the joint task offloading and expert merging problem, P2. In P2, we consider both the offloading of newly arrived tasks and expert merging decisions for optimization. According to the results in Table I, the service quality provided by different merged LLMs varies for the same task. Therefore, it is crucial for each server to update its merged MoE-based LLM from the cloud if the current model cannot deliver high-quality services. After updating, the service quality $q_s(r)$ for assigned tasks can improve, thereby minimizing the objective $c_{max} - \sum_{r_i \in \mathcal{R}} \alpha_i \sum_{s \in \mathcal{S}} x_{i,s} q_s(r_i)$. By providing an optimization algorithm for this problem, we ensure that all tasks receive high-quality and fast services from the edge.

B. JTOM Algorithm

We now propose a heuristic algorithm, JTOM (Joint Task Offloading and Expert Merging), to address the above joint task offloading and expert merging problem. The details are shown in Algorithm 2. The task offloading part of the aforementioned problem has been discussed in the previous section.

For the second part, the expert merging decision, an intuitive approach is for each server to adjust its merge strategy for received tasks at each time slot. However, this method incurs substantial communication costs because each server would need to request new merge schemes from the cloud and download updated merged MoE model parameters. This significant communication overhead necessitates strategic decisions regarding when each server should perform merge strategy updates.

Moreover, task distribution changes over time and the extent of this change is unknown. Therefore, using fixed time intervals for expert merging decisions is impractical, as it can lead to unnecessary communication costs during periods of stable task distribution and poor service performance during periods of rapid changes on task distribution. To address this, we dynamically adjust the timing of expert merging to better capture changes in task distribution.

To quantify the degree of task distribution change, we use the activation frequency of tasks on the MoE. Specifically, at each decision point, we compare the current expert activation frequencies in the MoE with the historical frequencies recorded during the previous decision. If the difference exceeds a certain threshold g, we proceed with a merge strategy update by requesting a new merged MoE from the cloud. A successful update indicates a significant change in task distribution, prompting a reduction in the interval δ before the next decision. Conversely, if the task distribution is stable, we appropriately extend the decision interval before the next update for lower communication overhead.

Now, we give the general procedure, which is shown as follows. The JTOM optimizes task offloading and expert merging decisions dynamically. First, in the initialization phase, it sets the decision interval δ, adjustment parameter m, and threshold g. At time slot $t = 0$, each server randomly selects a merging scheme for a downstream application based on its storage capacity to obtain the deployed merged MoE. Additionally, the activation frequency f', which refers to the activation frequency based on the expert merging strategy used for getting the merged MoE, is initialized at this stage. Then, for each time slot, the new tasks are assigned using Algorithm 1. Periodically, for each server, it analyzes the expert activation frequency f in current time slot and calculates the change $\|f - f'\|$ between the current task activation frequency f and the frequency of the expert merging strategy f'. If this change exceeds the threshold g, the server updates its expert merging strategy based on the new task activation frequency f, using the pre-determined expert merging algorithm like [3]–[7]; otherwise, it retains the current strategy. The decision interval δ is then adjusted: reduced by m if a new strategy is adopted, or increased by m if not. This adaptive mechanism ensures efficient task allocation and minimizes communication

overhead by adjusting the frequency of expert merging strategy updates based on task distribution dynamics.

Discussion on Theoretical Bounds. While the ONTO algorithm provides a strict competitive ratio of $\frac{2\mu}{\mu+1}$ for the task offloading component under a fixed merging policy, establishing a rigorous formal bound for the entire JTOM framework is more challenging because JTOM additionally involves adaptive merge-strategy switches. The number of such switches is determined by real-time task distribution shifts and therefore cannot be fixed in advance for arbitrary online workloads. To clarify the theoretical impact of these switches, we provide a conditional bound by explicitly incorporating the actual number of merge-strategy switches into the analysis.

Assumption 3. *(Constant merge-strategy switching delay). For a finite task sequence R with $N = |R|$ tasks and S edge servers, each merge-strategy switch triggered by JTOM introduces a constant worst-case delay*

$$T_{\text{merge}} = \frac{N}{S} t_{\min}^{\inf},$$

which is added to the completion-time component $c_{\max}$ of the objective. If JTOM triggers d merge-strategy switches during the whole task sequence, the total switching overhead is therefore dT_{merge}.

Proposition 1. Under Assumptions 1–3, if JTOM triggers d merge-strategy switches for a finite task sequence, then JTOM admits the following conditional competitive-ratio upper bound for the completion-time component of the objective:

$$\frac{2(\mu + d)}{\mu + 1}.$$

The proof of Proposition 1 and a detailed discussion are given in the Appendix.

Algorithm 2 JTOM Algorithm

Input: Initial interval δ, adjustment parameter m, threshold g
Output: Task Offloading and expert merging strategy decisions
for each time slot t **do**
  Use Algorithm 1 to allocate new arrival tasks
end for
for each server s in every δ slots **do**
  Analyze the expert activation frequency on s
  Calculate the change $\|f - f'\|$ between the current activation frequency f and the one f' of the merging strategy
  if $\|f - f'\| < g$ **then**
    Decide to maintain the current merge strategy
    $\delta \leftarrow \delta + m$
  else
    Reselect the merge strategy
    $\delta \leftarrow \min(\delta, \delta - m)$
  end if
end for

VI. Experiment

In this section, we evaluate the performance of the task offloading Algorithm 1 (ONTO) and the online joint task offloading and edge-side merge strategy Algorithm 2 (JTOM). The processing time, total quality, and weighted cost of both algorithms are evaluated under various server numbers and task generation rates. Additionally, necessary ablation experiments are conducted to validate our Expert Merging Update strategy in JTOM.

A. Experiment Setup

Hardware Environment. The simulations and model profiling are conducted on an edge server equipped with an AMD EPYC 7T83 64-Core CPU and an NVIDIA RTX 3090 GPU. Although the physical GPU has 24GB VRAM, we enforce a 6GB memory constraint in the simulation to emulate resource-limited edge devices.

Edge Network. We consider a cloud-edge-end MoE inference scenario with multiple edge servers and task-generating users randomly distributed in a fixed area. The edge servers are heterogeneous in computation, storage, and bandwidth capacities, and are categorized into three capability levels. The communication delay is calculated according to the transmitted data size and the bandwidth of the selected edge server.

Tasks and Models. The edge network serves five types of NLP inference tasks sampled from COPA [15], MultiRC [16], SQuAD [17], WikiQA [18], and WinoGrande [19]. We adopt the Switch Transformer-based MoE model switch-base-32 [2], which contains 12 MoE layers with 32 experts per layer. For different task distributions, we use the expert merging algorithm from [7] to generate merged MoE models deployed on edge servers.

Task Generation. Inference requests are generated by 10 users following a Poisson process. Each request is assigned a task type uniformly sampled from the five datasets, and its preference weight α_i is sampled from $\mathcal{U}(0, 1)$ to balance latency and service quality. All algorithms are evaluated using the same random seed, request sequence, task types, and preference weights to ensure fairness.

Baselines. We compare ONTO and JTOM with seven baselines, including four classical task-offloading/scheduling baselines, i.e., Multi-Armed Scheduling (MAS) [9], Least Connection Scheduling (LCS) [11], HERT [10], and Quality-based Greedy (Qual-Greedy), as well as three recent MoE-aware serving baselines, i.e., QoS-Quant [14], MoE-Load [13], and Multi-MoE [12]. All baselines are evaluated under the same distributed edge environment, request traces, model backbone, memory constraints, and metrics.

Metrics. We evaluate all algorithms on three Metrics: (1) the Weighted Cost $\max_{r_i \in \mathcal{R}} c_i - \sum_i \alpha_i \sum_s x_{i,s} q_s(r_i)$, i.e., the objective in our problem definition; (2) Process Time $\sum_{r_i \in \mathcal{R}} c_i - a_i$. It is important to note that for the JTOM algorithm, this Process Time explicitly incorporates the switching overhead (i.e., server downtime) incurred during the model updating process. This includes the time taken to download the new expert merging strategy from the cloud and reload the updated experts into the GPU memory. (3) Total Quality

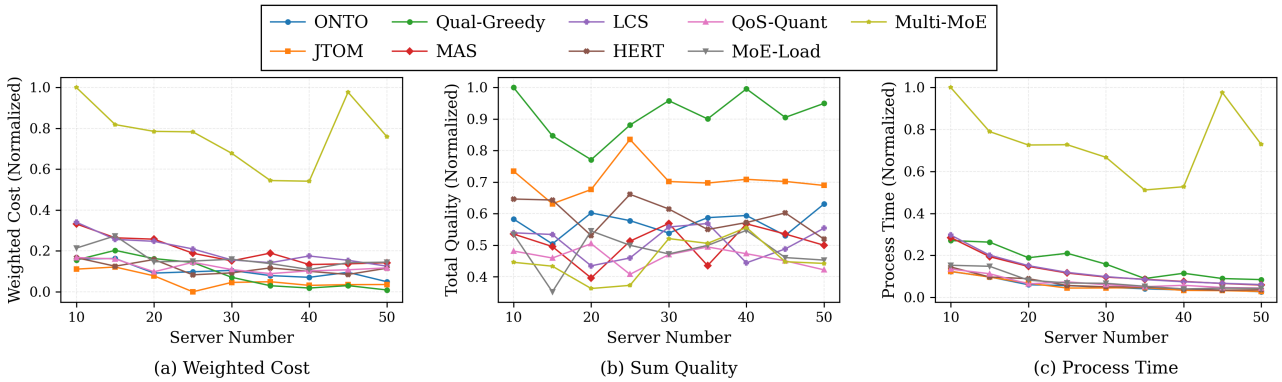


Fig. 3. Comparison of performance achieved by various offloading algorithms as the server number increases.

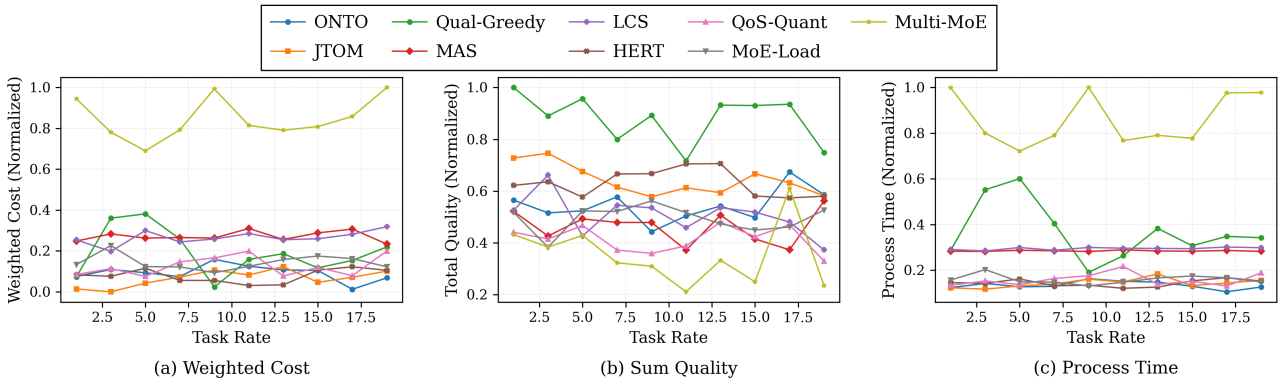


Fig. 4. Comparison across various task offloading algorithms as task generation rate increases.

$\sum_{r_i \in \mathcal{R}} q(r_i)$. Moreover, we evaluate the performance of our proposed algorithm in scenarios with varying numbers of servers and task-generating rates, where the task-generating rate represents the expected number of tasks generated by users at each time slot.

More implementation details are provided in the Appendix.

B. Analysis on Experiment Results

1) Effectiveness of Our Approach:

We compare our ONTO and JTOM against baseline algorithms across three key metrics: weighted cost, process time, and total quality, with varying server numbers and task-generating rates, as shown in Fig. 3 and Fig. 4. For a clear illustration, all the metrics reported are the normalized values.

Performance of JTOM with Various Server Numbers. As shown in Fig. 3, the baseline analysis reveals that our proposed JTOM framework consistently outperforms both generic scheduling strategies and modern MoE-aware serving baselines across various metrics. Specifically, as illustrated in Fig. 3(a), JTOM achieves the lowest normalized weighted cost, demonstrating its superior cost-efficiency and robust scalability as the edge network expands. Furthermore, in terms of Process Time (Fig. 3(c)), JTOM significantly reduces the inference latency compared to recent MoE-aware baselines like QoS-Quant, Multi-MoE, and MoE-Load, maintaining the lowest overall processing time alongside ONTO. This explicitly proves the critical advantage of tightly coupling adaptive expert merging with online multi-server task offloading, rather than relying solely on single-server optimizations. Crucially, this exceptional temporal efficiency does not come at the expense of accuracy. As shown in Fig. 3(b), JTOM maintains a remarkably high total quality, ranking second overall—surpassed only by the Quality-based Greedy algorithm, which naturally maximizes quality but suffers from worst-case, prohibitive process times. Compared to other efficiency-oriented MoE baselines, JTOM yields substantially higher task quality, perfectly showcasing its effectiveness in striking the balance between inference accuracy and system latency.

Performance of JTOM with Different Task Generating Rates. As shown in Fig. 4, evaluating the algorithms under varying task arrival rates further highlights the robustness of our JTOM framework against traffic surges. In terms of the overall objective, JTOM consistently sustains the lowest normalized weighted costs (Fig. 4(a)) even as the task generation rate intensifies, illustrating its exceptional scalability and cost-efficiency under heavy system loads. For Process Time (Fig. 4(c)), while modern MoE-aware baselines such as QoS-Quant and Multi-MoE suffer from increased latency under high task volumes due to their single-node bottlenecks, JTOM seamlessly distributes the inference pressure across the edge network. As a result, JTOM maintains a rigidly low and stable processing time alongside ONTO, regardless of the workload

TABLE III
CUMULATIVE MODEL-UPDATE DOWNTIME UNDER VARYING REQUEST RATES. ONTO IS OMITTED BECAUSE IT PERFORMS NO MODEL UPDATES BY DESIGN.

Downtime (s)	**Request Rate**									
Algorithm	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**
JTOM	180	162	180	126	180	162	180	198	180	180
Qual-Greedy	3276	486	666	306	324	342	324	288	306	306
MAS	180	180	180	180	180	198	180	198	180	180
LCS	180	198	198	180	180	180	198	234	180	180
HERT	180	180	180	180	180	180	180	180	180	198
QoS-Quant	180	198	198	180	198	216	216	198	180	252
MoE-Load	162	234	180	126	162	144	180	180	216	180
Multi-MoE	0	18	18	18	18	0	18	0	18	18

spikes. More importantly, this high-throughput efficiency does not compromise accuracy. As depicted in Fig. 4(b), JTOM delivers a highly stable and superior total quality score across all task rates. While it remains second only to the latency-prohibitive Qual-Greedy algorithm, JTOM drastically outperforms the other efficiency-oriented MoE baselines, which exhibit noticeable quality degradation when bombarded with high request rates. This robust performance across all metrics definitively proves that JTOM is highly effective in managing dynamic, high-traffic edge computing environments.

Model Update Overhead Evaluation. We further quantify the cost of dynamic model switching through the cumulative server downtime under varying request rates, as reported in Table III. ONTO is omitted because it adopts a static merging policy and therefore incurs no model-update downtime. The results show that JTOM maintains a consistently bounded update overhead across all tested request rates, with downtime ranging from 126 to 198 seconds. This indicates that the adaptive update mechanism in JTOM does not introduce high model-switching cost, even when the request rate changes. Compared with methods that update models more aggressively or less selectively, JTOM achieves a more stable update pattern by triggering merge-strategy updates only when the observed task distribution shift is sufficiently significant.

More importantly, the update overhead of JTOM should be interpreted together with its overall system performance. Although Multi-MoE incurs the smallest update downtime because it performs very limited model switching, its overall performance in terms of weighted cost, process time, and total quality is inferior to JTOM, as shown in Fig. 3 and Fig. 4. Therefore, the key advantage of JTOM is not merely reducing update downtime, but achieving a balanced trade-off between controlled switching overhead and long-term service quality improvement. By adaptively maintaining specialized merged MoE models on edge servers, JTOM effectively converts a small and stable amount of update overhead into improved task-specific inference quality and lower overall system cost.

2) Ablation Study Analysis:

To evaluate JTOM's merge strategy, we conduct an ablation study comparing it with the pure task offloading algorithm without the merge-strategy switch, ONTO.

As shown in Fig. 3 and Fig. 4, the ablation study indicates that JTOM outperforms ONTO in terms of lower process times, higher total quality, and reduced weighted costs across different server numbers and varying task rates. Specifically, JTOM achieves an average 30% improvement in normalized total quality compared to ONTO. Moreover, both algorithms maintain relatively low values for process time, with JTOM exhibiting a slight decrease in process time compared to ONTO. As a result, JTOM shows an average 8% improvement in the overall normalized weighted cost compared to ONTO. This underscores the critical role of the expert merging strategy decision in enhancing the algorithm's efficiency, quality, and cost-effectiveness.

3) Scalability of The Merge Strategy Decision Component: In order to further understand the scalability of the Merge Strategy Decision Component in the JTOM algorithm, we conducted experiments on several key parameters. Specifically, we explored the impact of adjusting the interval time m, the initial interval for performing merge updates δ, and the threshold g that triggers expert merge updates in JTOM. These parameters play a crucial role in determining the efficiency and effectiveness of the merge strategy, and their optimization is essential for achieving the best performance in the algorithm. In the following, we present the results and analysis of each of these parameters.

Impact of Interval Adjustment m. Table IV evaluates the effect of varying adjustment m in the JTOM algorithm's merge strategy decision component. Process time peaks at $m = 10$ (+19144.4) and is minimized at $m = 60$ (0.0). Weighted cost follows a similar trend, highest at $m = 10$ (+20451.4) and lowest at $m = 60$ (0.0). Sum quality increases with higher adjustments, reaching a maximum at $m = 60$ (+26157.4) and a minimum at $m = 0$ (+20515.2). Hence, $m = 60$ is the optimal adjustment for the JTOM algorithm, ensuring minimal process time and weighted cost, and maximal sum quality.

Impact of Interval δ. Table V assesses the influence of varying interval δ in the JTOM algorithm's merge strategy decision component. Process time is minimized at $\delta = 20$ (0.0) and maximized at $\delta = 50$ (+9711.0) and $\delta = 30$ (+7952.9). Sum quality peaks at $\delta = 20$ (+16232.1) and hits bottom at $\delta = 50$ (0.0). Weighted cost is lowest at $\delta = 20$ (0.0) and highest at $\delta = 50$ (+10522.6). Thus, $\delta = 20$ is the optimal interval parameter for the JTOM algorithm, yielding minimal process time and weighted cost, and maximal sum quality.

Impact of Threshold g. Table VI evaluates the impact of varying the update threshold g on the performance of the system. Process time is minimized at $g = 0.2$ (0.0) and maximized at $g = 0.3$ (+24338.7) and $g = 0.5$ (+16655.7). Total quality is highest at $g = 0.2$ (+57405.2) and lowest at $g = 0.3$ (0.0). Weighted cost is lowest at $g = 0.3$ (0.0) and highest at $g = 0.2$ (+30314.8). Therefore, $g = 0.2$ is the optimal update threshold, balancing minimal process time with maximal total quality and a reasonable weighted cost.

VII. RELATED WORK

With multiple expert networks, the MoE-based LLM provides users with specialized and high-quality inference services in data centers [2], [21]. Recently, how to deploy an MoE-based LLM on resource-constrained edge devices has

TABLE IV
IMPACT OF VARYING THE ADJUSTMENT m

Adjustment m	0	10	20	30	40	50	60
Process Time	+3311.0	+19144.4	+8554.4	+12520.2	+13511.0	+7849.1	**0.0**
Total Quality	+20515.2	0.0	+12469.7	+8112.6	+11112.9	+14574.2	**+26157.4**
Weighted Cost	+4312.3	+20451.4	+9958.0	+13422.3	+14263.1	+9148.2	**0.0**

TABLE V
IMPACT OF VARYING THE INTERVAL δ

Interval δ	10	20	30	40	50	60	70	80
Process Time	+1123.5	**0.0**	+7952.9	+6543.2	+9711.0	+6867.0	+8457.7	+1179.1
Total Quality	+11773.1	**+16232.1**	+6276.9	+3253.9	0.0	+3834.5	+1633.8	+10713.4
Weighted Cost	+1346.4	**0.0**	+8450.7	+7192.1	+10522.6	+7486.8	+9187.6	+1455.1

TABLE VI
IMPACT OF VARYING THE UPDATE THRESHOLD g

Threshold g	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Process Time	+3131.7	**0.0**	+24338.7	+1879.7	+16655.7	+3992.0	+4404.4	+12364.3	+5370.2
Total Quality	+45860.3	**+57405.2**	0.0	+48049.3	+18949.9	+42428.7	+41206.0	+12988.1	+31928.3
Weighted Cost	+23959.8	**+30314.8**	0.0	+25724.2	+11152.0	+22247.0	+22380.0	+7793.3	+18022.1

garnered significant attention [22]–[24]. The core challenges hidden behind the edge-based LLM deployment problem include addressing the gap between the over-large model size and limited resources on the edge side, as well as the corresponding task offloading problem.

Beyond edge-side MoE deployment and model compression, recent studies have also optimized MoE serving from QoS-aware, efficiency-oriented, and system-level perspectives. For example, Imani et al. [12] improve the QoS of serving multiple MoE LLMs through partial runtime reconfiguration on a single GPU, and Imani et al. [14] tune the quality-efficiency trade-off through partial quantization and mixed-precision expert placement across CPU and GPU. Huang et al. [13] improve MoE inference throughput through dynamic gating, expert buffering, and expert load balancing. Other works further study elastic MoE services [25], fine-grained expert offloading [26], expert sharding [27], adaptive MoE runtime systems [28], and large-scale disaggregated expert-parallel serving [29]. These studies mainly optimize MoE execution within a single node, centralized GPU cluster, or resource-abundant data-center environment, whereas our work focuses on resource-constrained edge inference, where adaptive expert merging and multi-server online task offloading must be jointly optimized across heterogeneous edge servers.

To address the model-size challenge in edge deployment, some compression techniques have been proposed, including quantization [23], [30], [31], pruning [32], [33], knowledge distillation [34], and model splitting [22], [35]. Quantization [23], [30], [31] compresses model parameters from high precision (FP32) to low precision (e.g., INT8) to reduce storage requirements but introduces significant computing overhead for quantization/dequantization during inference and potential accuracy loss. Pruning [32], [33] reduces the model size by removing parameters, which directly leads to information loss. Knowledge distillation [34] transfers the knowledge from a large model to a smaller edge model, maintaining knowledge integrity but requiring substantial computational resources and time for training on user data. Model splitting [22], [35], [36] divides model parameters across multiple devices to alleviate storage constraints, but this approach incurs high communication overhead. Exploiting the sparsity characteristic of experts in MoE models, some works [3]–[7] merge experts to reduce their number, thereby decreasing overall expert parameters while maintaining model quality. This technique shows potential for solving issues related to resource constraints, enabling deployment of LLMs on edge devices. However, existing studies focus on single-task, single-machine settings, making them unsuitable for complex edge scenarios with multiple tasks and servers.

As for the task offloading problem for LLM inference at the edge, several works have been proposed [37]–[39]. In [37], the energy consumption issues under various offloading configurations and task requirements are investigated in an intelligent edge testing platform. Then, the LLM offloading problem is formulated as a Multi-Armed Bandit problem to identify energy-efficient offloading configurations and reduce energy consumption. However, all of the works mentioned above are not specifically designed for the MoE framework. As presented in our motivation, simply applying a conventional task offloading strategy without considering the different activation rates of various tasks on different experts fails to leverage the full potential of the MoE architecture. After a deep investigation of the task preferences of the experts in the MoE structure, our work presents a co-design on the task

offloading strategy and expert merging method.

VIII. ACKNOWLEDGE

This work was supported in part by the National Key R&D Program of China (No. 2023YFB2703600), National Natural Science Foundation of China under Grant 62572280, U24A20244.

IX. CONCLUSION

In this paper, we tackled the complex problem of deploying MoE-based LLMs on edge devices by leveraging expert merging techniques. Our study reveals that static merging policies fall short of addressing the dynamic and heterogeneous nature of edge environments. To mitigate this, we introduce a flexible framework that allows each edge server to adaptively merge experts according to the tasks assigned and online offload the inference tasks based on the recently merged LLM. This joint optimization strategy ensures that resource-limited edge servers can provide efficient, low-latency, and high-accuracy inference services. We validated our approach through theoretical proofs and a series of experiments on multiple datasets, demonstrating at least a 16% reduction in overall weighted cost over existing methods. Extending to incorporate dynamic cloud-edge collaboration or multi-device cooperative inference, alongside applying it to other types of AI-based network services, will be the focus of our future work.

REFERENCES

[1] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations*, 2017.

[2] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *CoRR*, vol. abs/2101.03961, 2021.

[3] M. Muqeeth, H. Liu, and C. Raffel, "Soft merging of experts with adaptive routing," *arXiv preprint arXiv:2306.03745*, 2023.

[4] A. Jolicoeur-Martineau, E. Gervais, K. Fatras, Y. Zhang, and S. Lacoste-Julien, "Population parameter averaging (papa)," *arXiv preprint arXiv:2304.03094*, 2023.

[5] S. He, R.-Z. Fan, L. Ding, L. Shen, T. Zhou, and D. Tao, "Merging experts into one: Improving computational efficiency of mixture of experts," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, Dec. 2023, pp. 14 685–14 691.

[6] S. Park, "Learning more generalized experts by merging experts in mixture-of-experts," *arXiv preprint arXiv:2405.11530*, 2024.

[7] P. Li, Z. Zhang, P. Yadav, Y.-L. Sung, Y. Cheng, M. Bansal, and T. Chen, "Merge, then compress: Demystify efficient smoe with hints from its routing policy," *arXiv preprint arXiv:2310.01334*, 2023.

[8] Y. Jin, C.-F. Wu, D. Brooks, and G.-Y. Wei, "s^3: Increasing gpu utilization during generative inference for higher throughput," *Advances in Neural Information Processing Systems*, vol. 36, pp. 18 015–18 027, 2023.

[9] X. Wang, J. Ye, and J. C. Lui, "Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1199–1208.

[10] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[11] G. Singh and K. Kaur, "An improved weighted least connection scheduling algorithm for load balancing in web cluster systems," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 3, p. 6, 2018.

[12] H. Imani, J. Peng, P. Mohseni, A. Amirany, and T. El-Ghazawi, "QoS-efficient serving of multiple mixture-of-expert LLMs using partial runtime reconfiguration," in *Proceedings of the 42nd International Conference on Machine Learning*, vol. 267, 2025, pp. 26 433–26 445.

[13] H. Huang, N. Ardalani, A. Sun, L. Ke, H.-H. S. Lee, S. Bhosale, C.-J. Wu, and B. Lee, "Toward efficient inference for mixture of experts," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 84 033–84 059.

[14] H. Imani, A. Amirany, and T. El-Ghazawi, "Mixture of experts with mixture of precisions for tuning quality of service," in *2024 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2024, pp. 1–6.

[15] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *2011 AAAI spring symposium series*, 2011.

[16] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 252–262.

[17] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[18] Y. Yang, W.-t. Yih, and C. Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 2013–2018.

[19] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[20] F. Z. Xin Feng and Y. Xu, "Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times," *International Journal of Production Research*, vol. 54, no. 12, pp. 3706–3717, 2016.

[21] L. Li, S. Wang, C. Li, Y. Yuan, and G. Wang, "Dc-lora: Domain correlation low-rank adaptation for domain incremental learning," *High-Confidence Computing*, vol. 5, no. 4, p. 100270, 2025.

[22] S. Ye, J. Du, L. Zeng, W. Ou, X. Chu, Y. Lu, and X. Chen, "Galaxy: A resource-efficient collaborative edge ai system for in-situ transformer inference," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024.

[23] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 941–22 954, 2022.

[24] A. M. Anisha and S. Chellappan, "Scale-aware gaussian mixture loss for crowd localization transformers," *High-Confidence Computing*, vol. 5, no. 3, p. 100296, 2025.

[25] X. Xia, J. Liu, X. Hou, P. Tang, M. Zhang, W. Wang, and C. Li, "Moe-prism: Disentangling monolithic experts for elastic moe services via model-system co-designs," 2025. [Online]. Available: https://arxiv.org/abs/2510.19366

[26] H. Yu, X. Cui, H. Zhang, H. Wang, and H. Wang, "Taming latency-memory trade-off in moe-based llm serving via fine-grained expert offloading," in *Proceedings of the 21st European Conference on Computer Systems*, ser. EUROSYS '26. New York, NY, USA: Association for Computing Machinery, 2026, p. 176–191.

[27] O. Balmau, A.-M. Kermarrec, R. Pires, A. L. E. Santo, M. de Vos, and M. Vujasinovic, "Accelerating moe model inference with expert sharding," in *Proceedings of the 5th Workshop on Machine Learning and Systems*, ser. EuroMLSys '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 192–199. [Online]. Available: https://doi.org/10.1145/3721146.3721940

[28] C. Hwang, W. Cui, Y. Xiong, Z. Yang, Z. Liu, H. Hu, Z. Wang, R. Salas, J. Jose, P. Ram *et al.*, "Tutel: Adaptive mixture-of-experts at scale," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 269–287, 2023.

[29] R. Zhu, Z. Jiang, C. Jin, P. Wu, C. A. Stuardo, D. Wang, X. Zhang, H. Zhou, H. Wei, Y. Cheng, J. Xiao, X. Zhang, L. Liu, H. Lin, L.-W. Chang, J. Ye, X. Yu, X. Liu, X. Jin, and X. Liu, "Megascale-infer: Efficient mixture-of-experts model serving with disaggregated expert parallelism," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 592–608. [Online]. Available: https://doi.org/10.1145/3718958.3750506

[30] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," 2021.

[31] Z. Zheng, X. Cui, S. Zheng, M. Li, J. Chen, X. Chen *et al.*, "Moqa: Rethinking moe quantization with multi-stage data-model distribution awareness," *arXiv preprint arXiv:2503.21135*, 2025.

[32] T. Chen, S. Huang, Y. Xie, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei, "Task-specific expert pruning for sparse mixture-of-experts," 2022. [Online]. Available: https://arxiv.org/abs/2206.00277

[33] Y. Zou, S. Qi, Y. Yuan, D. Wang, S. Shen, L. Wu, S. Guo, and D. Yu, "Fed-moe: Efficient federated learning for mixture-of-experts models via empirical pruning," in *Parallel and Distributed Computing, Applications and Technologies: 25th International Conference, PDCAT 2024, Hong Kong, China, December 13–15, 2024, Proceedings*, vol. 15502. Springer Nature, 2025, p. 128.

[34] Z. Xie, Y. Zhang, C. Zhuang, Q. Shi, Z. Liu, J. Gu, and G. Zhang, "Mode: A mixture-of-experts model with mutual distillation among the experts," 2024.

[35] X. Yuan, W. Kong, Z. Luo, and M. Xu, "Efficient inference offloading for mixture-of-experts large language models in internet of medical things," *Electronics*, vol. 13, no. 11, 2024.

[36] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, and M. Xu, "Edgemoe: Empowering sparse large language models on mobile devices," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2025.

[37] X. Yuan, H. Li, K. Ota, and M. Dong, "Generative inference of large language models in edge computing: An energy efficient approach," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, 2024, pp. 244–249.

[38] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Cached model-as-a-resource: Provisioning large language model agents for edge intelligence in space-air-ground integrated networks," 2024.

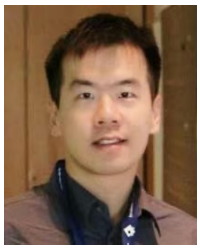
[39] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Ruhle, L. V. S. Lakshmanan, and A. H. Awadallah, "Hybrid llm: Cost-efficient and quality-aware query routing," 2024. [Online]. Available: https://arxiv.org/abs/2404.14618

Ruirui Zhang received the B.S. degree from Taishan College, Shandong University, in 2022. She is currently pursuing a Ph.D. degree with the School of Computer Science and Technology, Shandong University. Her research interests include mobile edge computing, Internet of Things, and federated learning.

Yifei Zou received the BEng degree from Wuhan University, China, in 2016, and the Ph.D. degree from The University of Hong Kong, China, in 2020. He is currently an associate professor at the School of Computer Science and Technology, Shandong University, Qingdao, China. His research interests include edge intelligence, Internet of agents, and distributed computing.

Peng Li is currently a Professor with the Xi'an Jiaotong University, China. He received the PhD degree in computer science from The University of Aizu, Japan. His research interests mainly focus on wired/wireless networking, cloud/edge computing, distributed AI systems, and blockchain. He has authored or co-authored over 100 papers in major conferences and journals. He won the 2020 Best Paper Award of IEEE Transactions on Computers. He serves as the chair of SIG on Green Computing and Data Processing in IEEE ComSoc Green Communications and Computing Technical Committee. He is a guest editor of IEEE Journal of Selected Areas on Communications, the editor of IEEE Open Journal of the Computer Society and IEICE Transactions on Communications. He is a senior member of IEEE.

Fahao Chen received the Ph.D. degree from The University of Aizu, Japan. His research interests mainly focus on cloud/edge computing, graph learning, and distributed machine learning systems. He is awarded the Japan Society for the Promotion of Science (JSPS) Research Fellowship for Young Scientists.



Yupeng Li was a postdoctoral research fellow at the University of Toronto. He is an assistant professor with the Department of Interactive Media at Hong Kong Baptist University. He is interested in studying trustworthy machine learning in networking and social computing. Dr. Li has received the distinction of Distinguished TPC Member of the IEEE INFOCOM 2022 and 2024. He serves on the technical committees of some top conferences in computer science, for example, TPC Track Chair of ICDCS 2024 and TPC Chair of the SocialMeta 2023. His works have been accepted in prestigious venues, such as MobiHoc, INFOCOM, ICDCS, JSAC, and ToN.

Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota Twin Cities in 2000 and 2002, respectively. She is a professor in the School of Computer Science and Technology, Shandong University, Qingdao, China. She has published more than 170 peer-reviewed papers. Her current research interests include cyber physical systems, wireless and mobile computing, sensor networking, wireless and mobile security, and algorithm design and analysis. She worked as a professor with the Department of Computer Science, the George Washington University, Washington, DC, USA, from 2013 to 2017. She worked as a program director for the US National Science Foundation (NSF) from April to October in 2006 (full time), and from April 2008 to May 2010 (part time). She received the NSF CAREER Award in 2004. She is Fellow of IEEE and a member of ACM.

Falko Dressler received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. He is a full professor and Chair for Data Communications and Networking at the School of Electrical Engineering and Computer Science, TU Berlin. Dr. Dressler has been associate editor-in-chief for IEEE Trans. on Mobile Computing and Elsevier Computer Communications as well as an editor for journals such as IEEE/ACM Trans. on Networking, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow as well as an ACM Distinguished Member. He is a member of the German National Academy of Science and Engineering (acatech). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking (radio, visible light, molecular communications) and embedded system design (from microcontroller to Linux kernel) with applications in ad hoc and sensor networks, the Internet of Things, and cooperative autonomous driving systems.

Dongxiao Yu received the B.S. degree in 2006 from the School of Mathematics, Shandong University and the Ph.D degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a professor in the School of Cryptography Science and Engineering, Shandong University. His research interests include wireless networks, distributed computing and graph algorithms.