SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

Special Topic: User-Centric Intelligent Networking

Learning to schedule: interference-aware optimization for edge AI inference on shared GPUs

Mingtao JI[1,2], Hehan ZHAO[2], Lei JIAO[3*], Bin TANG[1*], Zhihao QU[1], Zhuzhong QIAN[2*] & Baoliu YE[2]

[1]*College of Computer Science and Software Engineering, Hohai University, Nanjing 211100, China*
[2]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China*
[3]*Center for Cyber Security and Privacy, University of Oregon, Eugene, OR 97403, USA*

Abstract
While deep neural networks (DNNs) are increasingly deployed on graphics processing units (GPUs) at the network edge, the performance interference caused by co-locating and executing multiple models on a single GPU is often overlooked, leading to high inference latency and excessive energy consumption. Pursuing the best performance faces several challenges, including characterizing the non-linear interference-incurred latency, dispatching the unpredictable inference workloads, and balancing the different trade-offs related to performance and accuracy. In this study, we first construct an interference-incurred latency model via real-world profiling and measurements. With insights from this model, we formulate a time-varying integer program to minimize the long-term total cost of the edge AI inference system, including the inference latency, the inference error rate, the query-dispatching communication cost, and the energy consumption, subject to resource and workload constraints. We then propose a set of polynomial-time online algorithms that continuously make fractional control decisions for each time slot based on the feedback from the previously applied decisions and strategically round these decisions into integers. We conduct a formal theoretical analysis and prove that *both* the time-averaged gap between the total cost incurred by our online decisions and the total cost of the series of one-shot offline optimums *and* the time-averaged constraint violation gradually diminish over time. Extensive experiments on real-world testbeds and datasets demonstrate that the proposed algorithms can reduce the total cost by 40% compared to state-of-the-art and heuristic methods, with only a moderate computational overhead.

Keywords edge AI, inference, interference, GPU, online learning

Citation . Sci China Inf Sci, for review

1 Introduction

Deep neural networks (DNNs) are increasingly deployed at the network edge, driving the rapid advancement of diverse applications, including autonomous driving [1], smart home [2], virtual reality [3], and augmented reality [4]. To support such applications, service providers and operators extensively rely on graphics processing units (GPUs) at edge servers for low-latency and high-quality inference. For example, NVIDIA introduced the multi-process service (MPS) [5] to enable multi-DNN co-location on a single GPU, thereby optimizing resource utilization and enhancing overall efficiency.

However, the *interference* caused by co-locating multiple inference workloads on one GPU can significantly degrade the inference speed. As illustrated in Figure 1, the inference speeds of DNN A and DNN B decrease on Edge 1, compared to their standalone executions on Edges 2 and 3, respectively. Prior research [6] indicated that the inference speed can decrease by 35% owing to the co-location, making it difficult for service providers to meet the service level agreements (SLAs). In fact, service providers need to address several challenges when deploying models and scheduling inference on edge GPUs, including characterizing the interference-aware inference latency, dispatching the unpredictable inference workloads, and balancing the different trade-offs related to performance and accuracy, as elaborated respectively as follows.

First, it is difficult to predict the inference latency of a model in the presence of interference [7]. The inference latency of a DNN model is often affected by multiple factors, including the model structure, GPU scheduling, and cache and memory utilization when co-located with other models [6, 8]. One may profile the impact of interference beforehand, but there can be excessively many cases for profiling. The profiling results are often non-linear, making

* Corresponding author (email: ljiao2@uoregon.edu, cstb@hhu.edu.cn, qzz@nju.edu.cn)

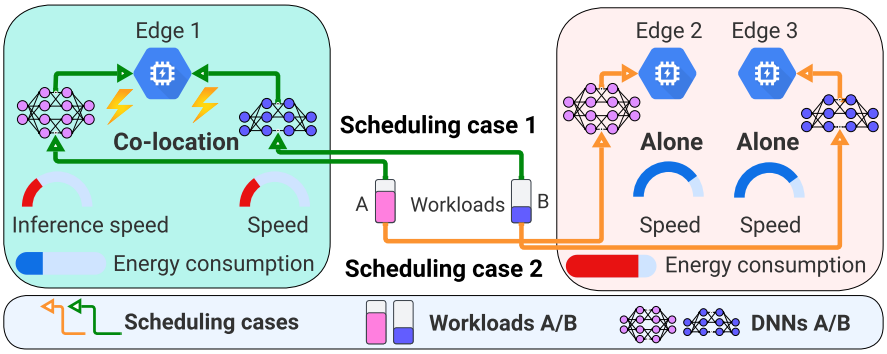


Figure 1 Interference-incurred trade-off: energy versus speed

it difficult to be applied analytically at runtime to mitigate the interference. The actual runtime degradation could also deviate from and invalidate the offline profiling results.

Second, models need to be placed before the inference queries are observed [9]. Because the number and distribution of inference queries often vary with time, the model placements must be updated dynamically [10, 11]; yet, a model could fail to be placed at the best location in hindsight [12], as the inference queries that arrive at an edge can change after the models are placed on that edge. One could then strategically dispatch the inference queries to the most suitable models given the model placements, but that may not always be possible due to the heterogeneous edge capacities for hosting models and serving queries [13]. The fact that the inference queries are revealed a posteriori is challenging to handle.

Third, two types of trade-offs need to be considered and balanced when determining model placements and inference query dispatching. (i) Overhead against speed. As shown in Figure 1, activating many GPUs can provide high inference speed, but can result in a significant energy footprint owing to the static base power consumption per GPU. In contrast, using a relatively small number of GPUs where one GPU hosts multiple models can decrease inference speed but may have better throughput-per-watt efficiency. (ii) Resource against accuracy. To deploy the multiple types or versions of candidate DNN models [11], high-version models (e.g., ResNet152) provide good accuracy but need massive GPU resources; and low-version models (e.g., ResNet18) need less GPU resources but have low inference accuracy.

Existing studies cannot effectively tackle the aforementioned challenges. Some studies on inference systems [7, 8, 14–18] do not focus on edge infrastructures with heterogeneous capacities and fluctuating inference workloads. As a result, they often lead to high energy consumption and high resource overhead. Other studies on edge inference [10, 12, 19–22] ignore the specific characteristics of GPU, and particularly, the interference when multiple models are co-located on a shared GPU. Therefore, they lead to low inference speed and are unable to meet user latency requirements in the dynamic edge environment [23].

In this study, we first demonstrate and validate the complexity of interference-aware latency by conducting real-world measurements on co-located DNNs on NVIDIA GPUs. With the insights from the experiments, we formulate a time-varying integer program with the objective of minimizing the long-term total cost of the edge AI system, including the inference latency, the inference error rate, the query-dispatching communication cost, and the energy consumption of the system, subject to resource and workload constraints. The present formulation is general and makes no assumptions about spatial heterogeneities in the system and temporal input fluctuations, featuring posterior inputs to capture the aforementioned challenges.

We further propose a group of algorithms that operate in polynomial time to jointly solve this problem in an online manner. We design (i) an online learning mechanism based on a modified saddle-point method, which can continuously make the control decisions for the next time slot without needing to know the inputs for that time slot, based on the feedback from the previous control decisions via primal-dual updates, and (ii) a randomized rounding mechanism which converts the fractional control decisions into integers while ensuring that the solutions can closely approximate the optimums in expectation.

Further, through rigorous theoretical analysis of the system's performance, we prove the sublinear *regret*, i.e., the time-averaged gap between the total cost incurred by the online decisions and the total cost incurred by the series of one-shot offline optimal decisions gradually diminishes over time; in other words, the online decisions asymptotically approach these optimums in the long run. We also establish the sublinear *fit*, i.e., the time-averaged constraint violation incurred by the online decisions diminishes as time goes on.

Finally, we implement a comprehensive testbed using real hardware (i.e., NVIDIA GeForce RTX 2080 Ti, RTX 3090 & V100), deploy AI models (i.e., ResNet18 $\sim$ ResNet152), and evaluate the inference latency when these models are co-located. Based on the testbed and the realistic traces, such as London dataset [24] and network statistics [25, 26], the evaluation is conducted from the following aspects: (i) the proposed approach saves system cost in real time, achieving a cumulative reduction of 40%$\sim$60% compared to baselines, and 10%$\sim$30% compared to state-of-the-art methods; (ii) the proposed approach scales effectively with inputs and outperforms others across different scenarios; (iii) the online learning component exhibits the lowest regret, achieving sublinear growth in both regret and fit; (iv) the developed algorithms run efficiently and finish within several seconds, which is acceptable for a 15-minute-long time slot.

This paper is organized as follows. Section 2 presents a model and formulation of the problem. In Section 3, a set of online algorithms is proposed and designed to solve the problem. Section 4 presents a theoretical analysis of the algorithms. In Section 5, the performance is verified. Section 6 presents a discussion of related research. Section 7 presents the conclusions. For ease of understanding, we provide the mind map of this section in Figure 2.

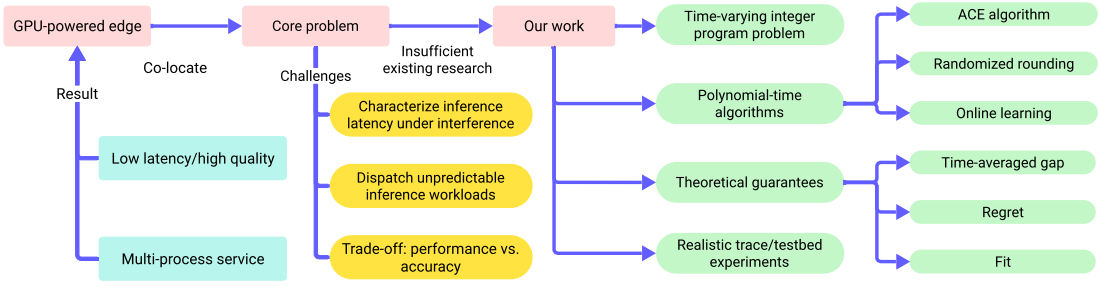


Figure 2 Mind map of the introduction section

2 System model and problem formulation

Our major notations used in the system model are shown in Table 1 below.

2.1 Interference-aware latency

Co-locating models or inference workloads on a GPU incurs performance interference, due to hardware resource contention when processing multiple inference queries [7,8]. Figure 3 illustrates the typical workflow [6], consisting of two stages: (i) the GPU scheduler serially assigns kernels to execution units, named Streaming Multiprocessors [27], where scheduling more kernels leads to longer delays; (ii) the GPU remains active to execute the inference queries, while accessing the L2 cache/DRAM, and the GPU active time is sensitive to the data volume. We experiment with and discuss these two stages below.

Kernel scheduling: We begin by investigating scheduling delays across different numbers of co-located models. Following a prior experimental setup [6], we deploy from 1 to 5 DNN inference models on a GPU server, as shown in Figure 4. Figure 5 illustrates how the scheduling delay for each model increases with the number of co-located models. Specifically, when the number of co-located models increases from 2 to 5, the inference delay rises by 1% to 30%. This increase can be attributed to two key factors. (i) kernel contention: as more models are placed on the same GPU, their kernels compete for scheduling on the Streaming Multiprocessors, resulting in heightened inference

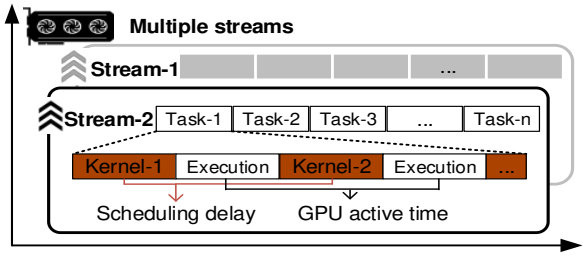


Figure 3 CUDA stream mechanism for multiple workloads

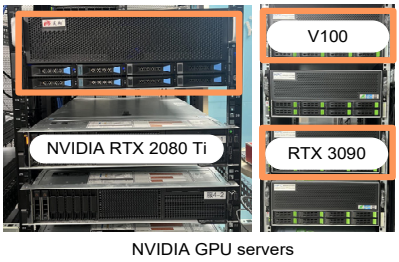


Figure 4 Real devices

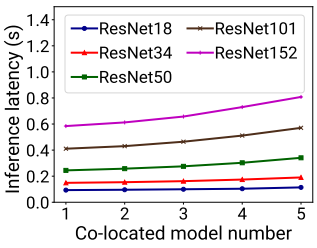


Figure 5 Number of kernels

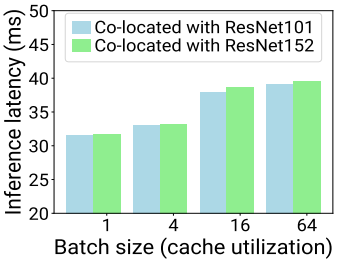


Figure 6 Utilization of memory

delays; (ii) model complexity: deeper models, such as ResNet152, contain more kernels than shallower models like ResNet18 and intensify scheduling contention, further increasing the delay.

Inference execution: We next explore the factors that influence GPU active time during inference execution. In this phase, the GPU loads data from the L2 cache and DRAM, where cache utilization impacts the data loading latency. We investigate GPU active time under varying cache utilization conditions by adjusting the batch size of inference queries. Two groups of experiments are conducted: In the first group, ResNet50 and ResNet101 are deployed concurrently on one GPU; in the second group, ResNet50 and ResNet152 are co-located on another GPU. For both experiments, the batch size of ResNet50 is fixed to 16, while the batch size for ResNet101 and ResNet152 is varied from 1 to 64. As shown in Figure 6, inference latency increases with batch sizes, as a large batch size incurs more contention of L2 cache and DRAM.

Interference-aware latency model: With the above experimental results, it is challenging to construct an accurate closed-form function to express the interference-aware inference latency. Figure 5 seems to be a piecewise linear function, where the breaking points are at the integers, i.e., the number of the co-located models: For example, if j refers to the ResNet18 model and i refers to the GPU, then we have the inference delay as $f^s_{i,j}(1) = 90$ ms, $f^s_{i,j}(2) = 94$ ms, $f^s_{i,j}(3) = 98$ ms, $f^s_{i,j}(4) = 102$ ms, and $f^s_{i,j}(5) = 114$ ms. While these values vary for different models and different GPUs, what is more important is that this figure is only for co-locating the same models on a single GPU. In reality, the co-location can be for different (types of) models, further complicating the results.

We note that, as will be described in the following sections, we are focused on the long-term optimization of the system in a dynamic setting in this paper. This provides an opportunity to treat the interference-aware delay differently. While the structural complexity of interference could be fatal in the *offline* world, this does not automatically make the *online* problem intractable or preclude the use of *online learning* techniques. Online learning is inherently for handling sequential decision making with revealed costs. It does not need to find the optimal model placement/co-location for each round; it just tries to learn to play a sequence that performs nearly as well as the best single actions for each round chosen in hindsight. As a result, we model the interference-aware inference latency as a *posteriori* function:

$$\Gamma_{i,j,t} = c_{i,j,t} \cdot y_{i,j,t}, \tag{1}$$

where $y_{i,j,t}$ is the decision variable denoting whether model j is placed on GPU i at time t, and $c_{i,j,t}$ is the input, i.e., the incurred delay perceived by model j if this model is placed on GPU i at time t, taking into account other co-located models' interference. We do not know $c_{i,j,t}$ when making the decision $y_{i,j,t}$ at t, and only know $c_{i,j,t}$ *after* making the decision $y_{i,j,t}$. That is, $c_{i,j,t}$ is allowed to take any value of the delay that actually occurs and is subsequently observed. While in nature $c_{i,j,t}$ may depend on $y_{i,j,t}$, we treat $c_{i,j,t}$ as exogenous in this work for more efficient and theoretically sound algorithms.

Discussion on modeling interference: The "linearity" in the latency model is only in the *decision variables*, and the coefficients themselves can be non-linear functions of the current co-location set. The total latency at time t is $\sum_i \sum_j \Gamma_{i,j,t} = \sum_i \sum_j (c_{i,j,t} \cdot y_{i,j,t})$, where $y_{i,j,t}$ is a binary decision variable indicating whether model j is placed on GPU i at time t, and $c_{i,j,t}$ is the observed actual latency coefficient for model j on GPU i, which *inherently captures the co-location interference* at that specific time and includes all combinatorial interference effects from other models co-located on that same GPU.

2.2 Edge AI inference system

Edge AI with GPU: Aligning with previous system scenarios [16,19], we consider GPU servers as the edge nodes operated and managed by edge AI service providers. Let the set of all GPU servers in our system be denoted as $\mathcal{I} = \{1, 2, \ldots, I\}$. These servers may be located at either the same or different locations. For each GPU server i of any location, the model hosting capacity is denoted by Ω_i. We then define a set of consecutive time slots in the

Table 1 Major notations used in our formulation

Symbol	Description
$\mathcal{T}, \mathcal{I}, \mathcal{J}$	Set of consecutive time slots, GPU servers, DNN models
$\Gamma_{i,j,t}$	Inference latency for DNN model j on GPU i at time slot t
$c_{i,j,t}$	Incurred delay perceived by model j when this model is placed on GPU i at time slot t
Ω_i	Model hosting capacity of edge GPU server i
$\sigma_{i,i',t}$	Latency between GPU server i and i' at time slot t
$a_{j,t}$	Error rate of the DNN model j at time slot t
d_j	Resource demand of DNN model j
$r_{i,t}$	Number of queries submitted to GPU server i at time slot t
$e_{i,t}$	Operational cost of edge GPU server i at time slot t
R_i	Processing capacity of edge GPU server i
Decision	**Description**
$x_{i',i,t}$	Number of inference queries dispatched from GPU server i' to i at time slot t
$y_{i,j,t}$	Whether to place DNN model j on GPU server i at time slot t
$z_{i,t}$	Whether edge server i is active at time slot t

edge system, denoted by $\mathcal{T} = \{1, 2, \ldots, T\}$. At time slot t, $\sigma_{i,i',t}$ represents the query-dispatching communication latency between GPU server i and i'. If servers i and i' are located at different locations, the communication latency is $\sigma_{i,i',t} \neq 0$. The communication latency is given by $\sigma_{i,i',t} = \frac{D}{B_{i,i',t}}$, where $B_{i,i',t}$ is the bandwidth between the two servers, and D is the data volume. If GPU servers i and i' are situated at the same location, the latency is zero.

Multi-version DNN models: According to prior studies [9, 13], there exist multiple versions of DNN models (e.g., YOLOv1∼YOLOv8). Even for YOLOv5, there exist versions YOLOv5-N∼YOLOv5-X for different scenarios. We denote the set of multiple-version DNN models by $\mathcal{J} = \{1, 2, \ldots, J\}$. Typically, higher-version DNN models consume more resources, thereby achieving lower error rates, whereas lower-version models consume fewer resources and have higher inference error rates. We use $a_{j,t}$ to denote the error rate of the j-th version DNN model. Note that it is a posterior parameter revealed only after the model resolves the inference query. Specifically, the inference error rate is related to two factors: (1) It depends on the selected model. In practice, a model family usually includes multiple versions, and larger versions generally achieve lower error rates. (2) The error rate is also affected by concept drift, where the relationship between features and labels changes over time in dynamic environments. As a result, the error rate is inherently time-varying during online inference. Actually, there exist many practical scenarios of edge inference where such feedback is immediately available post-execution (e.g., soft keyboard input, autonomous driving, smart manufacturing). Additionally, model j's resource demand is denoted by d_j.

Control decisions: We consider three types of control decisions in the edge AI system: $y_{i,j,t} \in \{0, 1\}$ indicates whether the service providers deploy DNN model j on server i at time slot t; $z_{i,t} \in \{0, 1\}$ indicates whether to activate GPU server i at time slot t; $x_{i,i',t} \in \mathbb{Z}_+$ indicates the number of inference queries dispatched from GPU server i to GPU server i' at time slot t.

Cost of communication and resolving: At each time slot t, $r_{i,t}$ inference queries are submitted to edge GPU server i by nearby users. This query volume is time-varying, as evidenced by representative real-world traces, including data from London underground stations [24] and Google's production system [28]. The service providers make decisions on processing them locally or dispatching some/all of them to the other suitable GPU server $i' \in \mathcal{I} \setminus \{i\}$, which incurs query-dispatching communication cost. The total communication cost can be represented by $\sum_{i,i'} \sigma_{i,i',t} x_{i,i',t}$. According to Subsection 2.1, the inference latency perceived by model j is denoted by $\Gamma_{i,j,t}$. Then, the total inference latency of the models at GPU server i is represented by $\sum_j \Gamma_{i,j,t}$.

Cost of energy and error rate: The total energy consumption depends on the GPU servers activated/selected by the service providers, denoted by $\sum_i e_{i,t} z_{i,t}$, where $e_{i,t}$ represents the operational cost of GPU server i at time slot t. In general, the GPU server with a high capability results in slight interference, but consumes massive energy cost afforded by the service providers. On the contrary, a low-energy GPU server can incur heavy interference. For each activated GPU server, the deployment of models should respect its capability, and the inference error rate depends on the model selected by the service providers. Since $a_{j,t}$ denotes the error rate of the model j at time slot

t, the total error rate is calculated by $\sum_{i,j} a_{j,t} y_{i,j,t}$. Generally, error rates are posteriors. After deploying decisions, they can be obtained through offline label generation [29] in the cloud, user feedback mechanisms [30], etc. We also note that $a_{j,t}$, i.e., the error rate of the model j at the time slot t, does not always need to represent *real-time* error rates. For many models deployed in production, their baseline accuracy on representative datasets can be pre-measured offline and updated periodically.

Therefore, the **total system cost** encompasses the four aspects above, i.e., $\sum_t \{\sum_{i,j} \Gamma_{i,j,t} + \sum_i \sum_j a_{j,t} y_{i,j,t} + \sum_i \sum_{i'} \sigma_{i,i',t} x_{i,i',t} + \sum_i e_{i,t} z_{i,t}\}$ that, as the goal of the edge AI system, should be as small as possible.

2.3 Problem formulations

However, it is a challenge to achieve the goal above, which is ideally constrained by the equations:

At time slot t, all the submitted $r_{i,t}$ inference queries to GPU server i should be dispatched:

$$\sum_{i'} x_{i,i',t} = r_{i,t}, \quad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}. \tag{a}$$

Furthermore, to reduce energy consumption, the service providers only activate some of the GPU servers to deploy DNN models. Only when edge GPU server i is activated, these DNN models can be deployed on it, and we have the following equation:

$$\sum_j y_{i,j,t} \leqslant |\mathcal{J}| z_{i,t}, \quad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}. \tag{b}$$

Then, the inference queries can only be dispatched to edge i where at least one model is deployed:

$$\sum_{i'} x_{i',i,t} \leqslant R_i \sum_j y_{i,j,t}, \quad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}, \tag{c}$$

where R_i represents the processing capacity of edge GPU server i.

Even if Multi-Process Service allows multiple co-located models within one GPU, the deployment of DNN models on GPU server i respects the capacity as follows:

$$\sum_j y_{i,j,t} d_j \leqslant \Omega_i, \quad \forall t \in \mathcal{T}, \forall i \in \mathcal{I}, \tag{2}$$

where d_j represents the model j's resource demand and Ω_i denotes the model hosting capacity of edge i.

Problem formulation and challenges: However, solving this problem with the above constraints on the fly is challenging because some inputs cannot be revealed to us when decisions are made (e.g., $c_{i,j,t}$, $\sigma_{i,i',t}$, $a_{j,t}$, and $r_{i,t}$). Aligning with prior studies [11, 22], we consider an analogous solvable problem with the constraints breaking down the barriers between time slots. That is to say, the constraints are satisfied from a long-term aspect instead of a single time slot, which provides a new scope for a problem with unknown inputs. Following this, Constraints (a) $\sim$ (c) are converted into the following long-term Constraints $C_1 \sim C_4$. We reformulate the problem $\mathscr{P}$ as follows:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \ & \sum_{t=1}^{T} \{\sum_{i,j} (\Gamma_{i,j,t} + a_{j,t} y_{i,j,t}) + \sum_{i,i'} \sigma_{i,i',t} x_{i,i',t} + \sum_i e_{i,t} z_{i,t}\}, \\
\text{s.t.} \ \ & C_1 : \sum_{t=1}^{T} g_t^{1,i} := \sum_{t=1}^{T} \{r_{i,t} - \sum_{i'} x_{i,i',t}\} \leqslant 0, \forall i, \\
& C_2 : \sum_{t=1}^{T} g_t^{2,i} := \sum_{t=1}^{T} \{\sum_{i'} x_{i,i',t} - r_{i,t}\} \leqslant 0, \forall i, \\
& C_3 : \sum_{t=1}^{T} g_t^{3,i} := \sum_{t=1}^{T} \{\sum_j y_{i,j,t} - |\mathcal{J}| z_{i,t}\} \leqslant 0, \forall i, \\
& C_4 : \sum_{t=1}^{T} g_t^{4,i} := \sum_{t=1}^{T} \{\sum_{i'} x_{i',i,t} - R_i \sum_j y_{i,j,t}\} \leqslant 0, \forall i, \\
& C_5 : h_t^i := \sum_j y_{i,j,t} d_j - \Omega_i \leqslant 0, \forall i, t, \\
\text{var.} \ \ & y_{i,j,t}, z_{i,t} \in \{0,1\}, x_{i,i',t} \in \mathbb{Z}_{\geqslant 0}, \forall t \in \mathcal{T}, \forall i, i' \in \mathcal{I}, \forall j \in \mathcal{J},
\end{aligned}$$

where Constraints C_1 and C_2 capture the dispatching of inference queries, derived from Constraint (a); C_3 ensures that DNN models are only deployed on active GPU servers; C_4 ensures that inference queries are only dispatched to the GPU servers with deployed DNN models; C_5 respects the model hosting capacity of each edge server at each time slot. This problem highlights the integration of communication and AI workloads, aligning with the fundamental vision of future 6G networks [31].

Simplified representation: To solve this problem, we simplify the representation as follows:

$$\min \quad \sum_{t=1}^{T} f_t(\mathbf{M}_t), \quad \text{s.t.} \sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t) \preceq \mathbf{0}, \mathbf{h}(\mathbf{M}_t) \preceq \mathbf{0},$$

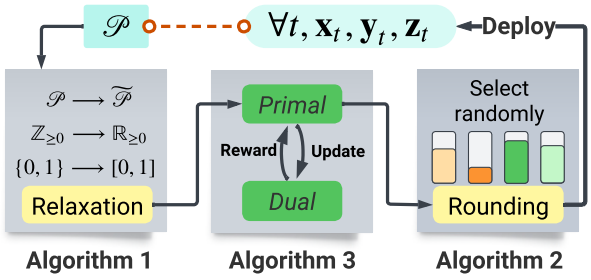


Figure 7 Our proposed *ACE* framework

where bold symbols represent vectors. $\mathbf{M}_t = [\mathbf{x}_t^\intercal, \mathbf{y}_t^\intercal, \mathbf{z}_t^\intercal]^\intercal$ is the aggregation of all decision variables at time slot t and $\forall t : \mathbf{x}_t \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{I}|\times|\mathcal{I}|}, \mathbf{y}_t \in \{0,1\}^{|\mathcal{I}|\times|\mathcal{J}|}, \mathbf{z}_t \in \{0,1\}^{|\mathcal{I}|}$; $f_t(\mathbf{M}_t)$ represents the objective function at time slot t; the long-term constraints $C_1 \sim C_4$ are denoted as follows:
$\mathbf{g}_t(\mathbf{M}_t) = [\underbrace{\ldots, g_t^{1,i}}_{C_1}, \underbrace{\ldots, g_t^{2,i}}_{C_2}, \underbrace{\ldots, g_t^{3,i}}_{C_3}, \underbrace{\ldots, g_t^{4,i}}_{C_4}]^\intercal$; the instantaneous constraint C_5 is $\mathbf{h}(\mathbf{M}_t) = \left[h_t^1, \ldots, h_t^{|\mathcal{I}|}\right]^\intercal$.

Problem challenges: We aim to design a polynomial-time algorithm that ensures the feasible solutions $\boldsymbol{M}$ are close to the optimal solutions $\boldsymbol{M}^*$. Specifically, the time-averaged deviation between the total cost incurred by $\boldsymbol{M}$ and the total cost incurred by $\boldsymbol{M}^*$ diminishes as time goes on. However, designing such an algorithm presents several challenges: (*c1*) *Integer decision* makes each per-slot sub-problem an integer program, typically NP-hard [32], which is difficult to solve optimally in polynomial time, (*c2*) *Posterior parameters* (e.g., $c_{i,j,t}$, $\sigma_{i,i',t}$, $a_{j,t}$, and $r_{i,t}$) are only revealed after the decisions are deployed, hindering us from making the optimal decisions in each time slot.

Theorem 1. Problem $\mathscr{P}$ is NP-hard by reduction from the minimum knapsack problem [33].

Proof. We prove that problem $\mathscr{P}$ is NP-hard via a reduction from the minimum knapsack problem [33]. Consider an instance of the minimum knapsack problem with n items, where each item i has cost c_i, capacity p_i, and the goal is to select a subset of minimum total cost whose total capacity is at least P.

We construct a special instance of Problem $\mathscr{P}$ as follows. We consider one time slot ($T = 1$) and $n+1$ edge servers indexed by $\{0, 1, \ldots, n\}$. Server 0 acts as a pure source server, while servers $\{1, \ldots, n\}$ correspond to the n items. We consider a single DNN model, i.e., $J = 1$, with model demand $d_1 = 1$. For server 0, we set $r_0 = P, R_0 = 0, \Omega_0 = 0, e_0 = 0$. For server $i \in \{1, \ldots, n\}$, we set $r_i = 0, R_i = p_i, \Omega_i = 1, e_i = c_i$. All other cost coefficients are set to zero, i.e., $\Gamma_{i,1} = 0, a_1 = 0, \sigma_{i,i'} = 0, \forall i, i'$.

Under this construction, the objective reduces to $\min \sum_{i=1}^n c_i z_i$, since all other terms are zero. Next, constraints C_1 and C_2 together imply that all P queries generated at server 0 must be dispatched, i.e., $P = \sum_{i' \in \{1,\ldots,n\}} x_{0,i'}$. Constraint C_3 enforces that a model can only be deployed on an activated server, i.e., $y_{i,1} \leqslant z_i$. Constraint C_4 ensures that queries can only be assigned to servers on which the model is deployed, and server i can process at most $R_i y_{i,1} = p_i y_{i,1} \geqslant x_{0,i}$ queries. Constraint C_5 is automatically satisfied, since $d_1 = 1$, $\Omega_i = 1$ for all $i \geqslant 1$, and $\Omega_0 = 0$, which forces $y_{0,1} = 0$. Then, there exists a feasible dispatch vector $\mathbf{x}$ satisfying C_1, C_2, and C_4, if and only if $\sum_{i=1}^n p_i y_{i,1} \geqslant P$. Furthermore, in any optimal solution, we must have $z_i = y_{i,1}$ for every $i \in \{1, \ldots, n\}$. This is because $y_{i,1} \leqslant z_i$, and assuming $z_i = 1$ but $y_{i,1} = 0$, then resetting z_i to 0 preserves feasibility while strictly reducing the objective value. Therefore, the objective can be equivalently written as $\min \sum_{i=1}^n c_i y_{i,1}$. Then, the reduced problem becomes $\min \sum_{i=1}^n c_i y_{i,1}$ s.t. $\sum_{i=1}^n p_i y_{i,1} \geqslant P$, $y_{i,1} \in \{0,1\}$, $\forall i$, which is exactly the minimum knapsack problem. Therefore, our problem $\mathscr{P}$ is NP-hard.

3 Online algorithm design

To address the challenges above, we design an interference-aware GPU scheduling algorithm called Allocation for Computational Edge (*ACE*), as illustrated in Figure 7. Specifically, we handle the NP-hardness by (*s1*) relaxing the problem $\mathscr{P}$ from the integral domain to the real domain and rounding the real solution to the integral solution without violating the instantaneous constraints (i.e., Algorithms 1 and 2 in Figure 7). Then, we design an online learning method to (*s2*) continuously capture the dynamic trends of posterior parameters in Algorithm 3. Solutions (*s1*) and (*s2*) are for challenges (*c1*) and (*c2*), respectively.

Algorithm 1 ACE algorithm

Require: Initialize $\widetilde{\mathbf{M}}_0$ and $\widetilde{\mathbf{M}}_1$.

1: *Relax* problem $\mathscr{P}$ to problem $\widetilde{\mathscr{P}}$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Deploy integral solution $\mathbf{M}_t$
$\trianglerighteq$ Schedule models and dispatch inference queries
4: Invoke **Algorithm 3** to obtain $\widetilde{\mathbf{M}}_{t+1}$
5: *Round* $\widetilde{\mathbf{M}}_{t+1}$ to obtain solution $\mathbf{M}_{t+1}$ via **Algorithm 2**
6: **end for**

Algorithm 2 Randomized rounding component

Require: The fractional solutions $\left[\widetilde{\mathbf{x}}_t^\intercal, \widetilde{\mathbf{y}}_t^\intercal, \widetilde{\mathbf{z}}_t^\intercal\right]^\intercal$;

Ensure: The integral solutions $\left[\mathbf{x}_t^\intercal, \mathbf{y}_t^\intercal, \mathbf{z}_t^\intercal\right]^\intercal$.

// Rounding $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{z}}_t$
1: Round $\tilde{\mathbf{x}}_t^\intercal$, $\tilde{\mathbf{z}}_t^\intercal$ to obtain $\mathbf{x}_t^\intercal, \mathbf{z}_t^\intercal$ based on equation (3)
// Rounding $\tilde{\mathbf{y}}_t$
2: **for** each edge server $i \in \mathcal{I}$ **do**
3: Define $\mathbf{B}_{i,t} = \left[\tilde{y}_{i,1,t} d_1, \ldots, \tilde{y}_{i,|\mathcal{J}|,t} d_{|\mathcal{J}|}\right]^\intercal$, $\upsilon = \mathbf{1}^\intercal \mathbf{B}_{i,t}$, $\pi_1 = \frac{\lfloor \upsilon \rfloor}{\upsilon}, \pi_2 = \frac{\lceil \upsilon \rceil}{\upsilon}$
4: $\mathbf{y}'_{i,t} = \begin{cases} \pi_1 \tilde{\mathbf{y}}_{i,t} = \left[\pi_1 \tilde{y}_{i,1,t}, \ldots, \pi_1 \tilde{y}_{i,|\mathcal{J}|,t}\right]^\intercal, \text{probability}=\lceil \upsilon \rceil - \upsilon \\ \pi_2 \tilde{\mathbf{y}}_{i,t} = \left[\pi_2 \tilde{y}_{i,1,t}, \ldots, \pi_2 \tilde{y}_{i,|\mathcal{J}|,t}\right]^\intercal, \text{probability}=\upsilon - \lfloor \upsilon \rfloor \end{cases}$
5: Define $\mathcal{J}' = \mathcal{J} \backslash \{j | y'_{i,j,t} \in \{0,1\} \ \ \& \ \ y'_{i,j,t} \in \mathbf{y}'_{i,t}\}$
6: **while** $\mathcal{J}' \neq \varnothing$ **do**
7: Select $j_1, j_2 \in \mathcal{J}', j_1 \neq j_2$, and define $\eta = \frac{d_{j_1}}{d_{j_2}}$
8: Take $\theta_1 = \min\{1 - y'_{i,j_1,t}, \frac{1}{\eta} y'_{i,j_2,t}\}$
9: Take $\theta_2 = \min\{\frac{1}{\eta}(1 - y'_{i,j_2,t}), \eta y'_{i,j_1,t}\}$
10: Calculate $(y''_{i,j_1,t}, y''_{i,j_2,t})$ according to equation (5)
11: **if** $y''_{i,j_1,t} \in \{0,1\}$ **then**
12: $y_{i,j_1,t} = y''_{i,j_1,t}$, $\mathcal{J}' = \mathcal{J}' \backslash \{j_1\}$
13: **else**
14: $y'_{i,j_1,t} = y''_{i,j_1,t}$
15: **end if**
16: **if** $y''_{i,j_2,t} \in \{0,1\}$ **then**
17: $y_{i,j_2,t} = y''_{i,j_2,t}$, $\mathcal{J}' = \mathcal{J}' \backslash \{j_2\}$
18: **else**
19: $y'_{i,j_2,t} = y''_{i,j_2,t}$
20: **end if**
21: **end while**
22: **end for**
23: **return** integral solutions $\left[\mathbf{x}_t^\intercal, \mathbf{y}_t^\intercal, \mathbf{z}_t^\intercal\right]^\intercal$

3.1 Relaxation-rounding mechanism

To address the NP-hardness in (c1), we propose a relaxation-rounding mechanism, which has been widely studied in the literature [34,35]. The approach involves optimally solving the problem over the real domain and then rounding the fractional solution to the integral solution for deployment in the edge AI system.

Relaxation: As shown in line 1 of Algorithm 1, we first relax the original problem $\mathscr{P}$ over integral domain to the following problem $\widetilde{\mathscr{P}}$ over the real domain:

$$\min \textstyle\sum_t f_t(\widetilde{\mathbf{M}}_t), \text{s.t.} \sum_t \mathbf{g}_t(\widetilde{\mathbf{M}}_t) \preceq \mathbf{0}, \mathbf{h}(\widetilde{\mathbf{M}}_t) \preceq \mathbf{0}, \widetilde{\mathbf{M}}_t \in \widetilde{\mathcal{M}},$$

where $\widetilde{\mathbf{M}}_t = [\widetilde{\mathbf{x}}_t^\intercal, \widetilde{\mathbf{y}}_t^\intercal, \widetilde{\mathbf{z}}_t^\intercal]^\intercal$ is the fractional version of $\mathbf{M}_t$; $\widetilde{\mathcal{M}}$ represents the real domain corresponding to $\mathcal{M}$, defined as $\widetilde{\mathcal{M}} = \{[\widetilde{\mathbf{x}}_t^\intercal, \widetilde{\mathbf{y}}_t^\intercal, \widetilde{\mathbf{z}}_t^\intercal]^\intercal | \widetilde{\mathbf{x}}_t \in \mathbb{R}_{\geqslant 0}^{|\mathcal{I}| \times |\mathcal{I}|}, \widetilde{\mathbf{y}}_t \in [0,1]^{|\mathcal{I}| \times |\mathcal{J}|}, \widetilde{\mathbf{z}}_t \in [0,1]^{|\mathcal{I}|}\}$.

As shown in line 3 of Algorithm 1, we first deploy the decisions calculated from the last time slot. Based on such decisions, the service providers schedule and dispatch inference. After that, lines 4∼5 make decisions for the next time slot. Specifically, line 4 solves problem $\widetilde{\mathscr{P}}$, which will be elaborated in detail in Subsection 3.2. Line 5 invokes the rounding algorithm to convert these fractions to integers. It should be designed carefully since (i) the rounded integral solutions should respect the instantaneous constraints; (ii) the rounded solutions should be theoretically bounded (or in expectation) by the optimal fractional solution, facilitating subsequent theoretical analysis.

Rounding: Towards these directions, we propose our randomized rounding component in Algorithm 2 that ensures (i) the integral solution does not violate the instantaneous constraint $\mathbf{h}(\cdot) \preceq \mathbf{0}$ in problem $\widetilde{\mathscr{P}}$; (ii) the expectation of the integers equals the optimal fractional solutions. According to line 1, the decision variables i.e., $[\mathbf{x}_t^\intercal, \mathbf{z}_t^\intercal]^\intercal$, which do not appear in constraint $\mathbf{h}(\cdot) \preceq \mathbf{0}$, are rounded as follows:

$$\forall \tilde{b} \in [\mathbf{x}_t^\intercal, \mathbf{z}_t^\intercal]^\intercal, \quad \bar{b} = \begin{cases} \lceil \tilde{b} \rceil, \text{ probability}=\tilde{b} - \lfloor \tilde{b} \rfloor, \\ \lfloor \tilde{b} \rfloor, \text{ probability}=\lceil \tilde{b} \rceil - \tilde{b}. \end{cases} \tag{3}$$

Obviously, this ensures that the expectations of $[\mathbf{x}_t^\intercal, \mathbf{z}_t^\intercal]^\intercal$ equal their fractional solutions:

$$E[\bar{b}] = E[\lceil \tilde{b} \rceil * (\tilde{b} - \lfloor \tilde{b} \rfloor) + \lfloor \tilde{b} \rfloor * (\lceil \tilde{b} \rceil - \tilde{b})] = \tilde{b}. \tag{4}$$

Next, the remaining decision $\widetilde{\mathbf{y}}_t$ is rounded in two steps corresponding to lines 3∼5 and lines 6∼21, respectively. For each edge GPU server i, we first calculate the total allocated resources v, as shown in line 3, by substituting the decisions $\widetilde{\mathbf{y}}_t$ to C_5. Line 4 ensures that the total resources allocated to all models at that edge server are integers by rounding up or down the sum v. Line 5 identifies the fractional decisions from $\mathbf{y}'_{i,t}$, forming the set $\mathcal{J}'$ for the subsequent pairwise rounding procedure in lines 6∼21. Specifically, in line 7, two fractional decisions (indexed by j_1 and j_2) are selected, and they are rounded into integers following the steps below:

$$(y''_{i,j_1,t}, y''_{i,j_2,t}) = \begin{cases} (y'_{i,j_1,t} + \theta_1, y'_{i,j_2,t} - \eta\theta_1), \text{probability}=\dfrac{\theta_2}{\theta_1+\theta_2}, \\ (y'_{i,j_1,t} - \theta_2, y'_{i,j_2,t} + \eta\theta_2), \text{probability}=\dfrac{\theta_1}{\theta_1+\theta_2}, \end{cases} \tag{5}$$

where η (defined in line 7) represents the conversion factor for the two selected decisions, and this step ensures that the total resources allocated to the two inference queries remain unchanged after rounding. In lines 11∼20, if the solutions after rounding are integers, their indices are removed from the set $\mathcal{J}'$. The iteration in lines 6∼21 continues until the set $\mathcal{J}'$ is empty. The expectation of $y_{i,j,t} \in \mathbf{y}_t$ after rounding can be calculated by taking expectation twice:

$$\begin{aligned} &E[E[y_{i,j,t}]] = E[(y'_{i,j,t}+\theta_1)\frac{\theta_2}{\theta_1+\theta_2} + (y'_{i,j,t}-\theta_2)\frac{\theta_1}{\theta_1+\theta_2}] = E[y'_{i,j,t}] = \pi_1 \tilde{y}_{i,j,t}(\lceil \upsilon \rceil - \upsilon) + \pi_2 \tilde{y}_{i,j,t}(\upsilon - \lfloor \upsilon \rfloor) \\ &= (\lfloor \upsilon \rfloor / \upsilon)\tilde{y}_{i,j,t}(\lceil \upsilon \rceil - \upsilon) + (\lceil \upsilon \rceil / \upsilon)\tilde{y}_{i,j,t}(\upsilon - \lfloor \upsilon \rfloor) = \tilde{y}_{i,j,t}. \end{aligned} \tag{6}$$

Core insight: This mechanism addresses the NP-hardness of the original formulation through a relaxation-rounding strategy. The key idea is to first relax the integer decision variables to a continuous feasible set, thereby converting the original combinatorial integer program into a tractable (linear/convex) program that can be solved efficiently at each time slot. Then Algorithm 2 is applied to round the fractional optimum into the integral solution, while (i) preserving feasibility with respect to instantaneous constraints and (ii) providing a bounded gap between the rounded solution and the fractional optimum. Overall, this mechanism runs in polynomial time.

3.2 Primal-dual mechanism

To overcome the challenge (c2), described in Subsection 2.3, we propose an online learning algorithm based on the primal-dual mechanism [36] to solve the relaxed problem $\widetilde{\mathscr{P}}$ as follows:

Prior work [37] has proposed the equivalent min-max form for the problem with the format of

$$\min_{\widetilde{\mathbf{M}}_t} \max_{\lambda_t} \textstyle\sum_t (f_t(\widetilde{\mathbf{M}}_t) + \lambda_t^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}_t)), \quad \text{s.t. } \mathbf{h}(\widetilde{\mathbf{M}}_t) \preceq \mathbf{0}, \quad \widetilde{\mathbf{M}}_t \in \widetilde{\mathcal{M}},$$

where $\lambda_t \in \mathbb{R}_{\geqslant 0}^{\dim(\mathbf{g}_t(\widetilde{\mathbf{M}}_t))}$ are the Lagrange multipliers [38]. This min-max formulation retains only the instantaneous constraints, while embedding all long-term constraints $\mathbf{g}_t$ into the optimization objective. The objective function at each time slot is then given by

$$f_t(\widetilde{\mathbf{M}}) + \lambda^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}), \tag{7}$$

Algorithm 3 Online learning component

Require: For any t; set $\lambda_1 = 0$; set parameters τ, μ; initialize $\Gamma_{i,j,1}$, $i \in \mathcal{I}, j \in \mathcal{J}$;

Ensure: Fractional solution $\widetilde{\mathbf{M}}_{t+1} = \left[\widetilde{\mathbf{x}}_{t+1}^{\intercal}, \widetilde{\mathbf{y}}_{t+1}^{\intercal}, \widetilde{\mathbf{z}}_{t+1}^{\intercal}\right]^{\intercal}$.

1: Observe the total system cost $f_t(\mathbf{M}_t)$ and $\mathbf{g}_t(\mathbf{M}_t)$
2: Update λ_{t+1} based on the dual problem (8)
3: Obtain $\widetilde{\mathbf{M}}_{t+1}$ by solving the primal problem (9)
4: **return** fractional solution $\widetilde{\mathbf{M}}_{t+1} = \left[\widetilde{\mathbf{x}}_{t+1}^{\intercal}, \widetilde{\mathbf{y}}_{t+1}^{\intercal}, \widetilde{\mathbf{z}}_{t+1}^{\intercal}\right]^{\intercal}$

where $\lambda^{\intercal}$ represents the weight of the long-term constraints, which is updated based on the historical cumulative violations of these constraints. To incorporate this, we calculate the gradient of λ, observe its change trend, and adjust it accordingly by adding the cumulative degree of violation. Thus, the *dual* update is expressed as follows:

$$\lambda_{t+1} = [\lambda_t + \mu \nabla_\lambda (f_t(\widetilde{\mathbf{M}}) + \lambda_t^{\intercal} \mathbf{g}_t(\widetilde{\mathbf{M}}))]^+ = [\lambda_t + \mu \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+, \tag{8}$$

where $\nabla(\cdot)$ represents the gradient function, and μ is a step size that balances the weight between historical violations and the current violation of long-term constraints. After obtaining λ_{t+1} from the dual update, the next time slot's decision $\widetilde{\mathbf{M}}_{t+1}$ is determined by solving the *primal* problem:

$$\min_{\widetilde{\mathbf{M}} \in \widetilde{\mathcal{M}}} J_t(\widetilde{\mathbf{M}}), \quad \text{s.t. } \mathbf{h}(\widetilde{\mathbf{M}}) \preceq \mathbf{0}, \tag{9}$$

where the objective function is $J_t(\widetilde{\mathbf{M}}) = \nabla f_t(\widetilde{\mathbf{M}}_t)^{\intercal}(\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t) + \lambda_{t+1}^{\intercal} \mathbf{g}_t(\widetilde{\mathbf{M}}) + \frac{||\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t||^2}{2\tau}$. The above problem is solved using the updated Lagrange multiplier λ_{t+1}. The first two terms (i.e., $\nabla f_t(\widetilde{\mathbf{M}}_t)^{\intercal}(\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t) + \lambda_{t+1}^{\intercal} \mathbf{g}_t(\widetilde{\mathbf{M}})$) represent the optimization objective, while $\frac{||\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t||^2}{2\tau}$ acts as a regularization term to ensure that consecutive decisions remain smooth and bounded. Therefore, this proposed online learning algorithm operates by alternately updating the dual expression (8) and solving the primal problem (9).

Core insight: This mechanism adopts an alternate update to handle both posterior variables (revealed only after decisions are executed) and long-term constraints. Specifically, the dual variables act as adaptive "prices" that accumulate historical constraint violations and translate long-term constraints into per-slot penalty terms. This transforms the originally time-coupled problem into a sequence of tractable per-slot subproblems, while enabling the algorithm to learn from realized feedback over time.

Delayed error-rate feedback: To algorithmically support delayed error-rate feedback, *ACE* can be extended by maintaining a pending-feedback queue. Let d_s denote the delay of the error-rate feedback generated at slot s, and define $\mathcal{A}_t = \{s \mid s + d_s = t\}$, which represents the set of historical slots whose error-rate feedback becomes available at slot t. The primal update can then incorporate both the available feedback and the delayed feedback:

$$\widetilde{\mathbf{M}}_{t+1} = \arg\min_{\widetilde{\mathbf{M}}} (\nabla f_t^{\text{imm}} + \textstyle\sum_{s \in \mathcal{A}_t} \nabla f_s^{\text{err}})^{\top} (\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t) + \lambda_{t+1}^{\top} g_t(\widetilde{\mathbf{M}}) + \frac{\|\widetilde{\mathbf{M}} - \widetilde{\mathbf{M}}_t\|^2}{2\tau}. \tag{10}$$

Here, ∇f_t^{imm} denotes the gradient of the immediately available part of the objective function, and ∇f_s^{err} denotes the delayed gradient associated with the error-rate term from slot s. If only one delayed feedback item, e.g., the feedback from slot s, is received at slot t, then the summation term reduces to $\nabla f_{t-d_s}^{\text{err}}$. Since the long-term constraints do not depend on the inference error rate, the dual update remains unchanged like equation (8).

3.3 Discussion on cold starts and rapidly-changing inputs

On cold starts: In online convex optimization (OCO) where our work falls, a *cold start* often refers to initializing the decision variable arbitrarily (e.g., at the origin or a random point) with zero prior knowledge of the dynamic inputs; conversely, a *warm start* involves using historical data, offline pre-training, or prior domain knowledge to set the decision variable closer to the expected feasible region before the online process begins. We highlight that our work has rigorously proven the sublinear regret bound, i.e., the time-averaged regret tends to zero as the number of time slots goes to infinity. The cold-start penalty is simply an additive constant which does not multiply but is provably absorbed over time. That is, the algorithm does not "fall behind" permanently, i.e., it just starts with a fixed overhead, and the finite loss incurred during the cold-start phase becomes negligible in the long run.

On rapidly changing inputs: Our algorithm is fine under rapid input (e.g., model populations) changes, precisely because the benchmark (i.e., the comparator class) moves in tandem with the environment. If the environment changes rapidly, any reasonable decision-maker, even one with full hindsight, would also incur high costs,

because the optimal decisions themselves are shifting. The regret framework captures this fairly: it charges the algorithm's loss against the cumulative loss of that shifting optimal sequence. In *static regret*, the benchmark is the single best decision chosen after seeing all inputs over the entire time horizon. In contrast, in *dynamic regret*, as adopted in our work, the benchmark is a sequence of time-varying decisions, typically chosen as the per-time-slot minimizers, i.e., when the inputs change rapidly, such a comparator sequence changes rapidly, too. The algorithm is evaluated on how well it tracks this moving target, not on whether it converges to a fixed one. Dynamic regret analysis guarantees that the algorithm's cumulative cost does not diverge significantly from this omniscient moving benchmark. Rapid changes are not a loophole that breaks the algorithm; they are simply a higher trajectory for both the algorithm and the comparator to follow. The regret measures the gap between them along this trajectory. The existence of a sublinear dynamic regret bound, even without quantifying the speed of change, proves that the algorithm's tracking error is structurally bounded regardless of how erratically the inputs behave.

4 Performance analysis

Our aim is to prove the *time-averaged gap* between the total cost incurred by the decisions $\mathbf{M}_t$ of our *ACE* algorithm and the total cost incurred by the offline optimal decisions $\mathbf{M}_t^*$, gradually diminishes as time elapses, formulated by

$$\lim_{T\to\infty} \frac{E[\sum_{t=1}^{T} f_t(\mathbf{M}_t)] - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*)}{T} = 0, \tag{11}$$

which ensures the online decisions of *ACE* gradually approach the offline optimal solutions. We also hope to prove that the *time-averaged violation* of long-term constraints disappears over time, denoted as

$$\lim_{T\to\infty} \frac{\|[E[\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t)]]^+\|}{T} = 0, \tag{12}$$

which ensures the long-term constraints are met over time.

Regret and fit: To achieve the theoretical goal above, we introduce two mathematical metrics, called *regret* and *fit*. We distinguish between two types of regret (Reg_T and $\widetilde{Reg}_T$) over the real and integral domains, respectively:

$$Reg_T := E[\textstyle\sum_{t=1}^{T} f_t(\mathbf{M}_t)] - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*), \mathbf{M}_t^* \in \arg\min_{\mathbf{M}\in\mathcal{M}} f_t(\mathbf{M}), \text{ s.t. } \mathbf{g}_t(\mathbf{M}_t^*) \preceq \mathbf{0}, \mathbf{h}(\mathbf{M}_t^*) \preceq \mathbf{0},$$

$$\widetilde{Reg}_T := \textstyle\sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t) - \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t^*), \widetilde{\mathbf{M}}_t^* \in \arg\min_{\widetilde{\mathbf{M}}\in\widetilde{\mathcal{M}}} f_t(\widetilde{\mathbf{M}}), \text{ s.t. } \mathbf{g}_t(\widetilde{\mathbf{M}}_t^*) \preceq \mathbf{0}, \mathbf{h}(\widetilde{\mathbf{M}}_t^*) \preceq \mathbf{0},$$

where $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ represent the integral and real domains, respectively; $\mathbf{M}_t$ and $\widetilde{\mathbf{M}}_t$ represent the integral solution from line 5 of Algorithm 1 and the fractional solution from line 4, respectively; $\mathbf{M}_t^*$ and $\widetilde{\mathbf{M}}_t^*$ represent the offline optimal solutions in the integral and real domains, respectively.

Similarly, two different fits are proposed over the real and integral domains, respectively:

$$Fit_T := \|[E[\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t)]]^+\|, \mathbf{M}_t \in \mathcal{M}, \tag{13}$$

$$\widetilde{Fit}_T := \|[\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+\|, \widetilde{\mathbf{M}}_t \in \widetilde{\mathcal{M}}. \tag{14}$$

Theoretical results: We demonstrate that both the fit and regret exhibit sublinear growth over time, as shown in Theorem 2 and Theorem 3, which further derive equations (11) and (12), as shown in Corollary 1. Moreover, we prove that our algorithms run in polynomial time in Theorem 4.

Theorem 2. The fit exhibits sublinear growth as follows:

$$Fit_T = \|[E[\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t)]]^+\| \leqslant O(T^{\frac{1}{\epsilon_1}}),$$

where $\epsilon_1 > 1$ is a constant.

Proof. See Section 4.1, via Lemmas 1, 2, 3, and 5. The proof follows the chain below:

$$Fit_T \overset{\text{Lemma 1}}{\leqslant} \widetilde{Fit}_T \overset{\text{Lemma 2}}{\leqslant} \tfrac{\|\lambda_{T+1}\|}{\mu} \overset{\text{Lemma 3}}{\leqslant} U^1(\tau,\mu) \overset{\text{Lemma 5}}{\leqslant} O(T^{\frac{1}{\epsilon_1}}). \tag{15}$$

Specifically, we first relate the integral-domain fit to the real-domain fit via Lemma 1, showing that $Fit_T \leqslant \widetilde{Fit}_T$. We then upper-bound the real-domain Fit by the norm of the dual variable $\|\lambda_{T+1}\|/\mu$ through Lemma 2. Then, Lemma 3 further bounds $\|\lambda_{T+1}\|/\mu$ by a function $U^1(\tau,\mu)$. Finally, Lemma 5 shows that $U^1(\tau,\mu)$ grows sublinearly with T under suitable choices of τ and μ. Combining these steps yields the claimed sublinear bound for Fit_T.

Theorem 3. The regret bound exhibits sublinear growth:

$$Reg_T = E[\sum_{t=1}^{T} f_t(\mathbf{M}_t)] - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*) \leqslant O(T^{\frac{1}{\epsilon_2}}),$$

where $\epsilon_2 > 1$ is a constant.

Proof. See Section 4.2, via Lemmas 1, 4, 5 and Proposition 1. The proof follows the chain below:

$$Reg_T \overset{\text{Lemma 1}}{\leqslant} \widetilde{Reg}_T = \sum_{t=1}^{T} \widetilde{reg}_t \overset{\text{Proposition 1}}{\leqslant} \sum_{t=1}^{T}(U^3(\tau,\mu)) \overset{\text{Lemma 4}}{\leqslant} U^2(\tau,\mu) \overset{\text{Lemma 5}}{\leqslant} O(T^{\frac{1}{\epsilon_2}}). \tag{16}$$

We first relate the integral-domain regret to the real-domain regret via Lemma 1. Then, Proposition 1 upper-bounds the one-slot real-domain regret by $U^3(\tau,\mu)$. By summing these per-slot bounds over the entire horizon, Lemma 4 further derives an overall upper bound of the form $U^2(\tau,\mu)$. Finally, Lemma 5 shows that this bound grows sublinearly with time, given appropriate τ and μ. Therefore, Reg_T also grows sublinearly over time.

Corollary 1. We prove equations (11) and (12) via Theorems 3 and 2, respectively.

Proof. For Theorem 2, we take $T \to \infty$ and have equations as follows:

$$\lim_{T\to\infty} \frac{Fit_T}{T} = \lim_{T\to\infty} \frac{\|[E[\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t)]]^+\|}{T} \leqslant \lim_{T\to\infty} \frac{O(T^{\frac{1}{\epsilon_1}})}{T} = \lim_{T\to\infty} O(T^{\frac{1-\epsilon_1}{\epsilon_1}}) \overset{(17a)}{=} 0, \tag{17}$$

where equation (17a) holds since $\epsilon_1 > 1$ and $\frac{1-\epsilon_1}{\epsilon_1} \leqslant 0$. Then, we take $T \to \infty$ for Theorem 3 and also have equations:

$$\lim_{T\to\infty} \frac{Reg_T}{T} = \lim_{T\to\infty} \frac{E[\sum_{t=1}^{T} f_t(\mathbf{M}_t)] - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*)}{T} \leqslant \lim_{T\to\infty} \frac{O(T^{\frac{1}{\epsilon_2}})}{T} = \lim_{T\to\infty} O(T^{\frac{1-\epsilon_2}{\epsilon_2}}) \overset{(18a)}{=} 0, \tag{18}$$

where equation (18a) holds since $\epsilon_2 > 1$ and $\frac{1-\epsilon_2}{\epsilon_2} \leqslant 0$.

Theorem 4. Algorithm 1, including Algorithm 2 and Algorithm 3, shows polynomial-time complexity.

Proof. Algorithm 2 has time complexity $\mathcal{O}(|\mathcal{J}| * |\mathcal{I}|)$. Obviously, the outer loop of this algorithm iterates $|\mathcal{I}|$ times; the inner loop has at most $|\mathcal{J}|$ iterations. This is because each selection converts a fractional variable into either 0 or 1, being removed from the candidate set with a maximum of $|\mathcal{J}|$.

Algorithm 3 also operates in polynomial time. Its time complexity is determined by the convex optimization problem solved in line 3. According to works [9, 39], this step's time complexity is $O(N_d^2 N_c^{2.5} + N_c^{3.5})$, where N_d denotes the number of constraints and N_c denotes the number of decision variables.

Therefore, Algorithm 1 has polynomial-time complexity with $\mathcal{O}(|\mathcal{J}| * |\mathcal{I}|) + O(N_d^2 N_c^{2.5} + N_c^{3.5})$ per time slot.

Physical meaning of theorems: *Regret* characterizes the performance gap incurred by an online algorithm compared to the offline optimum. In our formulation, the objective is a (weighted) sum of multiple terms, including inference latency, inference error rate, communication cost, and energy consumption. Thus, the performance captures all these terms. Similarly, latency, communication cost, and energy consumption are also reflected in the regret through their corresponding terms in the objective. Furthermore, *sublinear regret* implies that the time-averaged performance gap of *ACE* (i.e., our proposed online approach) relative to the series of the one-slot offline optimums diminishes as time goes on, implying that *ACE* eventually learns to make optimal control decisions in a dynamic edge environment. *Fit* measures the cumulative violation of the long-term constraints under an online algorithm, i.e., C_1-C_4 in our formulation. Any violation of these constraints at an individual time slot is captured by the fit. A large fit indicates that the system frequently violates the basic operational rules, such as leaving some requests undispatched, deploying models on inactive servers, or sending requests to servers without the required model deployment. In contrast, our proved *sublinear fit* implies that the cumulative constraint violation grows more slowly than the elapse of time; equivalently, the time-averaged cumulative violation of C_1-C_4 vanishes as time goes.

4.1 Proof of Theorem 2

Some assumptions are first introduced, which are commonly and widely adopted in similar works [40–42].

- A-1: For all $t \in \mathcal{T}$ and $\widetilde{\mathbf{M}} \in \widetilde{\mathcal{M}}$, $\mathbf{g}_t(\widetilde{\mathbf{M}})$ and the gradient of $f_t(\widetilde{\mathbf{M}})$ are bounded by $\|\mathbf{g}_t(\widetilde{\mathbf{M}})\| \leqslant G$ and $\|\nabla f_t(\widetilde{\mathbf{M}})\| \leqslant F$, where both G and F are constants.

• A-2: There exists a constant $\varepsilon > 0$ and an interior point $\widehat{\mathbf{M}}_t \in \widetilde{\mathcal{M}}$ such that $\mathbf{g}_t(\widehat{\mathbf{M}}_t) \leqslant -\varepsilon\mathbf{1}$; $\varepsilon > \overline{V}(\mathbf{g})$ and we define $\overline{V}(\mathbf{g}) := \max_t \max_{\widetilde{\mathbf{M}}\in\widetilde{\mathcal{M}}} ||[\mathbf{g}_{t+1}(\widetilde{\mathbf{M}}) - \mathbf{g}_t(\widetilde{\mathbf{M}})]^+||$.

• A-3: The radius of real domain $\widetilde{\mathcal{M}}$ can be bounded by R, i.e., $||\widetilde{\boldsymbol{M}}_i - \widetilde{\boldsymbol{M}}_j|| \leqslant R, \forall \widetilde{\boldsymbol{M}}_i, \widetilde{\boldsymbol{M}}_j \in \widetilde{\mathcal{M}}$.

For the $1st$ step of equations (15) and (16), we propose Lemma 1:

Lemma 1. The fit and regret in the integral domain and the real domain have the following relationship:

$$Fit_T \leqslant \widetilde{Fit}_T, Reg_T \leqslant \widetilde{Reg}_T.$$

Proof. For the fit, we have the following equations:

$$Fit_T = ||[E[\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{M}_t)]]^+|| \overset{(19a)}{=} \|[\sum_{t=1}^{T} \mathbf{g}_t(E[\mathbf{M}_t])]^+\| \overset{(19b)}{=} \|[\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+\| \overset{(19c)}{\leqslant} \|[\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+\| = \widetilde{Fit}_T. \tag{19}$$

Equation (19a) holds because the expectation of the sum of variables is equal to the expected sum in probability theory [43,44]; and equation (19b) holds since the expectation of the integral solution equals the fractional solution in equation (4) and equation (6) from our proposed rounding Algorithm 2, and the function is affine; equation (19c) holds since each one provides an upper bound on itself. For the sake of uniformity, we use "$\leqslant$" to describe the relationship between Fit_T and $\widetilde{Fit}_T$, instead of "=".

For regret, we have the following equations:

$$\begin{aligned} &E[\textstyle\sum_{t=1}^{T} f_t(\mathbf{M}_t)] - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*) \overset{(20a)}{=} \sum_{t=1}^{T} f_t(E[\mathbf{M}_t]) - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*) \overset{(20b)}{=} \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t) - \sum_{t=1}^{T} f_t(\mathbf{M}_t^*) \\ &\overset{(20c)}{\leqslant} \textstyle\sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t) - \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t^*) = \widetilde{Reg}_T, \end{aligned} \tag{20}$$

where equation (20a) holds due to the linearity of expectation [43,44], and the function is affine; equation (20b) holds since Algorithm 2 ensures the expectation of the integral solution equals the fractional solution in equations (4) and (6); and equation (20c) holds since $\sum_{t=1}^{T} f_t(\mathbf{M}_t^*) \geqslant \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t^*)$, which means the fractional optimal solution is lower than the integral optimal solution in a minimization problem [45].

For the $2nd$ step of equation (15), we prove $\widetilde{Fit}_T$ is bounded by $\frac{||\lambda_{T+1}||}{\mu}$ in the following Lemma 2.

Lemma 2. The fit in the real domain can be bounded by

$$\widetilde{Fit}_T \leqslant \|\lambda_{T+1}\|/\mu.$$

Proof. According to the dual expression in equation (8) of Algorithm 3, we have the derivation:

$$\lambda_{T+1} = [\lambda_T + \mu \mathbf{g}_T(\widetilde{\mathbf{M}}_T)]^+ \geqslant ... \geqslant \lambda_1 + \textstyle\sum_{t=1}^{T} \mu \mathbf{g}_t(\widetilde{\mathbf{M}}_t).$$

After rearranging the equations above, we have

$$\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t) \leqslant \frac{\lambda_{T+1}}{\mu} - \frac{\lambda_1}{\mu} \overset{(21a)}{=} \frac{\lambda_{T+1}}{\mu}, \tag{21}$$

where (21a) holds since $\lambda_1 = 0$, as shown in the first line of Algorithm 3. Then, we have the equations:

$$\widetilde{Fit}_T = \|[\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+\| \overset{(a)}{\leqslant} \|\textstyle\sum_{t=1}^{T} \mathbf{g}_t(\widetilde{\mathbf{M}}_t)\| \overset{(b)}{\leqslant} \|\frac{\lambda_{T+1}}{\mu}\|,$$

where (a) holds since $\|[k]^+\| \leqslant \|k\|, \forall k \in \mathbb{R}$; (b) holds since we take the norm for equation (21).

For the $3rd$ step of equation (15), we give Lemma 3 as follows:

Lemma 3. The ratio of Lagrange multipliers [38] at time slot T to step size μ can be bounded as follows:

$$\tfrac{\|\lambda_{T+1}\|}{\mu} \leqslant G + \tfrac{2FR + R^2/(2\tau) + (\mu G^2)/2}{(\varepsilon - \overline{V}(\mathbf{g}))\mu} \triangleq U^1(\tau, \mu),$$

where F and G are defined in assumption A-1; $\overline{V}(\mathbf{g})$ and ε are from assumption A-2; R is from A-3.

Proof. According to the update of the Lagrange multipliers in equation (8), we have the equations:

$$\begin{aligned}\|\lambda_{t+1}\|^2 &= ||[\lambda_t + \mu \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+||^2 = ||\lambda_t||^2 + 2\mu\lambda_t^\top \mathbf{g}_t(\widetilde{\mathbf{M}}_t) \\ &+ \mu^2||\mathbf{g}_t(\widetilde{\mathbf{M}}_t)||^2 \overset{(22a)}{\leqslant} ||\lambda_t||^2 + 2\mu\lambda_t^\top \mathbf{g}_t(\widetilde{\mathbf{M}}_t) + \mu^2 G^2, \end{aligned} \tag{22}$$

where (22a) holds since assumption A-1; reorganizing it and taking $t = t + 1$, we have equations:

$$\begin{aligned}&\frac{(\|\lambda_{t+2}\|^2 - ||\lambda_{t+1}||^2)}{2\mu} \leqslant \lambda_{t+1}^\top \mathbf{g}_{t+1}(\widetilde{\mathbf{M}}_{t+1}) + \mu G^2/2, \overset{(23a)}{\leqslant} \lambda_{t+1}^\intercal(\mathbf{g}_{t+1}(\widetilde{\mathbf{M}}_{t+1}) - \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1})) + \frac{\mu G^2}{2} + \lambda_{t+1}^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1}) \\ &\overset{(23b)}{\leqslant} \lambda_{t+1}^\intercal[\mathbf{g}_{t+1}(\widetilde{\mathbf{M}}_{t+1}) - \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1})]^+ + \frac{\mu G^2}{2} + \lambda_{t+1}^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1}) \overset{(23c)}{\leqslant} \overline{V}(\mathbf{g})\|\lambda_{t+1}\| + \frac{\mu G^2}{2} + \lambda_{t+1}^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1}), \end{aligned} \tag{23}$$

where (23a) holds since we add two additional complementary terms; (23b) holds since $k \leqslant [k]^+, \forall k \in \mathbb{R}$; (23c) holds since assumption A-2 and $k \leqslant \|k\|, \forall k \in \mathbb{R}$; then, we give an upper bound for $\mu\lambda_{t+1}^\intercal \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1})$.

Since $\widetilde{\mathbf{M}}_{t+1}$ is an optimal solution to subproblem (9), after plugging the interior point $\widehat{\mathbf{M}}_t$ and $\widetilde{\mathbf{M}}_{t+1}$, we have

$$\begin{aligned}&\nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t) + \lambda_{t+1}^\top \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1}) + \frac{\|\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t\|^2}{2\tau} \overset{(24a)}{\leqslant} \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t) - \lambda_{t+1}^\top \varepsilon \mathbf{1} + \frac{||\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t||^2}{2\tau} \\ &\overset{(24b)}{\leqslant} \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t) - \varepsilon\|\lambda_{t+1}\| + \frac{||\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t||^2}{2\tau}, \end{aligned} \tag{24}$$

where equation (24a) holds due to assumption A-2 and any feasible solution other than the optimal solution, e.g., $\widehat{\mathbf{M}}_t$, cannot achieve the minimum value in the minimization problem; equation (24b) holds due to λ is non-negative; reorganizing equation (24), we have the equations:

$$\begin{aligned}&\lambda_{t+1}^\top \mathbf{g}_t(\widetilde{\mathbf{M}}_{t+1}) \leqslant \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t) - \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t) - \varepsilon\|\lambda_{t+1}\| + \frac{(||\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t||^2 - ||\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t||^2)}{2\tau} \\ &\overset{(25a)}{\leqslant} \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t) - \nabla f_t(\widetilde{\mathbf{M}}_t)^\top(\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t) - \varepsilon\|\lambda_{t+1}\| + \frac{R^2}{2\tau} \\ &\overset{(25b)}{\leqslant} \|\nabla f_t(\widetilde{\mathbf{M}}_t)\|(\|\widehat{\mathbf{M}}_t - \widetilde{\mathbf{M}}_t\| + \|\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t\|) - \varepsilon\|\lambda_{t+1}\| + \frac{R^2}{2\tau} \overset{(25c)}{\leqslant} 2FR - \varepsilon\|\lambda_{t+1}\| + \frac{R^2}{2\tau}, \end{aligned} \tag{25}$$

where equation (25a) holds due to assumption A-3 and $||\widetilde{\mathbf{M}}_{t+1} - \widetilde{\mathbf{M}}_t||^2 \geqslant 0$; equation (25b) is derived through the repeated use of Cauchy-Schwarz inequality [46]; equation (25c) holds since assumptions A-1 and A-3. Combining the above equation (25) with equation (23), we obtain

$$\frac{(\|\lambda_{t+2}\|^2 - ||\lambda_{t+1}||^2)}{2\mu} \leqslant \overline{V}(\mathbf{g})\|\lambda_{t+1}\| + \frac{\mu G^2}{2} + (2FR - \varepsilon\|\lambda_{t+1}\| + \frac{R^2}{2\tau}) \triangleq U^4. \tag{26}$$

If U^4 is assumed to be less than 0, then $(\|\lambda_{t+2}\|^2 - ||\lambda_{t+1}||^2)$ would also be less than 0, which contradicts $\|\lambda_{t+2}\|^2 \geqslant \|\lambda_{t+1}\|^2$ (from equation (8)). Thus, we have $U^4 > 0$. Reorganizing (26), we have

$$\frac{\|\lambda_{T+1}\|}{\mu} \leqslant G + \frac{2FR + R^2/(2\tau) + (\mu G^2)/2}{(\varepsilon - \overline{V}(\mathbf{g}))\mu} \triangleq U^1(\tau, \mu). \tag{27}$$

As for the last step of equation (15), we prove that $U^1(\tau, \mu)$ grows sublinearly over time in Lemma 5.

4.2 Proof of Theorem 3

We have proved the $1st$ step of equation (16) in the previous Lemma 1. Actually, regret represents a concept of long-term and cumulative *one-shot regret* generated in each time slot. For the $2nd$ step of equation (16), we provide an upper bound for such *one-shot regret* in Proposition 1 as follows:

Proposition 1. From previous works [40, 47], the difference between the feasible real solutions and the optimal real solutions can be bounded by $U^3(\tau, \mu)$, represented by:

$$\begin{aligned}f_t(\widetilde{\mathbf{M}}_t) - f_t(\widetilde{\mathbf{M}}_t^*) \leqslant U^3(\tau, \mu) &\triangleq \frac{\tau F^2}{2} + \|\lambda_{t+1}\|V(\mathbf{g}_t) + \mu G^2/2 + \frac{(\|\lambda_{t+1}\|^2 - ||\lambda_{t+2}||^2)}{2\mu} \\ &+ \frac{2R\|\widetilde{\mathbf{M}}_t^* - \widetilde{\mathbf{M}}_{t-1}^*\| + \|\widetilde{\mathbf{M}}_t - \widetilde{\mathbf{M}}_{t-1}^*\|^2 - \|\widetilde{\mathbf{M}}_t^* - \widetilde{\mathbf{M}}_{t+1}\|^2}{2\tau}, \end{aligned}$$

where $V(\mathbf{g}_t) = \max_{\widetilde{\mathbf{M}} \in \widetilde{\mathcal{M}}} ||[\mathbf{g}_{t+1}(\widetilde{\mathbf{M}}_t) - \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+||, \forall t$.

For the $3rd$ step of equation (16), we show the regret can be bounded by $U^2(\tau,\mu)$ as follows:

Lemma 4. The regret in the real domain can be bounded:

$$\widetilde{Reg}_T = \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t) - \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t^*) \leqslant U^2(\tau,\mu),$$

Proof. By summing the equations from all time slots in Proposition 1, we obtain the following equations:

$$\begin{aligned}
&\widetilde{Reg}_T = \textstyle\sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t) - \sum_{t=1}^{T} f_t(\widetilde{\mathbf{M}}_t^*) \leqslant \frac{\tau F^2 T}{2} + ||\lambda_{T+1}|| V(\{\mathbf{g}_t\}_{t=1}^T) \\
&+\frac{\mu G^2 T}{2} + \frac{R\cdot V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)}{\tau} + \frac{||\widetilde{\mathbf{M}}_1 - \widetilde{\mathbf{M}}_0^*||^2}{2\tau} - \sum_{t=1}^{T} \frac{(\|\lambda_{t+1}\|^2 - ||\lambda_{t+2}||^2)}{2\mu} \\
&= \frac{\tau F^2 T}{2} + ||\lambda_{T+1}|| V(\{\mathbf{g}_t\}_{t=1}^T) + \frac{\mu G^2 T}{2} + \frac{R\cdot V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)}{\tau} + \frac{||\widetilde{\mathbf{M}}_1 - \widetilde{\mathbf{M}}_0^*||^2}{2\tau} + \frac{||\lambda_2||^2 - ||\lambda_{T+2}||^2}{2\mu} \\
&\overset{(28a)}{\leqslant} \frac{\tau F^2 T}{2} + \mu U^1(\tau,\mu) V(\{\mathbf{g}_t\}_{t=1}^T) + \frac{\mu G^2 T}{2} + \frac{R\cdot V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)}{\tau} + \frac{||\widetilde{\mathbf{M}}_1 - \widetilde{\mathbf{M}}_0^*||^2}{2\tau} + \frac{||\lambda_2||^2 - ||\lambda_{T+2}||^2}{2\mu} \triangleq U^2(\tau,\mu),
\end{aligned} \tag{28}$$

where (28a) holds since equation (27); $V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)$ and $V(\{\mathbf{g}_t\}_{t=1}^T)$ are defined as follows:

$$V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T) = \sum_{t=1}^{T} \|\widetilde{\mathbf{M}}_t^* - \widetilde{\mathbf{M}}_{t-1}^*\|, \tag{29}$$

$$V(\{\mathbf{g}_t\}_{t=1}^T) = \sum_{t=1}^{T} \max_{\widetilde{\mathbf{M}} \in \widetilde{\mathcal{M}}} ||[\mathbf{g}_{t+1}(\widetilde{\mathbf{M}}_t) - \mathbf{g}_t(\widetilde{\mathbf{M}}_t)]^+||. \tag{30}$$

For the last steps of equations (15) and (16), we have Lemma 5.

Lemma 5. Both of the functions $U^1(\tau,\mu)$ and $U^2(\tau,\mu)$, defined in (27) and (28), grow sublinearly over time, after taking $\tau = \mu = T^{-\frac{1}{3}} * \max\{V(\{\mathbf{g}_t\}_{t=1}^T)^{\frac{1}{3}}, V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)^{\frac{1}{3}}\}$:

$$U^1(\tau,\mu) \leqslant O(T^{\frac{1}{\epsilon_1}}), U^2(\tau,\mu) \leqslant O(T^{\frac{1}{\epsilon_2}}),$$

where $\epsilon_1, \epsilon_2 > 1$ and $V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)$ and $V(\{\mathbf{g}_t\}_{t=1}^T)$ are defined in equations (29) and (30), respectively.

Proof. For simplicity, we define $\Xi \triangleq \max\{V(\{\mathbf{g}_t\}_{t=1}^T)^{\frac{1}{3}}, V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)^{\frac{1}{3}}\}$. We give the proof for fit:

$$Fit_T \leqslant U^1(T^{-\frac{1}{3}} * \Xi, T^{-\frac{1}{3}} * \Xi) = G + \frac{T^{\frac{1}{3}} * \Xi^{-1} * (2FR + T^{\frac{1}{3}} * \Xi^{-1} R^2/2) + G^2/2}{(\varepsilon - \overline{V}(\mathbf{g}))} \overset{(31a)}{=} O(T^{\frac{1}{\epsilon_1}}), \tag{31}$$

where $\epsilon_1 > 1$; equation (31a) holds since all the powers associated with T are less than 1; the notation $\Xi^{-1} = \max\{V(\{\mathbf{g}_t\}_{t=1}^T)^{-\frac{1}{3}}, V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)^{-\frac{1}{3}}\}$. Similarly, we obtain the derivations for regret as follows:

$$\begin{aligned}
&Reg_T \leqslant U^2(T^{-\frac{1}{3}} * \Xi, T^{-\frac{1}{3}} * \Xi) = \frac{F^2 T^{\frac{2}{3}} * \Xi}{2} + \frac{\mu G^2 T^{\frac{2}{3}} * \Xi}{2} \\
&+ T^{\frac{1}{3}} * \Xi^{-1} R V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T) + T^{-\frac{1}{3}} * \Xi G + \frac{(2FR + T^{\frac{1}{3}} * \Xi^{-1} R^2/2) + T^{-\frac{1}{3}} * \Xi G^2/2}{(\varepsilon - \overline{V}(\mathbf{g}))} \\
&+ \frac{T^{\frac{1}{3}} \Xi^{-1} (||\widetilde{\mathbf{M}}_1 - \widetilde{\mathbf{M}}_0^*||^2 + ||\lambda_2||^2 - ||\lambda_{T+2}||^2)}{2} * \max\{V(\{\mathbf{g}_t\}_{t=1}^T)^{\frac{1}{3}}, V(\{\widetilde{\mathbf{M}}_t^*\}_{t=1}^T)^{\frac{1}{3}}\} \overset{(32a)}{=} O(T^{\frac{1}{\epsilon_2}}),
\end{aligned} \tag{32}$$

where $\epsilon_2 > 1$, and equation (32a) holds since all the powers associated with T are less than 1.

5 Experiment

Table 2 shows the inputs for our subsequent evaluations in this section.

5.1 Data and settings

Testbed setup: Figure 4 illustrates our experimental hardware, which includes the GeForce RTX 2080 Ti, GeForce RTX 3090, and NVIDIA Tesla V100. These servers are used to conduct inference measurements for co-located DNN models. We evaluate the incurred inference delay of various DNN models (i.e., ResNet18 to ResNet152) when they are co-located on a single GPU. The results of our measurement are shown in previous Figures 5 and 6, which are treated as the input to our following evaluations. Specifically, $\Gamma_{i,j,t}$ ranges in $0.01 \sim 0.03$ s from our built testbed.

Table 2 Experimental inputs

Inputs	Values
$c_{i,j,t}$	Inference latency in $0.01 \sim 0.03$ s from our testbed
$r_{i,t}$	London user data [24]; each user sends 10∼20 queries [11]
$\sigma_{i,i',t}$	Latency between edge servers in 0.25∼10 ms
d_j	Resource demand of model j in 1∼20 [22]
Ω_i	Resource capacity for server i in 80∼300 [11]
$e_{i,t}$	Operational cost in 0.050∼0.150 kWh [48, 49]
$a_{j,t}$	Error rate of DNN model in $0.1 \sim 0.3$ [13]

Edge environments: We construct the network topology based on the existing topology of the SNDlib project [50]. Aligning with the previous study [11], the query-dispatching communication cost $\sigma_{i,i',t}$ is configured with 0.25∼10 ms. We set the operational cost $e_{i,t}$ of the edge GPU server i to 0.050∼0.150 kWh [48, 49]. Specifically, we calculate it as $e_{i,t} = P_{i,t}^{\text{srv}} \Delta t$, where $\Delta t = 15$ minutes and $P_{i,t}^{\text{srv}} \in [0.2, 0.6]$ kW is calibrated from representative single-GPU server configurations and public NVIDIA GPU specifications [51, 52].

DNN inference workloads: We utilize the London open dataset [24] to determine the number of edge users $r_{i,t}$. Specifically, this dataset [24] records the number of users at London underground stations every 15 minutes over a total of three days, resulting in a total of 300 time slots. Taking into account previous research and datasets [13, 37, 41, 53], we set the time slot interval to 15 minutes with a total of 300 time slots. To align with previous studies [11], each user sends 10-20 inference queries per time slot (i.e., every 15 minutes). This setting also aligns with the existing real-world granularity of the London user data [24], which records the number of passengers (i.e., users in our case) entering London's underground stations at 15-minute intervals. Then, the inference error rate of the DNN model $a_{j,t}$ is configured within 0.1∼0.3 [13]. The resource demand d_j for model j is set to 1∼20 [11], while the resource capacity Ω_i of GPU server i is configured with the range of 80∼300 [11].

Other parameters: Then, the step sizes τ and μ in Algorithm 3, are set to $0.15 \approx \mathcal{O}(300^{-\frac{1}{3}})$ [11].

Algorithms and metrics: Our proposed *ACE* leverages CVXPY [54] to solve subproblems at each time slot. The primary objective is to minimize the *total system cost*. Additionally, we implement the following comparative algorithms:

- *Random*: A baseline that randomly generates decision variables.
- *Greedy*: Greedy strategy for dispatching inference workloads and deploying DNN models.
- *iGniter*: State-of-the-art method [6] that considers GPU interference while dispatching workloads. Specifically, the model deployment decision ($y_{i,j,t}$) is from the iGniter algorithm [6]; other control decisions ($x_{i,i',t}, z_{i,t}$) are based on the "Greedy" strategy.
- *ACE*: Our proposed *ACE* framework integrating Algorithms 1, 2 and 3.

5.2 Experimental results

Figures 8 and 9 show the cumulative cost and real-time cost in the edge AI system, respectively, both of which represent the objective values in our problem $\mathscr{P}$. Figure 8 demonstrates that our proposed *ACE* algorithm exhibits the slowest cost growth, whereas the *Random* algorithm performs the worst. Figure 9 displays the cost at each time slot, revealing fluctuations over time due to the dynamic nature of edge environments, including network bandwidth, inference workloads, and inference error rates. Even so, *ACE* effectively captures these trends and makes adaptive decisions accordingly. Our *ACE* algorithm achieves a 40%∼60% improvement compared to heuristic approaches (*Random* and *Greedy*). Furthermore, when compared to the state-of-the-art algorithm (iGniter [6]), our method reduces the system cost by 10%∼30% (15% on average). Overall, *ACE* improves performance by an average of 40%, demonstrating its effectiveness in dynamic edge environments. Furthermore, we discuss why *ACE* outperforms the baseline methods. *Compared with Random and Greedy*, *ACE* explicitly accounts for co-location interference through the measured/observed interference-incurred latency feedback, whereas *Random* and *Greedy* rely on heuristic placement rules that are blind to interference. As a result, *ACE* can avoid repeatedly selecting high-interference co-location patterns that would otherwise increase latency and overall system cost. *Compared with iGniter*, *ACE* not only considers interference, but also jointly captures multiple edge-system factors, including latency, energy consumption, communication cost, and inference error rate. In other words, *ACE* is designed to seek a better trade-off among these conflicting objectives, rather than optimizing a single aspect only.

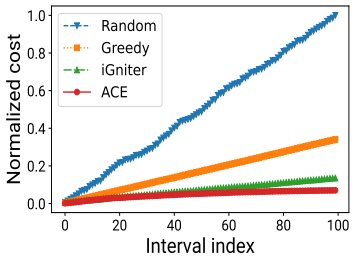


Figure 8 Cumulative cost

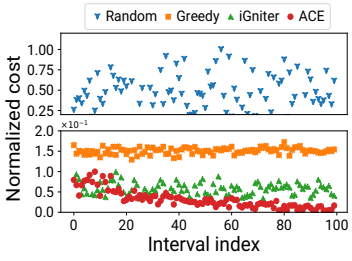


Figure 9 Real-time cost

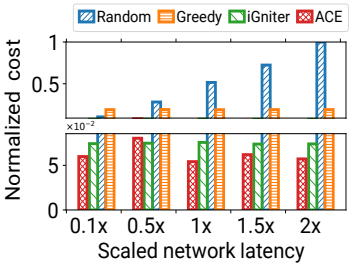


Figure 10 Impact of network status

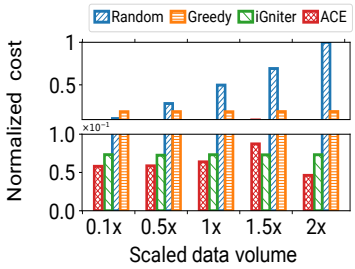


Figure 11 Impact of data volume

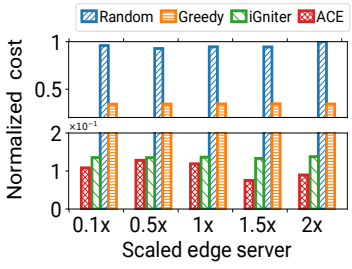


Figure 12 Impact of edge

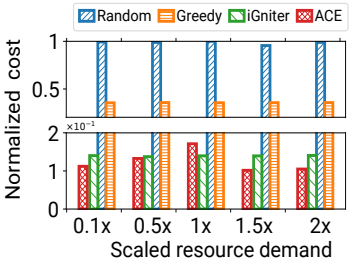


Figure 13 Impact of demand

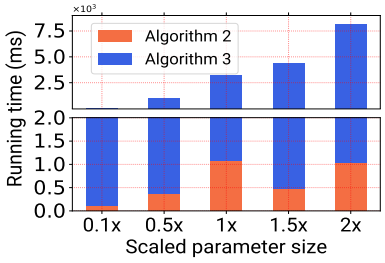


Figure 14 Overhead

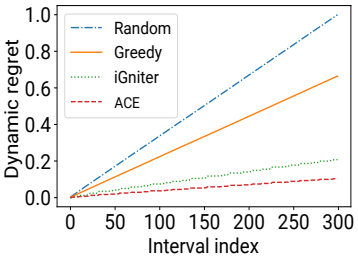


Figure 15 Regret

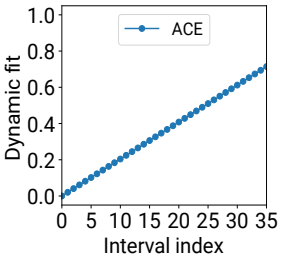


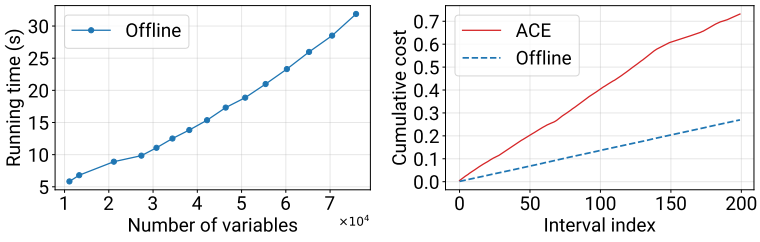
Figure 16 Fit

Figures 10∼13 illustrate the scalability of the algorithms under different input scales. In these figures, "1x" represents the default settings described in Section 5.1, while the others indicate scaled-up or scaled-down versions of these settings accordingly. Figure 10 exhibits the system cost for all algorithms under different network statuses. Our *ACE* achieves the lowest system cost, as it selects edge servers using the relaxation-rounding mechanism. Specifically, *ACE* leverages fractional optimal solutions as probabilities, ensuring that the final feasible decisions closely approximate the optimal ones. Compared to *Random*, *Greedy*, and *iGniter*, our *ACE* algorithm achieves at least 70%, 60%, and 15% improvement, respectively. Figure 11 investigates the performance of all four algorithms under different data volumes. In general, as the data volume increases, the system cost also rises. Our *ACE* algorithm employs the online learning algorithm to "learn" from the observed inference delay for the next time slot based on the decisions' feedback. This enables *ACE* to consistently maintain a lower cost, achieving a 52% improvement compared to the other algorithms. Figure 12 depicts the effect of edge server scalability. The performance of *ACE* becomes pronounced as the number of edge servers increases. This is because a larger number of servers provides more flexibility in deploying models and dispatching inference queries. As a result, our *ACE* outperforms the algorithm [6] and reduces the system cost by 20% on average. To some extent, this also highlights the robustness of *ACE*. Figure 13 visualizes the impact of resource demands on system cost for different algorithms. On the x-axis, "1x"→"2x" represents increasing model resource demands, whereas "1x"→"0.1x" represents decreasing demands. Across all scenarios, our *ACE* consistently selects suitable DNN models to minimize the total system cost. Compared to the state-of-the-art [6], our *ACE* improves performance by over 10%.

Figure 14 depicts the running time of our proposed ACE algorithm, which includes Algorithm 2 and Algorithm 3. Notably, the running time of Algorithm 1 primarily depends on Algorithm 2 and Algorithm 3. Therefore, we focus solely on the running time of Algorithm 2 and Algorithm 3, and omit the explicit measurement of Algorithm 1. On average, the running time is around three seconds, which is significantly shorter than the length of each time slot (i.e., 15 minutes). We also scale the number of parameters in our problem model, with "1x" representing hundreds

of parameters, as shown in Figure 14. The running time of *ACE* remains acceptable under different scales.

Figure 15 displays the regret of all four algorithms. The regret represents the gap between the objective value incurred by the online algorithm and the objective value achieved by the offline optimal solutions. Among all algorithms, our proposed *ACE*'s regret demonstrates the best performance. Additionally, Figures 15 and 16 show that both the regret and fit of our proposed *ACE* algorithm exhibit sublinear growth over time, which is consistent with our previous theoretical analysis in Section 4.



(a) (b)

Figure 17 Running time and comparison with offline optimums

We clarify that our problem defined over the entire time horizon is an integer program (IP) and is NP-hard. Our proposed online relaxation and rounding mechanism is essentially an approximation approach (with proven sublinear regret) specifically designed to address this challenge in polynomial time. In contrast, directly solving this IP over each time slot using standard solvers (e.g., Gurobi/CPLEX) takes exponential time, often unacceptable in practice. Specifically, Figure 17(a) shows the per-time-slot average running time for obtaining the exact offline optimums via Gurobi for problem instances of different scales. Figure 17(b) compares our approach with the offline optimums in real time on a small-scale problem instance containing 160 decision variables at each time slot t, including 100 $x_{i,i',t}$ variables, 50 $y_{i,j,t}$ variables, and 10 $z_{i,t}$ variables. In online convex optimization, the theoretical emphasis is placed squarely on the temporal dimension because the problem size, meaning the number of decision variables, is treated as a fixed, unchanging characteristic of the environment, not a quantity that grows alongside the decision horizon. This does not mean spatial scale is ignored; rather, it is fully accounted for in the regret bounds, often through the norms of gradients and the domain diameter, which scale naturally and often linearly with the number of variables. Consequently, increasing the problem size will indeed inflate the absolute magnitude of the regret, which is both intuitive and expected, i.e., larger optimization problems are inherently harder. However, this dimensional scaling affects only the multiplicative constants, never the fundamental rate at which the average performance improves. As time extends to infinity, the per-slot average regret still vanishes regardless of how many variables are involved, ensuring the algorithm's long-term optimality remains structurally intact.

6 Related work

We summarize previous works into two categories and highlight their drawbacks compared to ours.

6.1 Cluster inference optimization

Intelligent inference systems have attracted widespread attention and research. Gu *et al.* [14] proposed a GPU cluster scheduler that reduces the average competition time under the constraints of a carbon emission budget. Chen *et al.* [7] designed Opara to accelerate DNN inference by overlapping the execution of compute-intensive and memory-intensive operators. Han *et al.* [8] placed DNN workloads using a two-stage intelligent scheduler based on resource allocation and adaptive batch size decisions. Sun *et al.* [15] proposed a batch-aware inference workload redistribution scheme, reducing overall inference loss. Qiu *et al.* [17] proposed a method to support executing multiple independent neural network models by unifying virtual memory and demand paging. Choi *et al.* [18] developed a scheduling framework for multi-models on a machine learning inference server using multiple GPUs, exploring the solution space across three dimensions: spatial and temporal sharing, as well as decision-making on batch sizes. Xu *et al.* [6] proposed an interference-aware resource configuration for inference.

Table 3 Comparing previous work to our work

Ref	Problem space					Solution	Evaluation	Year
	GPU interference	Edge scenario	Operational cost	Online scenario	Long-term constraints	Theoretical guarantees	Testbed or prototype	
[14]					✓		✓	2025
[7]	✓				✓		✓	2025
[8]	✓						✓	2024
[15]				✓			✓	2023
[6]	✓					✓	✓	2023
[17]				✓				2022
[18]	✓						✓	2022
[22]		✓	✓	✓	✓	✓		2025
[9]		✓	✓	✓		✓	✓	2025
[16]		✓			✓		✓	2024
[55]		✓					✓	2024
[10]		✓	✓	✓		✓		2023
[19]		✓				✓		2023
[20]		✓		✓		✓	✓	2023
[21]		✓				✓	✓	2023
Ours	✓	✓	✓	✓	✓	✓	✓	2026

6.2 Edge inference system

A significant amount of work has been carried out to improve the performance of edge inference. Zhao *et al.* [22] incentivized edge users to improve inference performance. Ji *et al.* [9] considered the electricity cost when deploying the models across different locations and various time slots. Wu *et al.* [16] introduced Graft, a framework for hybrid deep learning designed to restructure non-uniform DNN fragments to share layers. Chen *et al.* [55] proposed a workload allocation method based on an auction mechanism, optimizing the computing and communication time of the inference process. Ji *et al.* [10] developed an online polynomial-time algorithm designed to handle dynamic resource demands and network states. Liu *et al.* [19] studied the optimal allocation of communication and computing resources, significantly improving inference throughput. Zhao *et al.* [20] utilized online optimization techniques to address the trade-off between inference accuracy, latency, and resource costs. Dong *et al.* [21] proposed a multi-exit DNN inference acceleration framework, achieving fast inference.

6.3 Research gap

Table 3 highlights the research gaps between prior work and our approach. Broadly, previous efforts fall into two categories. The first category predominantly focuses on optimizing single-GPU performance while neglecting edge infrastructures with heterogeneous capacities and fluctuating workloads, when dispatching the inference queries among GPUs. As a result, directly adopting these methods in large-scale edge deployments can incur significant energy consumption and high resource overhead. The second category of prior work deals mostly with general-purpose computational resources (e.g., CPUs) and ignores the characteristics of GPUs, and particularly, the *interference* when multiple models are co-located on a shared GPU. Therefore, they lead to low inference speed and are unable to meet user latency requirements in the dynamic edge environment.

7 Conclusion and future work

This study tackles the challenges of deploying deep neural networks (DNNs) on edge GPU servers, with a particular focus on the performance *interference* caused by running multiple models on a shared GPU server, which can lead to excessive resource consumption or reduced accuracy. We developed an interference-incurred latency model via real-world profiling and measurements. Using this model, we proposed a set of polynomial-time online algorithms

that make decisions online for model selection, resource provisioning, and inference dispatching, achieving low long-term cost, including inference latency, error rate, query-dispatching communication cost, and energy consumption. We also proved that such algorithms exhibit sublinear growth in regret and fit over time. Experimental results show that our algorithms reduce total cost by 40% on average over all baselines.

In future work, we plan to explore hybrid *models* that combine the scalability of our linear model with a separate pairwise interference matrix learned from observed data. A possible extension is to incorporate a pairwise interference matrix as an auxiliary estimator to calibrate the posterior latency coefficients in our current linear decision model. Specifically, it can be constructed from offline profiling and online observations to estimate a bounded interference correction, which can be incorporated into the posterior latency coefficient, e.g., $\widehat{c}_{i,j,t} = c^0_{i,j,t} + \Delta c^{\text{pair}}_{i,j,t}$, where $c^0_{i,j,t}$ denotes the baseline latency coefficient and $\Delta c^{\text{pair}}_{i,j,t}$ is inferred from the pairwise interference matrix before each time slot. Under this design, the feasible region, the long-term constraints, and the randomized rounding procedure remain unchanged. Therefore, the fit is expected to remain sublinear, since it mainly depends on the constraints which remain unchanged. For the regret, the proof can be preserved under the standard boundedness assumptions if the estimated coefficients remain bounded, and the cumulative coefficient-estimation error is sublinear over time. In that case, the original regret bound would include an additional sublinear error term. Therefore, the total regret will grow sublinearly under these conditions. On the algorithmic side, we also plan to extend our algorithmic and theoretical framework to handle delayed feedback, which would require further investigation—for instance, incorporating techniques from the literature on bandits with delayed feedback [56, 57], such as maintaining pending observations and updating only when feedback arrives.

Acknowledgements This work was supported in part by the Basic Research Program of Jiangsu (No. BK20253011), the National Natural Science Foundation of China (No. 62441225 and No. 62572171), the Jiangsu Provincial Special Project for the Transformation of Scientific and Technological Achievements under Grant BA2023030, the Fundamental Research Funds for the Central Universities (No. B250201250), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM118), and the "111 Center" (No. B26023), and the Jiangsu Provincial Science and Technology Project of Water Resources under 2025024. We also thank Qiang Feng and Yuhan Zhang for their contributions to this work.

References

1 Chen L, Zhang Y, Tian B, et al. Parallel driving os: A ubiquitous operating system for autonomous driving in cpss. IEEE Transactions on Intelligent Vehicles, 2022, 7: 886–895
2 Huda N U, Ahmed I, Adnan M, et al. Experts and intelligent systems for smart homes' transformation to sustainable smart cities: A comprehensive review. Expert Systems with Applications, 2024, 238: 122380
3 Yang Y, Feng L, Sun Y, et al. Multi-cluster cooperative offloading for vr task: A marl approach with graph embedding. IEEE Transactions on Mobile Computing, 2024, pages 1–16
4 Arena F, Collotta M, Pau G, et al. An overview of augmented reality. Computers, 2022, 11
5 Multi process service. https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf, 2026. Accessed: 2026-07-01
6 Xu F, Xu J, Chen J, et al. igniter: Interference-aware gpu resource provisioning for predictable dnn inference in the cloud. IEEE Transactions on Parallel and Distributed Systems, 2023, 34: 812–827
7 Chen A, Xu F, Han L, et al. Opara: Exploiting operator parallelism for expediting dnn inference on gpus. IEEE Transactions on Computers, 2025, 74: 325–333
8 Han Z, Zhou R, Xu C, et al. Inss: An intelligent scheduling orchestrator for multi-gpu inference with spatio-temporal sharing. IEEE Transactions on Parallel and Distributed Systems, 2024, 35: 1735–1748
9 Ji M, Qi J, Jiao L, et al. Cpn meets learning: Online scheduling for inference service in computing power network. Computer Networks, 2025, 256: 110903
10 Ji M, Qian Z, Ye B. When cpn meets ai: Resource provisioning for inference query upon computing power network. In: Proceedings of 2023 IEEE 29th International Conference on Parallel and Distributed Systems, 2023. 2261-2268
11 Jin Y, Jiao L, Qian Z, et al. Provisioning edge inference as a service via online learning. In: Proceedings of 2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking, 2020. 1-9
12 Ji M, Zhang Z, Zhang Y, et al. Incentivizing edge ai with accuracy preserving via online randomized auctions. In: Proceedings of 2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking, 2023. 384-385
13 Su S, Zhou Z, Ouyang T, et al. Learning to be green: Carbon-aware online control for edge intelligence with colocated learning and inference. In: Proceedings of 2023 IEEE 43rd International Conference on Distributed Computing Systems, 2023. 567-578
14 Gu D, Zhao Y, Sun P, et al. Greenflow: A carbon-efficient scheduler for deep learning workloads. IEEE Transactions on Parallel and Distributed Systems, 2025, 36: 168–184
15 Sun H, Chen X, Qian Z, et al. Birp: Batch-aware inference workload redistribution and parallel scheme for edge collaboration. In: Proceedings of The 52nd International Conference on Parallel Processing, New York, NY, USA: Association for Computing Machinery, 2023. 72–81
16 Wu J, Wang L, Jin Q, et al. Graft: Efficient inference serving for hybrid deep learning with slo guarantees via dnn re-alignment. IEEE Transactions on Parallel and Distributed Systems, 2024, 35: 280–296
17 Qiu Z W, Liu K S, Chen Y S. Barm: A batch-aware resource manager for boosting multiple neural networks inference on gpus with memory oversubscription. IEEE Transactions on Parallel and Distributed Systems, 2022, 33: 4612–4624
18 Choi S, Lee S, Kim Y, et al. Serving heterogeneous machine learning models on multi-gpu servers with spatio-temporal sharing. In: Proceedings of 2022 USENIX Annual Technical Conference, 2022. 199–216
19 Liu Z, Lan Q, Huang K. Resource allocation for multiuser edge inference with batching and early exiting. IEEE Journal on Selected Areas in Communications, 2023, 41: 1186–1200
20 Zhao K, Zhou Z, Chen X, et al. Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale. IEEE Transactions on Mobile Computing, 2023, 22: 5870–5886
21 Dong F, Wang H, Shen D, et al. Multi-exit dnn inference acceleration based on multi-dimensional optimization for edge intelligence. IEEE Transactions on Mobile Computing, 2023, 22: 5389–5405
22 Ji M, Zhao H, Jiao L, et al. Edge ai inference as a service via dynamic resources from repeated auctions. IEEE Transactions on Mobile Computing, 2025, pages 1–17

23 Xu X, Wu F, Bilal M, et al. Xrl-shap-cache: an explainable reinforcement learning approach for intelligent edge service caching in content delivery networks. Science China Information Sciences, 2024, 67: 170303
24 Open-data-users. https://tfl.gov.uk/info-for/open-data-users/our-open-data, 2026. Accessed: 2026-07-01
25 Planetlab. http://www.planet-lab.org/, 2026. Accessed: 2026-07-01
26 Seattle. https://seattle.poly.edu/, 2026. Accessed: 2026-07-01
27 Streaming multiprocessor. https://stevengong.co/notes/Streaming-Multiprocessor, 2026. Accessed: 2026-07-01
28 Bonawitz K, Eichner H, Grieskamp W, et al. Towards federated learning at scale: System design. In: Proceedings of Talwalkar A, Smith V, Zaharia M, editors, Proceedings of Machine Learning and Systems, 2019. 374–388
29 Hanyao M, Jin Y, Qian Z, et al. Edge-assisted online on-device object detection for real-time video analytics. In: Proceedings of IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021. 1-10
30 Li W, Kuo J C, Sheng M, et al. Beyond explicit and implicit: How users provide feedback to shape personalized recommendation content. In: Proceedings of Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, 2025. 1–17
31 Cui Q, You X, Wei N, et al. Overview of ai and communication for 6g network: Fundamentals, challenges, and future research opportunities. Science China Information Sciences, 2025, 68: 171301
32 Hochba D S. Approximation algorithms for np-hard problems. 1997, 28: 40–52
33 Csirik J, Frenk J B G, Labbé M, et al. Heuristics for the 0-1 min-knapsack problem. Acta Cybernetica, 1991, 10: 15–20
34 Linear programming relaxations and rounding. https://www.cs.dartmouth.edu/~deepc/Courses/F10/Notes/lec3.pdf, 2026. Accessed: 2026-07-01
35 Du D Z, Pardalos P, Hu X, et al. Relaxation and Rounding, pages 323–348. Springer International Publishing, Cham, 2022
36 Wright S J. Primal-dual interior-point methods. SIAM, 1997
37 Yuan Y, Jiao L, Zhu K, et al. Incentivizing federated learning under long-term energy constraint via online randomized auctions. IEEE Transactions on Wireless Communications, 2022, 21: 5129–5144
38 Rockafellar R T. Lagrange multipliers and optimality. SIAM review, 1993, 35: 183–238
39 Nguyen L D, Kortun A. Real-time optimisation for industrial internet of things (iiot): Overview, challenges and opportunities. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 2020, 7
40 Ji M, Su C, Fan Y, et al. Intaas: Provisioning in-band network telemetry as a service via online learning. Computer Networks, 2024, 241: 110211
41 Jin Y, Jiao L, Qian Z, et al. Learning for learning: Predictive online control of federated learning with edge provisioning. In: Proceedings of IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021. 1-10
42 Jin Y, Jiao L, Qian Z, et al. Resource-efficient and convergence-preserving online participant selection in federated learning. In: Proceedings of 2020 IEEE 40th International Conference on Distributed Computing Systems, 2020. 606-616
43 Jukna S, Jukna S. Linearity of expectation. Extremal Combinatorics: With Applications in Computer Science, 2011, pages 255–278
44 Klebanov I, Sprungk B, Sullivan T J. The linear conditional expectation in hilbert space. Bernoulli, 2021, 27: 2267–2299
45 Optimization Integer Linear Programming. https://mypage.cuhk.edu.cn/academics/junfengwu/materials/MAT3007/Slides/lecture26.pdf, 2026. Accessed: 2026-07-01
46 Wigren T. The cauchy-schwarz inequality : Proofs and applications in various spaces. 2015
47 Chen T, Ling Q, Giannakis G B. An online convex optimization approach to proactive network resource allocation. IEEE Transactions on Signal Processing, 2017, 65: 6350–6364
48 NVIDIA V100 TENSOR CORE GPU. https://www.nvidia.com/en-us/data-center/v100/. Accessed: 2026-07-01
49 GeForce RTX 20 Series. https://www.nvidia.com/en-us/geforce/20-series/. Accessed: 2026-07-01
50 Orlowski S, Wessäly R, Pióro M, et al. Sndlib 1.0—survivable network design library. Networks: An International Journal, 2010, 55: 276–286
51 Bridges R A, Imam N, Mintz T M. Understanding gpu power: A survey of profiling, modeling, and simulation methods. ACM Computing Surveys, 2016, 49: 1–27
52 gpu-specs. https://www.techpowerup.com/gpu-specs/, 2025. Accessed: 2026-07-01
53 Ouyang T, Zhao K, Zhang X, et al. Dynamic edge-centric resource provisioning for online and offline services co-location. In: Proceedings of IEEE INFOCOM, 2023
54 Welcome to CVXPY 1.1. https://www.cvxpy.org, 2026. Accessed: 2026-07-01
55 Chen Y, Luo T, Fang W, et al. Edgeci: Distributed workload assignment and model partitioning for cnn inference on edge clusters. ACM Transactions on Internet Technology, 2024
56 Shi L, Wang J, Wu T. Statistical inference on multi-armed bandits with delayed feedback. In: Proceedings of Krause A, Brunskill E, Cho K, et al., editors, Proceedings of the 40th International Conference on Machine Learning. PMLR, 2023. 31328–31352
57 Masoudian S, Zimmert J, Seldin Y. A best-of-both-worlds algorithm for bandits with delayed feedback with robustness to excessive delays. Advances in Neural Information Processing Systems, 2024, 37: 141071–141102