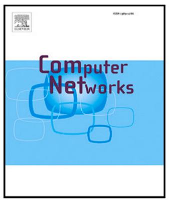
Computer Networks 256 (2025) 110903

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



CPN meets learning: Online scheduling for inference service in Computing Power Network[☆]

Mingtao Ji [a], Ji Qi [b], Lei Jiao [c], Gangyi Luo [b], Hehan Zhao [a], Xin Li [d], Baoliu Ye [a], Zhuzhong Qian [a,*]

[a] State Key Laboratory for Novel Software Technology, School of Computer Science, Nanjing University, Nanjing 210023, China
[b] China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou, Jiangsu, China
[c] Department of Computer Science, University of Oregon, Eugene, OR 97403, USA
[d] Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 211106, China

ARTICLE INFO

Keywords:
Scheduling for Computing Power Network
Online Learning
Resource Provisioning
Online Algorithm

ABSTRACT

The advent of Computing Power Network (CPN) has opened up vast opportunities for machine learning inference, yet the challenge of reducing high operational cost due to intensive computations and the sheer volume of inference tasks cannot be overlooked. Scheduling inference tasks for mitigating operational cost involves various challenges, such as migrating tasks under unpredictable CPN status, making time-coupled decisions for resource provisioning, and selecting computing sites based on dynamic electricity prices. To address these issues, we introduce CPN-Inference, a novel and flexible inference framework built upon CPN. Specifically, we formulate a time-varying integer program problem that aims to minimize long-term cost, involving switching cost, operational cost, communication cost, queuing cost, and accuracy loss. We also propose a group of polynomial-time online algorithms for supporting the formulated problem by solving delicately constructed subproblems based on the inputs predicted via online learning. Furthermore, our algorithms are proven for their competitive ratio, showcasing the performance gap between our approach and the offline optimum. A testbed is constructed to evaluate inference performance on real devices. Our comprehensive evaluations, based on datasets from real systems, demonstrate that our algorithms outperform multiple alternatives, by achieving an average cost reduction of 35%.

1. Introduction

Highlighted in the Network 2030 report [1], the continuous convergence of computing power and networks has led to the emergence of a new concept, namely, Computing Power Network (CPN). Specifically, CPN integrates a rich abundance of computational resources including clouds, edges, and devices of different enterprises or organizations, aiming at scheduling resources for different users on demand [2–4]. Presently, with the ITU-T's initial release of the CPN standard [5], massive enterprises and organizations have begun incorporating CPN into their strategic plans, as evidenced by initiatives such as China's "Eastern Data and Western Computing" [6] and UC Berkeley's "Sky Computing" plan [7]. This illustrates that CPN is a promising technological architecture to support a multitude of emerging AI services [8, 9].

As machine learning inference (e.g., DNN inference [10–12]) plays a significant role in various fields (e.g., Virtual Reality [13] and Augmented Reality [14]), these applications are expected to be key services deployed on CPN [15]. However, existing inference frameworks often neglect the variability in operational cost across different regions [8], which can lead to increased economic cost and hinder the flexible deployment of inference services within CPN. Although we can reduce energy consumption by switching to a lower version of the model (low computation), the accuracy of this version is relatively poor, making it difficult to meet users' QoS by relying solely on low-accuracy models in the long term. To address these kinds of challenges, we introduce the CPN-Inference framework, as illustrated in Fig. 1. Service providers (e.g., AR providers) only need to provide us with their model types and the number of inference queries per time slot. CPN-Inference controller will take over the resource scheduling and inference migration in a cost-minimal manner.

☆ The preliminary version entitled "When CPN Meets AI: Resource Provisioning for Inference Query upon Computing Power Network" was accepted by 2023 IEEE ICPADS (Ji et al., 2023).
* Corresponding author.
E-mail addresses: jmt@smail.nju.edu.cn (M. Ji), qiji@cmss.chinamobile.com (J. Qi), jiao@cs.uoregon.edu (L. Jiao), luogangyi2@cmss.chinamobile.com (G. Luo), hehan.zhao@mail.utoronto.ca (H. Zhao), lics@nuaa.edu.cn (X. Li), yebl@nju.edu.cn (B. Ye), qzz@nju.edu.cn (Z. Qian).

https://doi.org/10.1016/j.comnet.2024.110903
Received 26 April 2024; Received in revised form 11 October 2024; Accepted 5 November 2024
Available online 14 November 2024

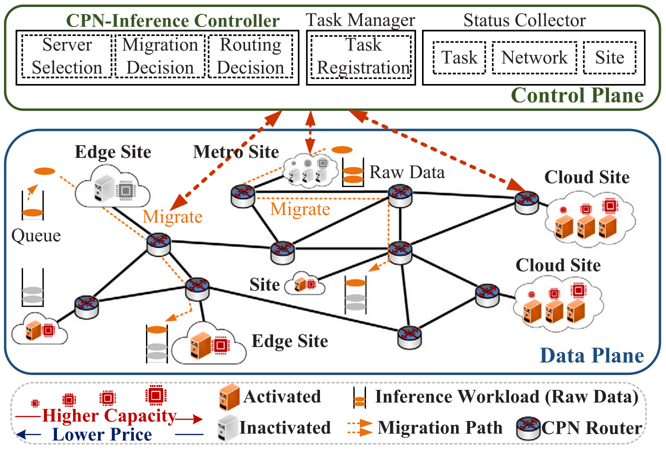


Fig. 1. Inference service framework in CPN.

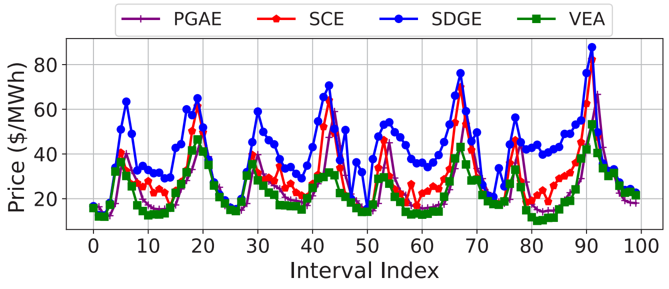


Fig. 2. CAISO electricity price public dataset [18].

Yet, it is non-trivial to design such a service controller to optimally conduct inference upon Computing Power Network, facing multiple challenges: computing site selections upon dynamic electricity prices, the time-coupled decision for resource provisioning, and task migration under the unpredictable CPN status.

Firstly, the dynamic and uncertain nature of real-world CPN platforms [16], especially concerning the link latency between computing sites, further complicates the migration of workloads. Specifically, as for one computing site where the number of inference workloads is too slight, we consider not using any servers of this site and migrating the inference workloads to another site for executions, as shown in Fig. 1. The uncertain network status [17] hinders us from making decisions for workload migrations since we likely choose migration paths with high latency, influencing the response time of inference queries. Especially, this kind of migration decision often has to be made without prior knowledge of network latency. Although one could employ a predictive mechanism to predict them and adjust the system accordingly, ensuring the accuracy of such predictions is challenging, particularly when compared to an offline optimum achieved through hindsight.

Secondly, managing these resources online remains challenging as the scale of inference workloads in each computing site often changes dynamically over time. In general, we often consider dynamic management of resources according to the inference workloads [8]. For example, as shown in Fig. 1, each computing site has different capacities for servers and we prefer to use (switch on) the high-performance servers when the inference workload is high and do not choose to use (switch off) them but use low-performance servers when the inference workload is low. However, this kind of dynamic management results in additional cost, called switching cost, such as the time required for booting the server hardware and initializing service software on the server, when we use a new server that has not been used before. Therefore, determining the on/off status in the long term for servers is complex: keeping a server always on might lead to additional operational cost (i.e., electricity cost or rental prices in each time slot), while turning it off could result in additional switching cost if we need to reuse it in the following time slot. The decisions of managing resources are time-coupled and complicated, when designing CPN-Inference Controller.

Thirdly, the fluctuating nature of electricity prices further complicates the scheduling of inference tasks. As shown in Fig. 2, we obtained real-time electricity prices for a certain period from public datasets [18] in California, covering four regions serviced by the power companies PGAE, SCE, SDGE, and VEA, respectively. Electricity prices not only vary significantly across regions but also fluctuate within the same region, impacting decision-making for optimal task scheduling. In China, commercial electricity prices are also set to be floating. According to the report [19], the nationwide maximum peak-to-valley price difference in October 2023 was 1.32 CNY/kWh. This variation in electricity prices poses challenges in decision-making when scheduling inference tasks in CPN. Specifically, inference tasks involve complex calculations and often have high power consumption [20]. In order to have low electricity cost, service providers often want to schedule inference tasks to sites with low electricity prices. Due to the dynamic changes in electricity prices, service providers are unable to achieve optimal deployment through a single decision. It is necessary to make decisions *online* according to the dynamic electricity prices. Moreover, even if a site often has the lowest electricity price, migrating all inference tasks into this site incurs extra communication latency, prolonging the response time. Thus, the electricity price of each computing site should be considered, delicately, when designing the CPN-Inference Controller.

Existing works are inadequate for addressing these challenges above. Some works [3,4,21] have studied the architecture and standards of CPN, but they do not capture the dynamic character of CPN, which cannot be directly applied to machine learning inference. While others [22–25] have explored online service provisioning, they have not taken into account the specific characteristics of inference tasks (e.g., model inference speed), easily resulting in long latency. A significant amount of works [8,9,26–28] study inference tasks in cloud–edge scenarios, but they assume the resources are all available, which ignores the cost of using this computing power in a CPN scenario and easily leads to high economic cost.

To address these challenges for the CPN-Inference Controller, this study initially formulates the optimization problem as a time-varying integer programming. The objective is to minimize long-term cost, including the switching cost of initializing a new server, the operational cost of maintaining servers, the communication cost for migrating the inference workloads, the queuing cost for arrived inference queries, as well as the accuracy loss. Note that, there exist stochastic inputs in our problem, which are revealed only after decisions are made. These stochastic inputs hinder us from designing an elegant online algorithm with a theoretical guarantee.

Afterward, we propose a collection of polynomial-time online algorithms that make decisions in an online manner. *For Challenge 1*: An online learning algorithm is designed to address the dynamic fluctuations of the CPN network. This is essentially an online convex optimization approach, which continually adjusts decisions for the next time slot based on feedback from previously deployed decisions. *For Challenge 2*: A decoupling & lazy switch algorithm is adopted to make time-coupled decisions. The long-term optimization problem is decoupled into single time-slot problems based on the switching cost, dynamically limiting the ratio between the switching cost and the rest terms. *For Challenge 3*: A relaxation & rounding algorithm is utilized to balance the trade-off between electricity cost and response time. Specifically, the original optimization problem with integer domain is relaxed to the one with real domain, and a carefully designed rounding algorithm is employed to obtain the integer solution. This proposed

Table 1
Summary of previous works and comparison with ours.

Ref.	Problem scenario				Method		Evaluation tools	
	CPN	Inference task	Dynamic allocation	Electricity price	Parameter prediction	Theoretical guarantee	TestBed	Real device
[3]	✓		✓					
[4]	✓							
[30]	✓	✓					✓	✓
[31]	✓		✓		✓	✓		✓
[32]			✓			✓		
[24]			✓			✓	✓	✓
[8]		✓	✓			✓		
[9]		✓	✓		✓		✓	✓
[26]		✓	✓			✓	✓	✓
[27]		✓	✓			✓		
[28]		✓	✓				✓	✓
[33]		✓	✓			✓		
[34]				✓	✓		✓	
[35]			✓	✓	✓	✓		
[36]			✓	✓			✓	✓
[37]				✓	✓			
[10]				✓	✓			
[38]	✓	✓	✓		✓	✓		
Our	✓	✓	✓	✓	✓	✓	✓	✓

rounding method ensures that the feasible solution solved is as close as possible to the optimal solution. The contributions of this paper are shown as follows:

- To reduce the operational cost, we propose to schedule inference in computing power network online and cast the time-varying integer programming problem to wisely arbitrate the challenges mentioned above, including the dynamic CPN, time-coupled decision, and fluctuating of electricity prices.
- We propose a lazy-switching-based switching resource management scheme, which invokes an online learning framework to overcome uncertainty. Our rounding algorithm transforms the fractional results into integers without violating any constraints. Theoretical analysis shows that our approach has a parameterized competitive ratio.
- Extensive experiments using real-world data under realistic settings are conducted [29]. We observe that our algorithm reduces the overall system cost by at least 12%, compared with the other algorithms. Besides, our algorithm is evaluated under various settings, achieving an average cost reduction of 35%, compared to the other algorithms.

As a reminder, the organization of this paper is illustrated as follows: Section 2 reviews related work and identifies our research gap. Section 3 provides a formal problem description and transformation. Sections 4 and 5 cover the algorithm design and theoretical analysis, respectively. Section 6 describes the testbed and presents experimental evaluations. We conclude our work and introduce the future research plan in Section 7.

2. Related work

We summarize previous works in three categories, and highlight the drawbacks compared with our work.

2.1. Overview of computing power network

Computing Power Network (CPN) was extensively investigated in work [2,39]. A lot of groups such as Internet Engineering Task Force (IETF), China Unicom, China Mobile, Broadband Forum, etc., established standardization [4] and/or released white papers for CPN. Then, a computing-networking scheme called compute-first networking is introduced [39]. This scheme is based on the deep fusion of cloud, edge, and network technologies. It provides a detailed explanation of the CFN technology framework and the CFN routing protocol. A prototype testbed [30] was developed for CPN using Kubernetes and microservice architecture, incorporating crucial technologies such as computing modeling, computing awareness, computing announcement, and computing offloading. This work demonstrated improved response times and enhanced load balancing compared to traditional edge computing models. Research [31] introduced CPN-FedSL, a new and adaptable Federated Split Learning (FedSL) framework designed for implementation over the Computing Power Network (CPN), which incorporates an integrated framework to handle basic settings and learning dynamics like training flow, latency, and convergence. Work [32] introduced a hybrid coded edge computing network to enable ubiquitous Artificial Intelligence (AI) in next-generation wireless communication networks, where energy-constrained end users can perform computation-intensive tasks such as data processing and model training through local computation powered by wireless power transfer, coded edge offloading, or a combination of both.

While most existing work extensively investigates the architecture and standards of CPN, they may lack a focus on dynamically allocating resources for specific tasks such as inference tasks. This could lead to inefficient utilization of computing resources and high resource cost.

2.2. Online resource provisioning for inference

Yin et al. [22] proposed a decision support system named DISC which helps service providers subscribe to the appropriate resource by solving the tradeoff between the cost and performance. Then, Douros et al. [23] considered sharing the profits fairly by maximizing the overall profits in the system. Tian et al. [24] proposed CCT to reduce flow completion time by reducer placement and bandwidth allocation. Furthermore, Dong et al. [25] introduced sub-optimal resource provision solutions for content providers by using deep reinforcement learning-based algorithms. Jin et al. [8] proposed to redistribute the inference queries according to the network bandwidth between the edge devices. Then, Sun et al. [9] proposed BIRP to accelerate the inference queries by combining the inference from different edge devices. Bai et al. [26] used a DNN ensemble to deploy the DL model on the edge devices, in order to improve the performance of inference services at the edge. She et al. [27] proposed to utilize several early-exit Deep Neural Networks to provide different kinds of services for users in the edge. Xu et al. [28] introduced iGniter to accelerate GPU interference in the cloud. Su et al. [33] focused on edge scheduling for learning and inference tasks, minimizing accuracy loss under resource constraints.

While these works optimize a variety of applications and manage resources for clusters, they do not consider the specific characteristics of inference tasks (e.g., inference speed), resulting in long inference latency. Although some works consider inference tasks in edge scenarios, they assume the resources are all available, which ignores the cost of using this computing power in the CPN scenario and easily leads to high resource cost.

2.3. Electricity price prediction

This literature [40] reviews current works on electricity cost prediction. Ugurlu et al. [34] employed a multi-layer gated recurrent unit (GRU) to forecast electricity prices, and experimental evaluations revealed that the effectiveness of a three-layer GRU surpassed other recurrent neural network (RNN) models. Zhang et al. [35] proposed a pricing approach and an auction approach to save electricity cost for co-location data centers. Zheng et al. [36] addressed the significant operating cost, primarily energy cost, that edge service providers incurred due to the fluctuating power demand in edge cloud (EC) infrastructures, where electricity cost comprise both energy and demand charges, with the latter often forming a large portion due to peak power usage. Jiang et al. [37] proposed using a long short-term memory (LSTM) model to predict the electricity price for the next day. This model comprehensively considers various factors such as holidays, weather conditions, and oil prices. Tan et al. [10] designed a hybrid deep learning model, combining convolutional neural networks with stacked sparse denoising autoencoders, to predict electricity prices.

However, these works solely focus on predicting electricity prices without considering their integration with the deployment of inference tasks. This kind of separation of these two aspects may lead to a dilemma: scheduling tasks to regions with low electricity prices may result in high latency, hard to ensure timely response for inference tasks; but scheduling tasks to low-latency sites without considering electricity prices leads to high electricity cost.

2.4. Research gap

Table 1 summarizes all relevant studies considering problem scenarios, methods, and experimental evaluations, from which several key conclusions can be drawn: (1) Most research on CPN often overlooks the characteristics of inference tasks, such as accuracy loss, which can easily lead to poor user QoS; (2) Studies on the deployment categories of inference services often do not consider the operational cost, potentially resulting in high economic expenses; (3) Work on operational expenses frequently ignores the characteristics of CPN and the features of inference tasks, making them inapplicable in our scenario.

Our research, based on CPN, takes into account the characteristics of inference tasks and the fluctuating nature of electricity prices, proposes a theoretically guaranteed algorithm, and constructs a testbed for validation.

3. System model and problem formulation

3.1. System settings and models

CPN-Inference Infrastructure: The major notations utilized throughout this work are summarized and explained in Table 2. We consider our CPN-Inference as a specialized network slicing in the computing power network, tailored specifically for the inference service. As shown in Fig. 1, the data plane of CPN-Inference can be formulated as a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, 2, .., N\}$ represents a group of computing sites (e.g., edge, metro, and cloud sites) and $\mathcal{E} = \{1, 2, .., E\}$ represents a set of links. Each site has its own CPN router (supporting segment routing) and all computing sites are connected to one another through the network links. Typically, we consider the computing power sites that are powered by the grid [5], rather than the battery. At each computing site, we use $\mathcal{J} = \{1, 2, \ldots, J\}$ to represent the set of different types of servers constructed by service providers or other enterprises/individuals [2]. Service providers can use the servers owned by themselves and/or utilize the servers owned by other enterprises/individuals if they do not have enough resources. In general, the server with a larger processing capability is charged a higher rental price by its owner [2,41]. We use $p_j, \forall j \in \mathcal{J}$ to represent the computing/processing capability for server j. The control plane of CPN-Inference comprehensively controls both computing and network resources, ensuring (1) selecting the appropriate server resources from multiple owners to deploy inference services, (2) efficient scheduling of each inference task to the optimal site along the most efficient paths.

Table 2
Major notations used in our formulation.

Symbol	Description
$\mathcal{N}, \mathcal{E}$	Set of computing sites and network links in Computing Power Network (CPN)
$\mathcal{J}, \mathcal{T}$	Set of all types of servers and time slots
p_j	Processing capacity for the j type of server
ϖ_j	Power of the j type of server
$q_{i,t-1}$	Remaining unresolved inference queries at computing site i after time slot $t-1$
$\aleph_{i,t}$	Unit of electricity price for computing site i at time slot t
$r_{i,t}$	Number of queries submitted to site i at t
s_j	Switching cost per unit for activating server j
$o_{i,j,t}$	Operational (electricity/rental) cost of server j at computing site i
$\iota_{j,t}$	Accuracy of the model deployed on the jth type server
Δt	Length of each time slot in this system
M_i	Queuing capacity for computing site i
Decision	**Description**
$y_{i,j,t}$	Whether to activate server j at computing site i at time slot t
$z_{i',i,t}$	Ratio of queries migrated from computing site i' to computing site i

Operational Cost: We consider a group of consecutive time slots denoted by $\mathcal{T} = \{1, 2, .., T\}$. It will incur operational cost (denoted by $o_{i,j,t}, \forall i \in \mathcal{N}, j \in \mathcal{J}, t \in \mathcal{T}$) if service providers utilize these servers: when they use servers owned by themselves, the operational cost is called the electricity cost; when they utilize servers from other enterprises/individuals, it is called rental cost. *(i) Electricity Cost*: The electricity cost for server j at computing site i can be calculated as $o_{i,j,t} \triangleq \aleph_{i,t}\varpi_j \Delta t, \forall i \in \mathcal{N}, j \in \mathcal{J}, t \in \mathcal{T}$, where $\aleph_{i,t}$ represents the unit of electricity price for computing site i at time slot t, ϖ_j indicates the power of server $j \in \mathcal{J}$, and Δt represents the length of each time slot. *(ii) Rental Cost*: In each time slot, the *owners* of computing sites set the rental prices for their servers, and the service providers lease servers according to the decisions made by the service controller. After paying the fee according to the rental prices, service providers deploy inference services on these servers. In general, the prices of servers at each time slot are also correlated with the electricity cost of the servers. Specifically, the rental price for server j can be calculated by $o_{i,j,t} = o^1_{i,j,t} + \Upsilon_{i,j,t}$, where $o^1_{i,j,t}$ represents the electricity cost and $\Upsilon_{i,j,t}$ represents the profits for the server j of computing site i at time slot t.

Inference Workload Processing: We use $r_{i,t}, \forall i \in \mathcal{N}, t \in \mathcal{T}$ to represent the number of inference queries submitted by the users around computing site i at time slot t. Each computing site maintains a first-in-first-out (FIFO) queue and the total inference query tasks enter the queue before being served. For the computing site without activated servers, the inference queries from such a site can be migrated to another site. For example, the grey servers in Fig. 1 are represented inactivated. We use $b_{i',i,t}$ to represent the unit cost of migrating a single inference task from the site i' to i. Inference queries that were not processed in the previous time slot will be delayed for processing in the current time slot. We use $q_{i,t-1}$ to represent the remaining/untreated queries at time slot $t-1$.

Control Decisions for INT-Inference Controller: We use $y_{i,j,t} \in \{0, 1\}, \forall i \in \mathcal{N}, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}$ to represent the resource provisioning (i.e., whether to activate (use/rent) server j at computing site i in time slot t). We use $z_{i',i,t} \in [0,1], \forall i' \in \mathcal{N}, \forall i \in \mathcal{N}, \forall t \in \mathcal{T}$ to denote the ratio of inference tasks migrated from site i' to site i.

Cost of Resource Provisioning: We consider the operational and switching cost associated with maintaining the running servers at computing sites. Operational cost is the electricity cost or the fees charged by the resource owners (e.g., enterprises or individuals [2]). The operational cost of site i are calculated by $\sum_j y_{i,j,t} o_{i,j,t}$ at time slot t, which is incurred only if any one server in a site is in activated status. To reduce operational cost, we allow the servers in sites to be switched on and off, dynamically. In this work, we consider server and service as a bundle [42]. When we activate a server, we not only boot the server hardware, but also start and initialize the operating system, the runtime environment, and the service software on the server. The start-up cost, also called the switching cost, including the lead time on all of these operations, is incurred only when we use a new server that has not been used before. We use s_j to represent the switching cost for initializing the server j at time slot t, and define the corresponding switching cost as $[y_{i,j,t} - y_{i,j,t-1}]^+ s_j$, where $[\cdot]^+ = \max\{\cdot, 0\}$, linking two consecutive time slots.

Cost of Inference Migration and Queuing (QoS): Since $b_{i',i,t}$ represents the unit cost when one inference query is migrated to computing site i along path[1] $\mathcal{L}_{i',i}$, the total migration cost is illustrated as $\sum_{i',i} b_{i',i,t} z_{i',i,t} r_{i',t}$ at time slot t. The total inference queries arrived at the computing site i in time slot t consist of: (i) the remaining/unresolved queries in the time slot $t-1$, denoted by q_{t-1}; (ii) the newly arrived queries (i.e., $\sum_{i'} r_{i',t} z_{i',i,t}$). We define the queuing cost for inference queries as the queue's length of site i at the end of the time slot, which can be calculated by $q_{i,t} \triangleq [q_{i,t-1} + \sum_{i'} r_{i',t} z_{i',i,t} - \sum_j p_j y_{i,j,t}]^+$, where $\sum_j p_j y_{i,j,t}$ represents the total processing capacity of computing site i. In our machine learning inference system, deadlines for individual queries are not applicable, as resolving each query typically occurs quickly, in milliseconds based on real-world measurements. Instead, we adopt a time-slotted model to represent system behavior, where each time slot (in minutes or longer) handles multiple inference queries. Thus, the quality of service (QoS) in this work is not measured by meeting strict deadlines but by minimizing the unresolved queries at each time slot.

Cost of Inference Accuracy Loss: There is a set of model versions for each type of inference service, denoted by $\mathcal{M} = \{1, 2, \ldots, |\mathcal{M}|\}$, where models with higher accuracy consume more resources (e.g., CPU and memory). By default, we select the most suitable model for each type of server, and use $\imath_{j,t}$ to represent the accuracy of the model deployed on the jth type of server at time slot t. Here, the inference loss can be defined as 1 minus the model accuracy. Therefore, accuracy loss for the overall system is then defined as $\sum_{i,j}(1 - \imath_{j,t}) y_{i,j,t}$. We need to ensure that the accuracy loss is minimized as much as possible, as a smaller loss implies better performance (QoS) for the inference tasks. But higher-version models offer better accuracy at the expense of higher operational cost (involving complex calculations). On the other hand, lower-version models incur lower operational cost but exhibit poorer accuracy (QoS). Not only that, accuracy loss for inference queries often varies over time. To validate this behavior, we conduct experiments on a real device using YOLOv3 to process real-time video streams (Tokyo streets from YouTube [44]). As shown in Fig. 3, the accuracy loss continuously fluctuates along with the arrived video content, which is a posterior parameter.

[1] According to previous work [15,43], we also consider the conventional approach of single-path routing (unsplittable flow). Then, we can use the segment routing technique to implement it practically.

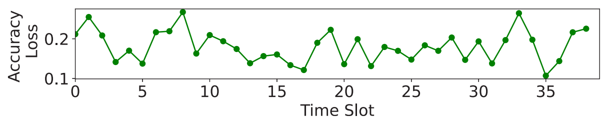


Fig. 3. Dynamic accuracy loss over time.

3.2. Problem formulations and challenges

Control Problem $\mathbb{P}$: Based on the descriptions above, we formulate the following optimization problem for our CPN-Inference Controller, which controls resource provisioning for the inference tasks upon the computing power network (CPN) infrastructure:

$$\min\ \mathcal{P} = \sum_{t\in\mathcal{T}} \{ \sum_{i,j} [y_{i,j,t} - y_{i,j,t-1}]^+ s_j + \sum_{i,j} y_{i,j,t} o_{i,j,t} + \sum_{i,i'} z_{i',i,t} b_{i',i,t} r_{i',t} + \sum_i q_{i,t} + \sum_{i,j} (1 - \imath_{j,t}) y_{i,j,t} \}$$

$$s.t.\ \sum_{i\in\mathcal{N}} z_{i',i,t} = 1, \forall i' \in \mathcal{N}, \forall t \in \mathcal{T}, \quad (1)$$

$$\sum_{i'\in\mathcal{N}} z_{i',i,t} \le \sum_{j\in\mathcal{J}} y_{i,j,t} M_i, \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \quad (2)$$

$$var.\ y_{i,j,t} \in \{0,1\}, z_{i',i,t} \in [0,1],$$
$$\forall i' \in \mathcal{N}, \forall i \in \mathcal{N}, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}, \quad (3)$$

whose objective is to minimize the long-term cost of inference query provisioning, specifically, including switching cost and operational cost, communication cost for task migration, query queuing cost, and inference loss. Constraint (1) ensures that all of the inference queries reached have to be distributed. Constraint (2) ensures that the total received queries respect the queuing capacity of the computing site. Constraint (3) contains the domains of variables. Please note that our problem is a multi-objective optimization problem, and any service provider can adaptively set weights for the five items according to their needs.

Control Problem $\widehat{\mathbb{P}}$ with Predicted Inputs: In each time slot, when we make decisions, we cannot observe the part real inputs of $\mathbb{P}$ (i.e., $b_{i',i,t}$, $o_{i,j,t}$ and $\imath_{j,t}$). This means that we can only solve this problem by using the predicted inputs instead of the real inputs. Therefore, we first introduce an online learning based prediction mechanism to generate these inputs. Specifically, the predicted unit cost for one query migration, denoted by $\hat{b}_{i',i,t}$, the predicted operational cost is denoted by $\hat{o}_{i,j,t}$, and the predicted accuracy of model is denoted by $\hat{\imath}_{j,t}$. Then, the problem constructed with the above-predicted inputs is reformulated as $\widehat{\mathbb{P}}$:

$$\min\ \widehat{\mathcal{P}} = \sum_{t\in\mathcal{T}} \{ \sum_{i,j} [y_{i,j,t} - y_{i,j,t-1}]^+ s_j + \sum_{i,j} y_{i,j,t} \hat{o}_{i,j,t} + \sum_{i,i'} z_{i',i,t} \hat{b}_{i',i,t} r_{i',t} + \sum_i q_{i,t} + \sum_{i,j} (1 - \hat{\imath}_{j,t}) y_{i,j,t} \}$$

$$s.t.\ Constraints\ (1)\sim(3).$$

Algorithmic Goal: Based on the above two formulated problems, we now introduce the goal of our algorithm. To this end, we first give some notations: $\boldsymbol{X}$ are the aggregations[2] of $\{\boldsymbol{y}_t, \boldsymbol{z}_t, \forall t \in \mathcal{T}\}$; $\boldsymbol{X}^*$ represents the optimal solutions of our problem $\mathbb{P}$ assuming that all inputs are known in advance and define $\mathcal{P}^* = \mathcal{P}(\boldsymbol{X}^*)$; $\bar{\boldsymbol{X}}_\triangle$ refers to the feasible solutions solve from our problem $\widehat{\mathbb{P}}$. We aim to design algorithms that generate solutions $\bar{\boldsymbol{X}}_\triangle$ in an online manner while ensuring the upper bound for the competitive ratio defined as $\xi = \mathcal{P}(\bar{\boldsymbol{X}}_\triangle)/\mathcal{P}^*$.

Problem Challenges: However, designing algorithms to meet our above goal is non-trivial due to the following several challenges:

[2] Bold symbols denote column vector, e.g., $\boldsymbol{z}_t^\top = [\ldots, z_{i',i,t}, \ldots]$.

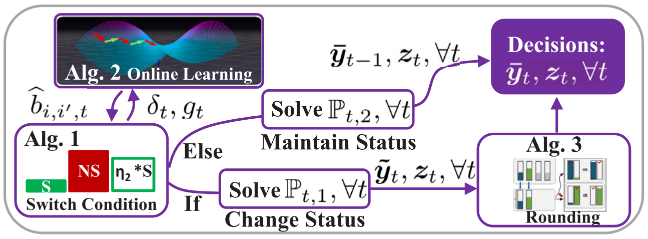

Alg. 2 Online Learning
$\widehat{b}_{i,i',t}$
δ_t, g_t
Alg. 1
NS
η_2*S
S
Switch Condition
Else
If
Solve $\mathbb{P}_{t,2}, \forall t$
Maintain Status
Solve $\mathbb{P}_{t,1}, \forall t$
Change Status
$\bar{y}_{t-1}, z_t, \forall t$
$\tilde{y}_t, z_t, \forall t$
Decisions: $\bar{y}_t, z_t, \forall t$
Alg. 3
Rounding

Fig. 4. Flowcharts of our proposed algorithms.

We first need to ensure that the solution derived from the problem $\widehat{\mathbb{P}}$ with predicted inputs has performance guarantees when they are substituted in the original problem $\mathbb{P}$ with the actual inputs. Specifically, we need to compare such a solution with the offline optimum of problem $\mathbb{P}$, which is a strict requirement of algorithm design. To achieve this, we need to jointly design both the online prediction algorithm (for generating the predicted inputs) and the online control algorithm (for decision-making).

Another hurdle is the unpredictability of real-time inputs in the CPN system. Even if given the predicted inputs, problem $\widehat{\mathbb{P}}$ must be solved in real time. Furthermore, the switching cost in problem $\mathbb{P}$ links the decisions in the current time slot and the decisions in the next time slot. Making decisions for the current time slot to minimize the system cost becomes challenging without foresight into the next time slot's decisions which are only revealed after we make these decisions.

The last challenge is that our proposed problem is highly complex. It is a mixed-integer program problem with several non-linear discontinuous functions, e.g., the aforementioned switching cost. Even without these non-linear functions, our problem is still an NP-hard [45] problem that cannot be solved optimally in polynomial time.

In most countries and regions, commercial electricity prices fluctuate, indicating that our work, focused on dynamic electricity pricing, has broad applicability. Even for countries with fixed electricity prices, the three challenges above still exist, and this research remains practical value.

4. Algorithm design

To overcome all the challenges mentioned before, we design three online polynomial-time algorithms. The relationships between the algorithms are illustrated in Fig. 4. Specifically, our Algorithm 1 adopts the "lazy switch" which maintains the on/off status of the servers until a carefully designed condition is satisfied. Once satisfied, we solve the constructed problem with fractional domains and obtain fractional solutions. Then, Algorithm 3 rounds them into integers without violating any constraints in $\mathbb{P}$. Based on the current observations and decisions, Algorithm 2 predicts the inputs of problem $\mathbb{P}$ for the next time slot via an online learning method.

4.1. Algorithm overview with problem decomposition

We present auxiliary notations to aid in our algorithm design. First, we split the objective function of problem $\widehat{\mathbb{P}}$ according to the nonlinear term $[\cdot]^+$. For $t \in \mathcal{T}$, we have the following two items:

$$C^t_S(\boldsymbol{y}_t, \boldsymbol{y}_{t-1}, \boldsymbol{z}_t, \boldsymbol{z}_{t-1}) = \sum_{i,j}[y_{i,j,t} - y_{i,j,t-1}]^+ s_j$$

$$\widehat{C}^t_{\neg S}(\boldsymbol{y}_t, \boldsymbol{z}_t) = \sum_{i,j} y_{i,j,t}\widehat{o}_{i,j,t} + \sum_{i,i'} z_{i',i,t}\widehat{b}_{i',i,t} r_{i',t}.$$
$$+\sum_i [q_{i,t-1} + \sum_{i'} r_{i',t} z_{i',i,t} - \sum_j p_j y_{i,j,t}]^+ + \sum_{i,j}(1-\widehat{\imath}_{j,t}) y_{i,j,t},$$

Then, we define two auxiliary problems $\mathbb{P}_{t,1}$ and $\mathbb{P}_{t,2}$ as follows:

$$\min\ \mathcal{P}_{t,1} = \widehat{C}^t_{\neg S}(\boldsymbol{y}_t, \boldsymbol{z}_t)$$

Algorithm 1 Online Control Algorithm (OCA)

```
1: Initialize: $t = t' = 1$; suitable $\boldsymbol{z}_0$, $\boldsymbol{y}_0 = \boldsymbol{y}_1 = \boldsymbol{0}$;
2: while $t \leq T$ do
3:     if $\widehat{C}^{t'}_S(\boldsymbol{y}_{t'}, \boldsymbol{z}_{t'}, \boldsymbol{y}_{t'-1}, \boldsymbol{z}_{t'-1}) \leq \frac{1}{\eta_2}\sum_{v=t'}^{t-1}\widehat{C}^v_{\neg S}(\boldsymbol{y}_v, \boldsymbol{z}_v)$ then
4:         Obtain $\widetilde{\boldsymbol{y}}_t, \widetilde{\boldsymbol{z}}_t$ by solving $\mathbb{P}_{t,1}$ via solver [46];
5:         $\boldsymbol{y}_t$ = Algorithm 3($\widetilde{\boldsymbol{y}}_t$);
6:         $\boldsymbol{z}_t \leftarrow \mathbb{P}_{t,1}$, given $\boldsymbol{y}_t$ if solvable;
7:         Otherwise set $\boldsymbol{y}_t = \boldsymbol{y}_{t-1}$;
8:         if $\boldsymbol{y}_t \neq \boldsymbol{y}_{t-1}$ then
9:             $t' = t$;
10:        end if
11:    end if
12:    if $t' < t$ then
13:        Obtain $\boldsymbol{y}_t, \widetilde{\boldsymbol{z}}_t$ by solving $\mathbb{P}_{t,2}$ via solver [46];
14:    end if
15:    $t = t + 1$;
16:    $\widehat{b}_{i',i,t+1}$ = Algorithm 2($\delta_t = b_{i',i,t}$, $g_t = z_{i',i,t} r_{i',t}$);
17:    $\widehat{o}_{i,j,t+1}$ = Algorithm 2($\delta_t = o_{i,j,t}$, $g_t = y_{i,j,t}$);
18:    $\widehat{\imath}_{j,t+1}$ = 1− Algorithm 2($\delta_t = \imath'_{j,t}$, $g_t = y_{i,j,t}$);
19: end while
```

Algorithm 2 Online Learning Based Prediction

```
Input: $\delta_t$ and $g_t, \forall t \in \{1, 2, ..., T\}$;
Output: $\widehat{\delta}_{t+1}, \forall t \in \{1, 2, ..., T\}$.
1: Initialize: proper step size $\lambda$;
2: for $t = 1, 2, ..., T$ do
3:     $\delta_t, g_t$ is revealed, construct $f_t(\delta) = (\delta - \delta_t)^2 g_t^2$;
4:     $\widehat{\delta}_{t+1} = arg\min_\delta\{\nabla f_t(\widehat{\delta}_t)(\delta - \widehat{\delta}_t) + \frac{||\delta - \widehat{\delta}_t||^2}{2\lambda}\}$;
5:     Submit $\widehat{\delta}_{t+1}$ to Algorithm 1;
6: end for
```

$$\text{s.t. } C^t_S(\boldsymbol{y}_t, \boldsymbol{y}_{t-1}, \boldsymbol{z}_t, \boldsymbol{z}_{t-1}) \leq \eta_1 \widehat{C}^t_{\neg S}(\boldsymbol{y}_t, \boldsymbol{z}_t), \tag{4}$$
$$\text{Constraints (1), (2),}$$
$$\text{var. } y_{i,j,t} \in [0,1], z_{i',i,t} \in [0,1],$$

which is denoted by $\mathbb{P}_{t,1}$. Constraint (4) ensures a specified relationship between the non-switching cost and the switching cost. Then, we define problem $\mathbb{P}_{t,2}$, as follows:

$$\min\ \mathcal{P}_{t,2} = \widehat{C}^t_{\neg S}(\boldsymbol{y}_t, \boldsymbol{z}_t)$$
$$\text{s.t. Constraints (1), (2),}$$
$$\text{var. } y_{i,j,t} = \bar{y}_{i,j,t-1}, z_{i',i,t} \in [0,1],$$

where the decisions of this problem are all taken from the previous time slot. To overcome the challenge of $[\cdot]^+$ in $\mathbb{P}_{t,1}$, we replace it with an equivalent of substitute, i.e.,

$$[y_{i,j,t} - y_{i,j,t-1}]^+ \Leftrightarrow u_{i,j,t},\ s.t. \begin{cases} u_{i,j,t} \geq y_{i,j,t} - y_{i,j,t-1}, \\ u_{i,j,t} \geq 0, \end{cases}$$

$$[q_{i,t-1} + \sum_{i'} r_{i',t} z_{i',i,t} - \sum_j p_j y_{i,j,t}]^+ \Leftrightarrow$$
$$v_{i,j,t},\ s.t. \begin{cases} v_{i,j,t} \geq q_{i,t-1} + \sum_{i'} r_{i',t} z_{i',i,t} - \sum_j p_j y_{i,j,t}, \\ v_{i,j,t} \geq 0, \end{cases}$$

After the above transformations, we introduce two types of new variables in $\mathbb{P}_{t,1}$ (i.e., $u_{i,j,t}$ and $v_{i,j,t}$). Here, both $\mathbb{P}_{t,1}$ and $\mathbb{P}_{t,2}$ can be solved via standard optimization solvers [46], since they are convex optimization problems.

Based on the transformations, we provide a detailed introduction to the algorithm procedure below. Algorithm 1 does not change the current on/off status of servers until the cumulative non-switching cost surpass the η_2 times switching cost. As shown in line 3, η_2 is the "laziness" parameter controlling the proportion of the switching cost and non-switching cost over a period of time. We also use η_1 in problem $\mathbb{P}_{t,1}$ to control the proportion of them in each time slot. η_1 and η_2 are jointly used to control the overall cost. Once the condition in line 3 for switching is met, we use the existing mature solver (i.e., CVXPY [46]) to solve $\mathbb{P}_{t,1}$ and obtain fractional solutions. Then, in line 5, we call Algorithm 3 to round them into integers without constraint violation, which will be introduced in detail in Section 4.3. After that, we will plug the integer solutions $\bar{\boldsymbol{y}}_t$ into problem $\mathbb{P}_{t,1}$ and obtain the fractional solutions for the ratio of inference queries migration (i.e., $\tilde{\boldsymbol{z}}_t$). Thus, only when $\mathbb{P}_{t,1}$ with the rounded integer solutions has feasible solutions for $\tilde{\boldsymbol{z}}_t$, these rounded integer solutions are accepted by Algorithm 1, as shown in lines 6∼11; Otherwise, we will hold the solutions $\boldsymbol{y}_{t-1}$ used in the previous time slot $t-1$, and invoke solver [46] to solve problem $\mathbb{P}_{t,2}$, as illustrated in lines 12∼14.

4.2. Inputs prediction with online learning

At the end of the current time slot, Algorithm 2 is called to predict the inputs of the next time slot (i.e., $\hat{b}_{i',i,t+1}$, $\hat{o}_{i,j,t+1}$ and $\hat{\iota}_{j,t+1}$) by using an alternating "primal–dual" approach.

Specifically, we use L_2-norm [47] (i.e., $\|\delta_{t+1}-\hat{\delta}_{t+1}\|^2$), to measure the distance between the actual value and the predicted value for the next time slot. Then, our goal is to find the optimum δ based on L_2-norm (i.e., $\hat{\delta}_{t+1} = \arg\min_\delta \|\delta_{t+1}-\sigma\|^2 g^2_{t+1}$). However, in an online manner, we cannot observe the actual inputs for the current time slot (i.e., δ_{t+1}), before we make decisions for the current time slot. Therefore, we consider the long-term version for this L_2-norm-based problem. To minimize the cumulative distance between the actual values and predicted values, the conventional approach often, equivalently, converts the problem with the form of $\min_{\delta_t}\{\sum_{t\in\mathcal{T}} f_t(\delta_t)\}$, s.t. $h_t(\delta_t)\le 0$, var. $\delta_t\in\mathcal{X}$, into the problem with the form of

$$\min_{\delta_t}\max_{\omega_t}\left\{\sum_{t\in\mathcal{T}} f_t(\delta_t)+\omega_t h_t(\delta_t)\right\},\ \text{var. } \delta_t\in\mathcal{X}, \omega_t\in\mathbb{R}^+,$$

where ω_t is the Lagrange multiplier [48,49], and $h_t(\cdot)$ are the constraint respect to δ_t. It is intuitive to employ the gradient generated from time slot t to adjust the predicted value for time slot $t+1$. Consequently, we can solve the subproblem, alternately, between minimizing the objective regarding the primal decision δ_t via a modified descent step and maximizing the objective regarding the Lagrange multiplier via a dual ascent step. Specifically, the first phase is described as follows:

$$\min \quad \nabla f_t(\delta_t)(\delta-\delta_t)+\omega_{t+1}h_t(\delta)+\frac{\|\delta-\delta_t\|^2}{2\lambda}. \tag{5}$$

At time $t+1$, the Lagrange multiplier can be calculated as

$$\omega_{t+1} = [\omega_t+\lambda'\nabla_\omega(\mathcal{L}_t(\delta_t,\omega))]^+ = [\omega_t+\lambda' h_t(\delta_t)]^+, \tag{6}$$

where $\mathcal{L}_t(\delta_t,\omega)=f_t(\delta_t)+\omega_t h_t(\delta_t)$ and λ, λ' are the step size. Since the variable δ_t has no constraint, the primal step ω_t often keep zero, $\omega_t = 0, \forall t$. Therefore, the primal phase can be further adjusted as follows:

$$\min \quad \nabla_{\delta'}\{\|\delta_t-\delta'\|^2 g_t^2\}|_{\delta'=\hat{\delta}_t}(\delta-\hat{\delta}_t)+\|\delta-\hat{\delta}_t\|^2/2\lambda,$$

where $\hat{\delta}_t$ represents the predicted input in time slot t.

4.3. Randomized rounding with preservation

Algorithm 3 is invoked after we obtain fractional solutions. Fig. 5 shows the process of this algorithm. Specifically, Algorithm 3 converts these fractional solutions to integers ensuring that (1) the expectations of integers equal the fractions and (2) the constraints are not violated. In line 2, the algorithm maintains the sets $\tilde{\boldsymbol{\Omega}}_i$, tracking the indices set for the decisions of server selection in computing site i. Then, in line 4, Algorithm 3 selects a pair of fractional solutions to round, stochastically, (i.e., Step 1 in Fig. 5). In line 6, we use $\tilde{y}_{i,j,t}$ as the probability to let the two fractions compensate for each other, and the

Algorithm 3 Randomized Rounding Mechanism

```
Input: The fractional solutions $\tilde{\boldsymbol{y}}_t = \{\tilde{y}_{i,j,t}|\forall i,j\}$;
Output: The integer solutions $\bar{\boldsymbol{y}}_t = \{\bar{y}_{i,j,t}|\forall i,j\}$.
1:  for each $i \in \mathcal{N}$ do
2:      Construct set $\tilde{\boldsymbol{\Omega}}_i = \{j|\tilde{y}_{i,j,t} \in \tilde{\boldsymbol{y}}_t\}$;
3:      while $\tilde{\boldsymbol{\Omega}}_i \ge 2$ do
4:          Choose $u, v \in \tilde{\boldsymbol{\Omega}}_i$, where $u \ne v$;
5:          $\theta_1$=min$\{1-\tilde{y}_{i,u,t}, \tilde{y}_{i,v,t}\}$,$\theta_2$=min$\{\tilde{y}_{i,u,t}, 1-\tilde{y}_{i,v,t}\}$;
6:          Update $(\tilde{y}_{i,u,t}, \tilde{y}_{i,v,t})$ according to Equality (7);
7:          if $\tilde{y}_{i,u,t} \in \{0,1\}$: $\bar{y}_{u,j,t} = \tilde{y}_{i,u,t}$, $\tilde{\boldsymbol{\Omega}}_i = \tilde{\boldsymbol{\Omega}}_i \backslash \{u\}$;
8:          if $\tilde{y}_{i,v,t} \in \{0,1\}$: $\bar{y}_{v,j,t} = \tilde{y}_{i,v,t}$, $\tilde{\boldsymbol{\Omega}}_i = \tilde{\boldsymbol{\Omega}}_i \backslash \{v\}$;
9:      end while
10:     if $|\tilde{\boldsymbol{\Omega}}_i| = 1$ then
11:         $\bar{y}_{i,u,t} = 1, u \in \tilde{\boldsymbol{\Omega}}_i$;
12:     end if
13: end for
14: return $\bar{\boldsymbol{y}}_t = \{\bar{y}_{i,j,t}|\forall i,j\}$;
```

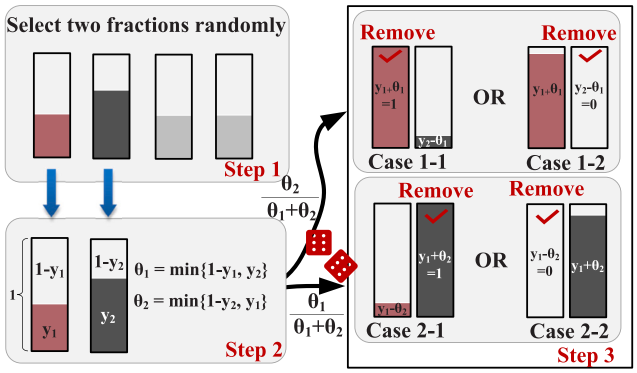


Fig. 5. Flowcharts of our randomized rounding.

specific update step is shown in Equality (7), ensuring the sum of the fractions is unchanged (i.e., Steps 2∼3 in Fig. 5).

$$(\tilde{y}_{i,u,t},\tilde{y}_{i,v,t}) = \begin{cases}(\tilde{y}_{i,u,t}+\theta_1,\tilde{y}_{i,v,t}-\theta_1), \text{ Prob}=\dfrac{\theta_1}{\theta_1+\theta_2},\\[2ex] (\tilde{y}_{i,u,t}-\theta_2,\tilde{y}_{i,v,t}+\theta_2), \text{ Prob}=\dfrac{\theta_2}{\theta_1+\theta_2}.\end{cases} \tag{7}$$

The expectation of $\tilde{y}_{i,j,t}$ in lines 3∼9 can be calculated by

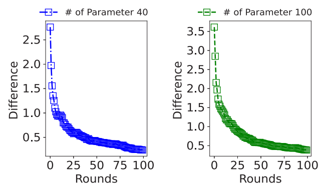
$$E[\bar{y}_{i,j,t}]=(\tilde{y}_{i,j,t}+\theta_1)*\frac{\theta_1}{\theta_1+\theta_2}+(\tilde{y}_{i,j,t}-\theta_2)*\frac{\theta_2}{\theta_1+\theta_2}=\tilde{y}_{i,j,t}. \tag{8}$$

which ensures the final integral solutions keep the expectation equal to fractional ones, which is vital to our theoretical analysis in Section 5. Due to that the sum of all the fractional solutions might not be an integer, we set the last one as 1, ensuring no constraints are violated.

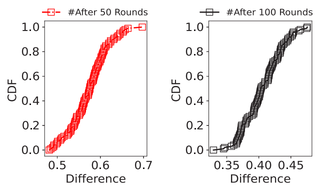
Lemma 1. *For the total $\tilde{y}_{i,j,t}, \forall i,j,t$, we have the following equality: $\sum_j \bar{y}_{i,j,t} \le \sum_j \tilde{y}_{i,j,t}+1, \forall i,t$. Besides, the time complexity of our randomized rounding mechanism is $\mathcal{O}(N*J)$.*

Proof. See Section 5.1 by randomization compensation. □

As shown in Fig. 6, we give a preliminary result on our rounding strategy based upon different parameters and a different number of rounds. We evaluate the difference between fractional solutions and integral solutions, and the result shows that it gradually shrinks along with rounds. After several tens of rounds, our proposed randomized rounding strategy could have obvious performance, which benefits our following evaluations.



(a) Different Parameters



(b) Different Rounds

Fig. 6. Performance guarantee of randomized rounding.

Actually, our proposed mechanisms can be applied, either partially or fully, to other similar scenarios. (1) Algorithm 1 can be applied when the optimization problem includes a switching cost term (i.e., $[X_t - X_{t-1}]^+ = \max\{X_t - X_{t-1}, 0\}$, where X_t is the decision variable at time slot t and X_{t-1} is the decision at time slot $t-1$) that is common in many fields. (2) Algorithm 2 is applicable when the optimization problem involves posterior parameters that change smoothly and have bounded variation. (3) Algorithm 3 can be applied when the optimization problem includes constraints, such as $\sum_i X_i \le C$, where X_i is the decision variable and C is a constant.

4.4. Time complexity

The time complexity of Algorithm 1 is primarily determined by the functions in lines 4, 5, 13, and 16∼18.

(1) *Lines 4 and 13*: Since both $\mathbb{P}_{t,1}$ and $\mathbb{P}_{t,2}$ are convex optimization problems, various algorithms [50,51] can be applied to solve them in polynomial time. The time complexity depends on the solvers used, and it is generally regarded in academia as $O(N_d^2 N_c^{2.5} + N_c^{3.5})$ where N_d is the number of decision variables and N_c is the number of constraints.

(2) *Line 5*: From Lemma 1, the time complexity of Algorithm 3 is $O(NJ)$, where N represents the number of sites and J represents the number of different types of servers.

(3) *Lines 16∼18*: For each line, the time complexity is primarily dependent on the time taken to solve the optimization problem in line 4 of Algorithm 2, which, being a convex optimization problem, has the same time complexity as problems $\mathbb{P}_{t,1}$ and $\mathbb{P}_{t,2}$.

In summary, the time complexity of Algorithm 1 is $O(N_d^2 N_c^{2.5} + N_c^{3.5}) + O(NJ)$. Moreover, since certain functions can run in parallel (e.g., lines 16, 17, and 18 in Algorithm 1), the actual running time of our algorithm is often acceptable in practice, which is verified in Section 6.3.

5. Performance analysis

Preliminaries: We first present some notations to illustrate our analysis. For our original problem $\mathbb{P}$ with objective function $\mathcal{P}$, we use $\widetilde{\boldsymbol{X}}$ to represent the optimum solutions and $\widetilde{\boldsymbol{X}}^*$ to denote the optimum fractional solutions. For our prediction problem $\widehat{\mathbb{P}}$ with objective function $\widehat{\mathcal{P}}$, we use $\bar{\boldsymbol{X}}_\vartriangle$ and $\widetilde{\boldsymbol{X}}_\vartriangle$ to represent the feasible integral solutions and corresponding fractional solutions without the rounding step, respectively. And we use $\widetilde{\boldsymbol{X}}_\vartriangle^*$ to denote the optimal fractional solutions for problem $\widehat{\mathbb{P}}$.

Assumptions: Before giving our theoretical analysis, we first introduce some assumptions which are common and easy to meet. Besides, they are widely adopted by similar problems [51–53].

- The gradients of $f_t(\widetilde{\boldsymbol{X}}_t)$ in Algorithm 2 can be bounded by $\|\nabla f_t(\boldsymbol{X}_t)\| \le \mathfrak{R}_f$, where $\mathfrak{R}_f$ is a constant.
- The radius of fractional domain $\tilde{\mathcal{X}}$ can be bounded by $\mathfrak{R}_r$ (i.e., $\|\widetilde{\boldsymbol{X}}_i - \widetilde{\boldsymbol{X}}_j\| \le \mathfrak{R}_r, \forall \widetilde{\boldsymbol{X}}_i, \widetilde{\boldsymbol{X}}_j \in \tilde{\mathcal{X}}$, where $\mathfrak{R}_r$ is a constant).

Then, we present our Proposition 1, Lemma 2∼3 and Theorem 1. Specifically, Proposition 1 is used for Algorithm 2. Lemma 2 is used for Algorithm 1 without the rounding step. Lemma 3 is used for Algorithm 3. Theorem 1 is used for our total prediction-based online approach, i.e., Algorithm 1.

Proposition 1. *When we take the step size $\lambda = \mathcal{O}(T^{-\epsilon_1})$ [8,51,54], the dynamic regret produced by Algorithm 2 has sub-linearly growth, i.e., $\sum_{t=1}^{T} \|\widehat{\sigma}_t - \sigma_t\|^2 g_t^2 = O(T^{\epsilon_2})$, where $\epsilon_1 \in (0,1)$ and $\epsilon_2 \in (0,1)$.*

Lemma 2. *Before the rounding step, Algorithm 1 achieves γ_1-competitive, i.e., $\widehat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\vartriangle) \le \gamma_1 \widehat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\vartriangle^*)$, where $\gamma_1 = \gamma_1'(1+\max\{\eta_1, \frac{1}{\eta_2}\})$. Parameters γ_1, γ_1', η_1 and η_2 are all constants.*

Proof. See Section 5.2 via the "laziness" parameters. □

Lemma 3. *After the rounding step, Algorithm 3 achieves the relationship $E[\widehat{\mathcal{P}}(\bar{\boldsymbol{X}}_\vartriangle)] \le \omega_1 \widehat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\vartriangle)$ between the integral solutions and fractional solutions, where ω_1 is a constant.*

Proof. See Section 5.3 via the expectation preservation. □

Theorem 1. *The competitive ratio ξ of total online algorithm is illustrated as follows:*

$$E[\mathcal{P}(\bar{\boldsymbol{X}}_\vartriangle)]/\mathcal{P}(\boldsymbol{X}^*) \le \omega_1\gamma_1 + O(T^{2(\epsilon_2-1)}) \triangleq \xi, \tag{9}$$

where $T^{2(\epsilon_2-1)}$, ω_1 and γ_1 are all constants, given T.

Proof. See Section 5.4, via Proposition 1, Lemma 2∼3. □

5.1. Proof of Lemma 1

Proof. We first prove the boundedness of $\sum_i y_{i,j,t}, \forall j, t$. Our randomized rounding mechanism employs a compensatory approach which randomly adds values to one variable and subtracts the same random value from another variable. Then, $\forall u, v \in \mathcal{J}$ we have

$$\bar{y}_{i,u,t} + \bar{y}_{i,v,t} \overset{OR}{=} \left\{ \begin{matrix} \tilde{y}_{i,u,t}+\theta_1 + \tilde{y}_{i,v,t}-\theta_1 \\ \tilde{y}_{i,u,t}-\theta_2 + \tilde{y}_{i,v,t}+\theta_2 \end{matrix} \right\} = \tilde{y}_{i,u,t}+\tilde{y}_{i,v,t}.$$

From lines 3∼9 of Algorithm 3, we have the equality:

$$\textstyle\sum_{j\in\tilde{\boldsymbol{\Omega}}_i\&|\tilde{\boldsymbol{\Omega}}_i|\ge 2} \bar{y}_{i,j,t} = \sum_{j\in\tilde{\boldsymbol{\Omega}}_i\&|\tilde{\boldsymbol{\Omega}}_i|\ge 2} \tilde{y}_{i,j,t}, \forall i, t. \tag{10}$$

Since that, the last $\tilde{y}_{i,j',t}$ (if it exists) is set to 1 as shown in line 10, we have the following inequality:

$$\textstyle\sum_{j'\in\tilde{\boldsymbol{\Omega}}_i\&|\tilde{\boldsymbol{\Omega}}_i|=1} \bar{y}_{i,j',t} \overset{OR}{=} \left\{ \begin{matrix} 1 \\ 0 \end{matrix} \right\} \le 1 + \sum_{j'\in\tilde{\boldsymbol{\Omega}}_i\&|\tilde{\boldsymbol{\Omega}}_i|=1} \tilde{y}_{i,j',t}, \forall i, t. \tag{11}$$

After adding Equality (10) to (11), we have the conclusion: $\sum_j \bar{y}_{i,j,t} \le \sum_j \tilde{y}_{i,j,t} + 1, \forall i, t$.

Then, we give the time complexity for our randomized rounding algorithm. It can be calculated by: (i) The outer loop (i.e., lines 1∼13 in Algorithm 3) has N iterations; (ii) For the inner loop (i.e., lines 3∼9): In each iteration, this algorithm typically rounds at least one real variable, causing it to either increase to 1 or decrease to 0. As shown in Step 3 of Fig. 5, the rounding results for every iteration within the inner loop can be categorized into four distinct types. Each of these result types is characterized by the presence of either 1 or 0. Then, after each iteration, such an integral solution's index will be removed from set $\tilde{\boldsymbol{\Omega}}_i$ defined in line 2. Thus, the inner loop has N iterations. The time complexity of Algorithm 3 is $\mathcal{O}(N * J)$. □

Table 3
Inference performance upon heterogeneous servers with GPUs.

Model type	Throughput			Inference latency (ms)			Average power (W)		
	RTX 2080Ti	3090	Tesla V100	2080Ti	3090	V100	2080Ti	3090	V100
YOLOv4 [55]	22	35	41	49	29	25	86	155	97
DeepLabv3 [56]	26	40	42	39	25	24	80	140	66
BERT [57]	30	62	66	33	16	15	65	131	47
VGG16 [58]	146	150	184	7	7	5	68	152	63
Inceptionv3 [59]	36	52	50	28	19	20	65	125	43

5.2. Proof of Lemma 2

Proof. We first build the connection between switching cost term ($C_S^v(\widetilde{\boldsymbol{X}}_\triangle)$) and non-switching cost item ($\hat{C}_{\neg S}^v(\widetilde{\boldsymbol{X}}_\triangle)$) via the "laziness" parameters η_1 and η_2. Then, we link the non-switching cost item ($\hat{C}_{\neg S}^v(\widetilde{\boldsymbol{X}}_\triangle)$) with its fractional optimum ($\hat{C}^v(\widetilde{\boldsymbol{X}}^*_\triangle)$) via the defined constant γ_1'.

For the switching cost incurred in the previous time slots, i.e., $C_S^{t'_u}(\widetilde{\boldsymbol{X}}_\triangle)$, where t'_u represents the timestamp when switching on servers, $1 \le u \le u'$, and u' represents the maximum record when incurring a switch. Then, the non-switching cost in the period $[t'_u, t'_{u+1}-1]$, is at least η_2 times the switching cost, where t represents the current time slot. Besides, the potential switching cost in $[t'_{u'}, t]$ is at most η_1 times the non-switching cost. Thus, $\forall t \le T$, we have

$$\sum_{v=1}^{t} C_S^v(\widetilde{\boldsymbol{X}}_\triangle) = \sum_{u\le u'}\sum_{v=t'_u}^{t'_{u+1}-1} C_S^v(\widetilde{\boldsymbol{X}}_\triangle) + \sum_{v=t'_{u'}+1}^{t} C_S^v(\widetilde{\boldsymbol{X}}_\triangle)$$
$$= \sum_{u\le u'}\{C_S^{t'_u}(\widetilde{\boldsymbol{X}}_\triangle) + \sum_{v=t'_u+1}^{t'_{u+1}-1} 0\} + \{C_S^{t'_{u'}}(\widetilde{\boldsymbol{X}}_\triangle) + \sum_{v=t'_{u'}+1}^{t} 0\}$$
$$\le \sum_{u\le u'}\{\frac{1}{\eta_2}\sum_{v=t'_u}^{t'_{u+1}-1} \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle) + 0\} + \{\eta_1 \hat{C}^{t'_{u'}}_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle) + 0\}$$
$$\le \max\{\eta_1, 1/\eta_2\}\sum_{v=1}^{t} \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle). \tag{12}$$

Then, we take $\gamma_1' \triangleq \max_v \frac{\max_{\widetilde{X}} \hat{C}^v_{\neg S}(\widetilde{X})}{\min_{\widetilde{X}} \hat{C}^v_{\neg S}(\widetilde{X})}$, and have

$$\hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle) \le \gamma_1' \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}^*_\triangle), \forall v \le T. \tag{13}$$

After that, $\forall t \le T$, we have the following inequality:

$$\sum_{v=1}^{t} \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle) \le \gamma_1' \sum_{v=1}^{t} \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}^*_\triangle) \le$$
$$\gamma_1' \sum_{v=1}^{t}\{C^v_S(\widetilde{\boldsymbol{X}}^*_\triangle) + \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}^*_\triangle)\} \le \gamma_1' \sum_{v=1}^{t} \hat{C}^v(\widetilde{\boldsymbol{X}}^*_\triangle). \tag{14}$$

Therefore, the overall cost of Algorithm 1 without the rounding step under the prediction could be bounded by

$$\sum_{v=1}^{T}[C^v_S(\widetilde{\boldsymbol{X}}_\triangle) + \hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle)] \le (1+\max\{\eta_1, \frac{1}{\eta_2}\})\sum_{v=1}^{T}\hat{C}^v_{\neg S}(\widetilde{\boldsymbol{X}}_\triangle)$$
$$\le \gamma_1'(1+\max\{\eta_1, \frac{1}{\eta_2}\})\sum_{v=1}^{T}\hat{C}^v(\widetilde{\boldsymbol{X}}^*_\triangle) \triangleq \gamma_1 \sum_{v=1}^{T}\hat{C}^v(\widetilde{\boldsymbol{X}}^*_\triangle). \quad \square$$

5.3. Proof of Lemma 3

Proof. When problem $\mathbb{P}_{t,1}$ is solvable, the rounding algorithm is just triggered; Otherwise, the rounding results will be dropped and use the decisions of the previous time slot. When $\mathbb{P}_{t,1}$ is invoked in line 5 of Algorithm 1, we use $\widetilde{y}_{i,j,t}, \widetilde{z}_{i',i,t}$ to represent the fractional solutions. We further use $y_{i,j,t}, z_{i',i,t}$ to denote the integral solution after the rounding step (i.e., line 5 of Algorithm 1). Note that, decision variable z_t is in the fraction domain which does not require rounding to be integers.

First, we link the rounding fractional solutions and integral solutions. We use index $u : 1 \le u \le u'$ to record the timestamp regarding the server switches. Then, we have the following inequalities:

$$\sum_{i,j,t} y_{i,j,t} = \sum_{u\le u'}\{(t'_{u+1} - t'_u)(\sum_{i,j} y_{i,j,t'_u})\}$$
$$\overset{((15)a)}{\le} \sum_{u\le u'}\{(t'_{u+1} - t'_u)(\sum_{i,j} \widetilde{y}_{i,j,t'_u} + 1)\}$$
$$\overset{((15)b)}{=} \sum_{u\le u'}\{(t'_{u+1} - t'_u)(\sum_{i,j} \widetilde{y}_{i,j,t'_u} + \frac{\sum_{i'}\sum_i \widetilde{z}_{i',i,t'_u}}{|\mathcal{N}|})\}$$
$$\overset{((15)c)}{\le} \sum_{u\le u'}\{(t'_{u+1} - t'_u)(\sum_{i,j} \widetilde{y}_{i,j,t'_u} + \frac{\sum_{i,j} y_{i,j,t'_u} M_i}{|\mathcal{N}|})\}$$
$$\le (1 + M_i/|\mathcal{N}|)\sum_{i,j,t}\widetilde{y}_{t,j} \triangleq \kappa_0 \sum_{i,j,t}\widetilde{y}_{t,j}, \tag{15}$$

where Inequality ((15)a) holds due to our rounding strategy ensures $\forall j, \sum_i y_{i,j,t} \le \sum_i \widetilde{y}_{i,j,t} + 1$ (i.e., Lemma 1). Inequality ((15)b) holds since $\sum_i \widetilde{z}_{i',i,t'_u} = 1$ in Constraint (1). Note that, problem $\mathbb{P}_{t,1}$ also includes Constraint (1); Inequality ((15)c) holds due to Constraint (2); Finally, κ_0 is a constant.

Then, we consider all of the four terms in the objective of $\mathbb{P}_{t,1}$ (i.e., $\hat{C}^t_{\neg S}(\cdot)$). For $\sum_i\sum_j\sum_t y_{i,j,t}\hat{o}_{i,j,t}$, we have

$$\sum_t\sum_i\sum_j y_{i,j,t}\hat{o}_{i,j,t} \overset{((16)a)}{\le} \kappa_0 o_{max}\sum_{t,j,i}\widetilde{y}_{i,j,t} \triangleq \kappa_1 \sum_{t,j,i}\widetilde{y}_{i,j,t}, \tag{16}$$

where Inequality ((16)a) holds due to Inequality (15) and $o_{max} \triangleq \max_{i,j,t}\{o_{i,j,t}\}$.

For the second term $\sum_{i,i'} z_{i',i,t}\hat{b}_{i',i,t}r_{i',t}$, we have

$$\sum_t\sum_{i,i'} z_{i',i,t}\hat{b}_{i',i,t}r_{i',t} \le \sum_{t,j,i}\widetilde{y}_{i,j,t}M \triangleq \kappa_2\sum_{t,j,i}\widetilde{y}_{i,j,t}. \tag{17}$$

where Inequality ((17)a) holds since Constraint (2).

For the third term in the objective function, we have

$$\sum_{t,i}[q_{i,t-1} + \sum_{i'} r_{i',t}z_{i',i,t} - \sum_j p_j y_{i,j,t}]^+$$
$$\le \sum_t\sum_i\sum_j \widetilde{y}_{i,j,t}M \triangleq \kappa_2 \sum_{t,j,i}\widetilde{y}_{i,j,t}. \tag{18}$$

For the fourth term in the objective function, we have

$$\sum_{i,j,t}(1-\hat{\iota}_{j,t})y_{i,j,t} \le \sum_{i,j,t} y_{i,j,t} \overset{((19)a)}{\le} \kappa_0 \sum_{t,j,i}\widetilde{y}_{i,j,t}, \tag{19}$$

where ((19)a) holds since Inequality (15). Then, based on Equality (16)∼(19), the sum of objective function of $\mathbb{P}_{t,1}$ is

$$\sum_t \hat{C}^t_S(\bar{\boldsymbol{X}}_\triangle) = \sum_{i,j,t} y_{i,j,t}\hat{o}_{i,j,t} + \sum_{i,i',t} z_{i',i,t}\hat{b}_{i',i,t}r_{i',t}$$
$$+ \sum_{i,t} q_{i,t} + \sum_{i,j,t}(1-\hat{\iota}_{j,t})y_{i,j,t} \le (\kappa_0 + \kappa_1 + 2\kappa_2)\sum_{i,j,t}\widetilde{y}_{i,j,t}. \tag{20}$$

Then, we have the following inequality

$$\hat{\mathcal{P}}(\bar{\boldsymbol{X}}_\triangle) = \sum_t(\hat{C}^t_{\neg S} + \hat{C}^t_S) \overset{((21)a)}{\le} (1 + 1/\eta_1)\sum_t \hat{C}^t_S(\bar{\boldsymbol{X}}_\triangle)$$
$$\overset{((21)b)}{\le} (1 + 1/\eta_1)(\kappa_0 + \kappa_1 + 2\kappa_2)\sum_{t,j,i}\widetilde{y}_{i,j,t}$$

Table 4
Model and dataset.

YOLOv4	DeepLabv3	BERT	VGG16	Inceptionv3
VOC07+12 CV	VOC-Val12 CV	THUCNews NLP	CIFAR10 CV	CIFAR10 CV

$$\overset{((21)c)}{\leq} (1+1/\eta_1)(\kappa_0+\kappa_1+2\kappa_2)\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle) \triangleq \omega_1 \hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle), \tag{21}$$

where ((21)*a*) holds since Constraint (4); ((21)*b*) holds since Inequality (20); ((21)*c*) holds since $\sum_{t,j,i} \tilde{y}_{i,j,t}$ is a part of objective function $\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)$. We have Lemma 3, where $\omega_1 \triangleq (1+1/\eta_1)(\kappa_0+\kappa_1+2\kappa_2)$ is a constant. □

5.4. Proof of Theorem 1

Proof. From Proposition 1, we can link the real objective function $\mathcal{P}(\bar{\boldsymbol{X}}_\triangle)$ with the predicted function $\hat{\mathcal{P}}(\bar{\boldsymbol{X}}_\triangle)$, via

$$\mathcal{P}(\bar{\boldsymbol{X}}_\triangle) = \hat{\mathcal{P}}(\bar{\boldsymbol{X}}_\triangle) + O(T^{\epsilon_2}). \tag{22}$$

Then, we introduce the following equation, where on the right-hand side, there exists a product of two terms:

$$\frac{E[\mathcal{P}(\bar{\boldsymbol{X}}_\triangle)]}{\mathcal{P}^*} = \frac{E[\mathcal{P}(\bar{\boldsymbol{X}}_\triangle)]}{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)} * \frac{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)}{\mathcal{P}^*}. \tag{23}$$

Then, for the first term on the right-hand side of Equality (23), we have the following inequalities:

$$\begin{aligned}\frac{E[\mathcal{P}(\bar{\boldsymbol{X}}_\triangle)]}{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)} &\overset{((24)a)}{=} \frac{E[\hat{\mathcal{P}}(\bar{\boldsymbol{X}}_\triangle)+O(T^{\epsilon_2})]}{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)} \overset{((24)b)}{\leq} \frac{\omega_1\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)+O(T^{\epsilon_2})}{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)}\\ &\overset{((24)c)}{\leq} \omega_1 + \frac{O(T^{\epsilon_2})}{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}^*_\triangle)} \leq \omega_1 + \frac{O(T^{\epsilon_2})}{T * \Lambda_1} = \omega_1 + O(T^{\epsilon_2-1}),\end{aligned} \tag{24}$$

where ((24)a) holds since Equality (22); ((24)b) holds since Equality (21) in Lemma 3; ((24)c) holds since the optimum $\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}^*_\triangle)$ is less than $\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)$; Constant Λ_1 represents the lower bound cost in each time slot.

For the second term on the right-hand side of Equality (23), we also have the following inequalities:

$$\begin{aligned}\frac{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle)}{\mathcal{P}^*} &= 1 + \frac{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle) - \mathcal{P}(\widetilde{\boldsymbol{X}}^*) + \mathcal{P}(\widetilde{\boldsymbol{X}}^*) - \mathcal{P}^*}{\mathcal{P}^*}\\ &\overset{((25)a)}{\leq} 1 + \frac{\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}_\triangle) - \mathcal{P}(\widetilde{\boldsymbol{X}}^*)}{\mathcal{P}^*} \overset{((25)b)}{\leq} 1 + \frac{\gamma_1\hat{\mathcal{P}}(\widetilde{\boldsymbol{X}}^*) - \mathcal{P}(\widetilde{\boldsymbol{X}}^*)}{\mathcal{P}^*}\\ &\overset{((25)c)}{\leq} 1 + \frac{\gamma_1\mathcal{P}(\widetilde{\boldsymbol{X}}^*) + O(T^{\epsilon_2}) - \mathcal{P}(\widetilde{\boldsymbol{X}}^*)}{\mathcal{P}^*} = 1 + \frac{(\gamma_1-1)\mathcal{P}(\widetilde{\boldsymbol{X}}^*) + O(T^{\epsilon_2})}{\mathcal{P}^*}\\ &\overset{((25)d)}{\leq} 1+\gamma_1-1+\frac{O(T^{\epsilon_2})}{\mathcal{P}^*} \overset{((25)e)}{\leq} \gamma_1 + \frac{O(T^{\epsilon_2})}{T * \Lambda_1} = \gamma_1 + O(T^{\epsilon_2-1}),\end{aligned} \tag{25}$$

where ((25)a) holds since $\mathcal{P}(\widetilde{\boldsymbol{X}}^*) - \mathcal{P}^* \leq 0$; ((25)b) holds since Lemma 2; ((25)c) holds since Equality (22); ((25)d) holds since $\frac{\mathcal{P}(\widetilde{\boldsymbol{X}}^*)}{\mathcal{P}^*} \leq 1$; ((25)e) holds for the same reason with above Inequality ((24)c) (i.e., Λ_1 represents the lower bound).

Combining Equality (23) and Inequality (24)~(25), we have the competitive ratio as follows:

$$\frac{E[\mathcal{P}(\bar{\boldsymbol{X}}_\triangle)]}{\mathcal{P}^*} = \omega_1\gamma_1 + O(T^{2(\epsilon_2-1)}). \quad \square \tag{26}$$

6. Experiment

6.1. Testbed and results

As shown in Fig. 7, we build a testbed to evaluate the performance of heterogeneous resources for inference tasks in computing power network (CPN). Specifically, our testbed consists of five real devices (two types): servers equipped with GPUs (i.e., GeForce RTX 2080Ti, GeForce RTX 3090, and Nvidia Tesla V100) and edge devices (i.e., Huawei Atals 200DK and Nvidia Jetson NX). Then, we select five widely used inference models from the fields of computer vision and natural language processing (i.e., YOLOv4 [55], DeepLabv3 [56], BERT [57], VGG16 [58] and Inceptionv3 [59]). We test the inference performance (e.g., throughput, inference latency, and energy consumption) of five models on these two types of devices.

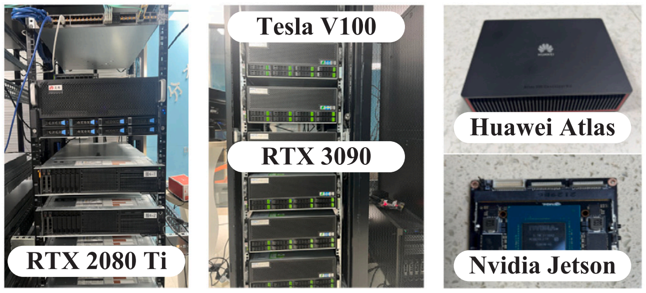


Fig. 7. Real device in our testbed.

Testbed Configurations: We select diverse datasets (e.g., VOC [60], THUCNews [61], and CIFAR-10 [62]) to assess the inference performance of these models. The specific correspondences are outlined in Table 4. Suitable batch and chunk sizes are configured for these inference tasks. For each dataset, we conducted repetitions ranging from 100 to 500 times, and the average results are presented in both Tables 3 and 5.

Results on GPU Servers: Table 3 shows the inference performance of five models on three different GPUs, including throughput, inference latency, and average GPU power. We conclude that for the same model, there is a noticeable disparity in throughput/latency across different hardware configurations. For instance, YOLOv4 exhibits varying throughput and latency on the three hardware configurations. Additionally, there is a significant difference in power across different hardware, with power consumption ranging from 65w to 86w for 2080Ti, 125w to 155w for 3090, and 43w to 97w for Tesla V100.

Results on Edge Devices: From Table 5, it can be observed that the inference latency on the Nvidia Jetson NX is lower than that on the Huawei Atlas 200DK. This difference is particularly obvious for computationally intensive models (e.g., DeepLabv3). This difference is mainly caused by the heterogeneity of the hardware. Specifically, in Fig. 8, we depict the distribution of inference latency for multiple executions of the same inference task on Jetson NX. We obtain that, for a specific model, the inference latency exhibits a centralized distribution pattern. Therefore, in our experimental evaluations, we use the average as a replacement for the inference latency. Besides, we also observe that the power of 200DK AI Cores remains constant (i.e., 12.8 W) regardless of the model being run and the power fluctuations of NX are minor (less than 8 W), even varying GPU utilization from 0% to 99%.

Preliminary Conclusion: The data obtained from the testbed will be applied in our subsequent experimental evaluations. Besides, we have the following conclusions: (1) The inference latency for the same model is significantly different across different types of devices. For instance, the inference latency on GPU servers (20 ms~30 ms) is superior to that on edge devices (100 ms~300 ms). (2) The inference latency for the different models on the same device is also different due to differences in computational complexity. Particularly, on low-performance edge devices, the inference latency tends to be longer.

6.2. Simulation setting

Our experimental environment is shown in Fig. 7. And we conduct the evaluation for our algorithms by using the data from our testbed above.

Table 5
Inference performance upon edge devices.

Device	Model	Y.4t	D.3	B.T	In.v3
Atlas	Latency	111	311	304	230
	TP.	9	3	3	4
Jetson	Latency	110	191	212	198
	TP.	10	5	5	5

Y.4t is YOLOv4-tiny; D.3 is DeepLabv3, B.T is BERT;
In.v3 denotes Inceptionv3 and TP. is throughput;
Power of Atlas is 12.9 W; Power of Jetson is in [1 W, 9 W].

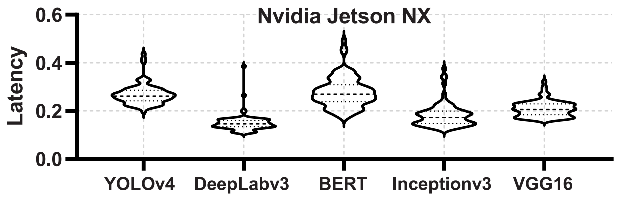


Fig. 8. Results upon edge devices in our testbed.

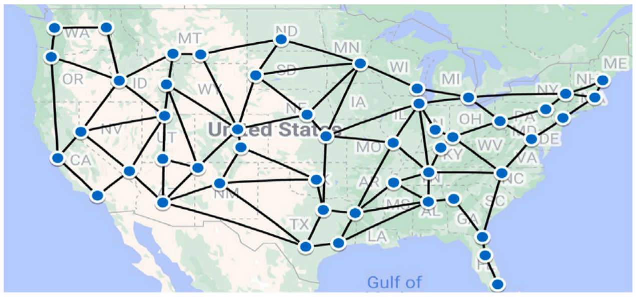
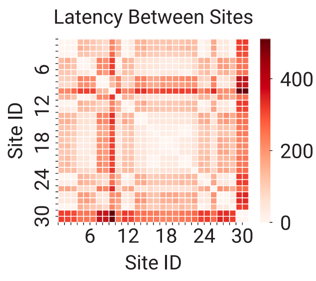
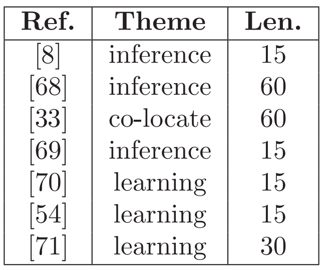


Fig. 9. Computing power network topology.



(a) Latency



Ref.	Theme	Len.
[8]	inference	15
[68]	inference	60
[33]	co-locate	60
[69]	inference	15
[70]	learning	15
[54]	learning	15
[71]	learning	30

(b) Time Window Length

Fig. 10. Motivation study (see Refs. [8,33,54,68–71]).

Network Topology, Link, Computing Site Server: Based on the realistic US backbone network topology, named *janos-us-ca*, from the SNDlib project [63], we build our network topology by using NetworkX [64], which consists of fifty computing sites and eighty-six links as shown in Fig. 9. The latency of each link is obtained from [65], which is collected from two public network platforms (i.e., PlanetLab [66] and Seattle [67]). As shown in Fig. 10, we use a heatmap to illustrate the latency between some sites. Each computing site is equipped with 20 different types of servers for various inference workloads. We use the parameters of real virtual machine instances from Alibaba Cloud [41], including, *ecs.c7.large*~*ecs.c7.32xlarge*, where the number of *vcpu* is from set from 1 to 128 and the size of *Mem* is set from 0.5 GiB to 258 GiB.

Inference Workload, Processing and Resource Cost: We use the latency of the inference model (i.e., YOLOv4 [55], DeepLabv3 [56], BERT [57], VGG16 [58] and Inceptionv3 [59]) from our testbed mentioned before. The inference workloads of each computing site are obtained from passenger data of the 268 London underground stations [72] which is collected in 15-minute intervals over a four-day period on November 16, 2016, resulting in a total of about 300 time slots. We calculate the average number of passengers at these stations and classify them into three categories: Low (<150 passengers), Medium (150~250 passengers), and High (250~500 passengers). Then, we assume that each passenger emits 5~45 inference queries to his corresponding computing site. The computing capacity for each instance in the computing site is randomly configured as 15~300 [8,73]. The operational cost for these virtual machine instances is set from 0.063\$/hour to 26.092733\$/hour which are the real prices obtained from [74]. The electricity price data are from CAISO [18], and the operational cost of servers at a specific site fluctuates with its local electricity price. The unit switching cost is twice the operational cost [71].

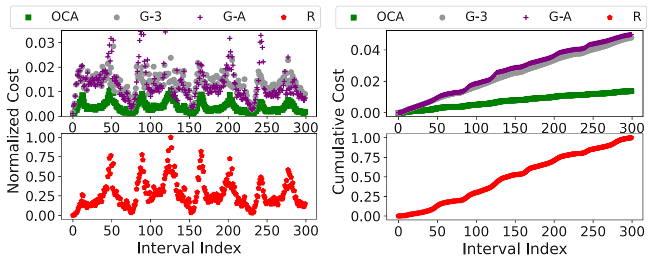


Fig. 11. (a) Real-time total cost and (b) Cumulative cost.

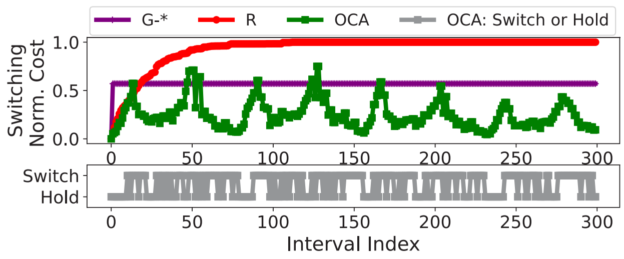


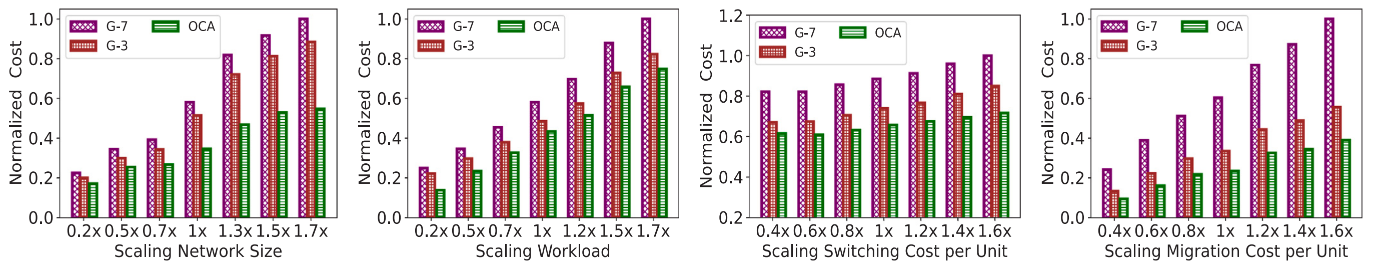
Fig. 12. Real-time switching cost and decisions.

Other Parameters: We consider 300 time slots where each time slot consists of 15 min. Then, step size λ of Algorithm 2 is set as 0.15 $\approx \mathcal{O}(300^{-\frac{1}{3}})$ [8]. In Algorithm 1, we set parameters $\eta_1 = 1/2$ and $\eta_2 = 2$ which are derived from previous work [75]. Our time window length is selected based on a survey of recent research about scheduling for AI workloads. As shown in Fig. 10(b), the time windows used in these works range in [15, 60] minutes. We set the time window to 15 minutes, as it provides a smaller and more granular time window.

Algorithms and Metrics: For **CPN-Inference Controller**, our proposed approach is called *OCA* which uses lazy switch and online learning prediction (via CVXPY [46] to solve the subproblems). We implement various algorithm combinations for comparison.[3] For the prediction of network latency, the algorithms are as follows:

- $*$-*3* indicates predictions made based on the average inputs from the previous three time slots.
- $*$-*7* indicates predictions made based on the average inputs from the previous seven time slots.
- $*$-*A* indicates predictions made based on the average inputs from the previous all-time slots.

[3] The mark of * is a wildcard to match different prediction algorithms and subproblems solving algorithms.



(a) Impact of Network Size (b) Impact of Workload (c) Impact of Switching Cost (d) Impact of Migration Cost

Fig. 13. Further results of our proposed *OCA*.

For solving subproblems, we consider algorithms as follows:

- *G-$*$* indicates the general approach by using solver [46] without considering controlling the switching cost item and non-switching cost item.
- *R* indicates the random approach by setting decisions variables from $[0,1]$, randomly, satisfying the constraints.

Besides, we implement the state-of-the-art (*SOTA*) algorithm from work [33] for comparison. Specifically, this study [33] employs a modified online saddle-point method for edge resource scheduling to reduce the system cost. Subsequently, we observe the cumulative system cost incurred by SOTA and our proposed OCA algorithm.

6.3. Experimental results

Fig. 11(a) initially depicts the real-time total cost for each time slot. Our *OCA* consistently outperforms other algorithms, leading to an average cost reduction of 70%. When compared to random algorithms, denoted as *R*, our *OCA* achieves a significant reduction in total cost. Specifically, the total real-time cost consists of operational cost, communication cost, etc. Some of them include time-varying environmental parameters (e.g., $\aleph_{i,t}$, $b_{i',i,t}$), whose values are taken from real-world datasets [18,67]. Figs. 2 and 10 show some samples from these datasets. It is clear that these parameters fluctuate over time, leading to the dynamic changes in the system cost seen in Fig. 11(a). Moreover, as observed in Fig. 11(a), although the total cost produced by each algorithm differs, their overall fluctuation trends (e.g., the time slots of peaks and troughs) are similar. This further emphasizes the significant impact of environmental parameters on the system cost. Nonetheless, our designed prediction algorithm with theoretical bounds accurately tracks the trends of these posterior parameters. In contrast, other algorithms (e.g., *$*$-3*, *$*$-7*, and *$*$-A*) use prediction mechanisms without guarantees, which fail to capture these parameter fluctuations effectively. Consequently, the decisions from such algorithms tend to deviate significantly from the optimum.

Fig. 11(b) extends this analysis by visualizing the normalized cumulative cost for various algorithms over consecutive time slots. We note that the cumulative cost for our *OCA* grows more slowly than for the other four algorithms. Specifically, our OCA algorithm uses a theoretically bounded decoupling method, which dynamically balances the switching cost with the other terms to minimize long-term cost. In contrast, other algorithms, such as *G-$*$* or *R*, do not account for switching cost, resulting in significantly higher system cost over time.

Fig. 12 shows the switching cost (above) separately and the resource provisioning decisions on servers (below). Our *OCA* often outperforms *G-$*$* and *R*, in terms of switching cost, achieving an overall reduction of 55% and 72%, respectively. The figure also reveals that algorithm *R* has the highest switching cost since such an algorithm often neglects time coupling between adjacent decisions on resource provisioning. Additionally, we illustrate the decisions on resource provisioning (below). We find that our *OCA* dynamic switches on/off the servers according

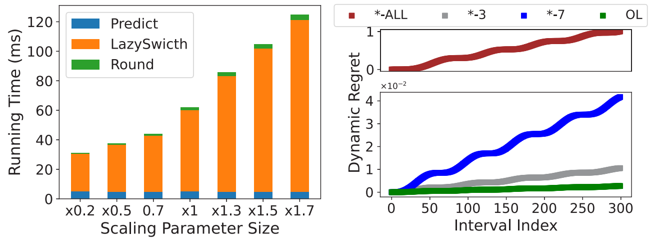


Fig. 14. (a) Running time and (b) Dynamic regret.

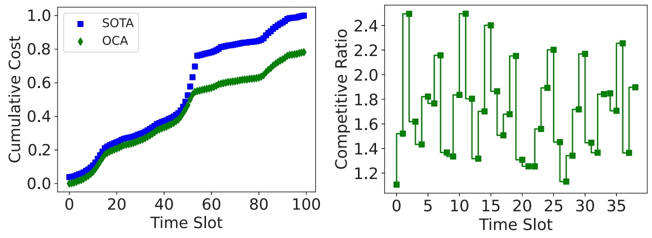


Fig. 15. (a) Comparison with SOTA and (b) Competitive ratio.

to the status of the system, which ensures a low switching cost in the long term.

Fig. 13 illustrates the cumulative total cost throughout the entire time horizon for various algorithms. In Fig. 13(a), the cumulative cost change along the variation of network size. Note that, in the small network topology (i.e., 10 computing sites), the average cost reduction of *OCA* is only 14% and 23%, compared to *G-3* and *G-7*, respectively. With the growth of network size, our *OCA* shows better performance and the average cost reduction is about 21.2%. Fig. 13(b) shows the various results as the inference query workloads for each user vary from 5 to 45. Our *OCA* consistently has stable performance, achieving an average cost reduction of 19%. Specifically, the 0.2x∼1.7x in Fig. 13(b) refer to the following number of queries for each user: 5∼10, 10∼15, 15∼20, 20∼25, 25∼30, 30∼40, and 35∼45, respectively. In Fig. 13(c), the cumulative cost rises as the unit of switching cost increases. When compared to *G-$*$* algorithms, our lazy switch based algorithm performs the best, particularly when the unit switching cost is high. *OCA* achieves a cost reduction ranging from 8% to 28%, with an average reduction of 12%. In Fig. 13(d), the cumulative cost notably increases as the unit latency cost grows. Notably, *G-3* algorithm tends to be advantageous when the unit latency cost is low. Overall, compared to other algorithms, *OCA* improves performance by an average of 35%.

Fig. 14(a) illustrates the dissection running time of our proposed *OCA* which finishes within tens of milliseconds achieving an average of 70 ms. Specifically, the 0.2x∼1.7x in the x-axis of Fig. 14(a) are 500, 825, 1200, 2100, 3200, 3825, 4500, respectively. Fig. 14(b) further presents the dynamic regret of the prediction on network latency and electricity prices, where the cost grows sub-linearly along with time, and we also observe the dynamic regret of our Algorithm 2, named *OL*, has the best performance compared to other methods.

As shown in Fig. 15(a), the cumulative system cost generated by our proposed OCA algorithm and SOTA are compared. Since our OCA takes both switching cost and communication cost into account, it achieves lower overall system cost. Overall, our OCA algorithm outperforms SOTA, achieving a 10%∼20% improvement in system performance, demonstrating its practical advantages in real-world scenarios. Furthermore, we make a clear distinction between the ''theoretical competitive ratio'' and the ''empirical competitive ratio''. The former refers to the worst-case performance, as proven in Section 5, whereas the latter reflects the model's practical performance in real-world settings. Importantly, a large theoretical competitive ratio does not necessarily imply that the empirical ratio will also be high. In fact, as illustrated in Fig. 15(b), our experiments demonstrate that the empirical competitive ratio remains between 1.1 and 2.5 across different time slots, which is a reasonable and manageable result in practice.

7. Conclusion

Scheduling inference tasks at computing power network (CPN) in an online manner is a nontrivial problem. In this work, we advocate CPN-Inference, a flexible inference service framework upon CPN, and model an online time-varying non-linear integer programming problem. This formulation takes into account various factors including switching cost, operational cost, query communication cost, queuing cost (QoS), and accuracy loss. To tackle this problem, we design a group of online polynomial-time algorithms containing a lazy-switch control approach for efficient resource management, an online learning method for accurate and dynamic prediction, and a randomized rounding mechanism for fractional decision deployment. We rigorously analyze the performance of our control approach, demonstrating its competitive ratio in comparison to the offline optimum. Additionally, we show that the dynamic regret for our online learning method grows sub-linearly along with time. Through comprehensive evaluations using real-world datasets, our evaluations of real-world data demonstrate the advantages of our approach, achieving an average cost reduction of 35% compared to other algorithms. Although our mechanism focused on single-type applications (e.g., either object detection or image classification), it can be easily extended to handle multiple types of applications after reusing the mechanism. Moving forward, we plan to explore the optimization of hybrid application scenarios, expanding the system's capability to support diverse and mixed application tasks.

CRediT authorship contribution statement

Mingtao Ji: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Ji Qi:** Supervision, Investigation, Funding acquisition, Data curation. **Lei Jiao:** Formal analysis, Funding acquisition, Investigation, Project administration. **Gangyi Luo:** Methodology, Investigation. **Hehan Zhao:** Data curation. **Xin Li:** Supervision, Resources, Investigation. **Baoliu Ye:** Supervision. **Zhuzhong Qian:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute, and by the U.S. National Science Foundation (CNS-2047719 and CNS-2225949).

Data availability

Data will be made available on request.

References

[1] Network 2030, 2023, https://www.itu.int/dms_pub/itu-t/opb/fg/T-FG-NET2030-2020-SUB.G1-PDF-E.pdf.

[2] Y. Sun, B. Lei, J. Liu, H. Huang, X. Zhang, J. Peng, W. Wang, Computing power network: A survey, 2022, arXiv:2210.06080.

[3] L. Bo, L. Zengyi, W. Xuliang, Y. Mingchuan, C. Yunqing, Computing network: a new multi-access edge computing, Telecommun. Sci. (2019).

[4] L. Geng, P. Willis, Compute First Networking (CFN) Scenarios and Requirements, Internet-Draft draft-geng-rtgwg-cfn-req-00, Internet Engineering Task Force, 2019, Work in Progress, URL https://datatracker.ietf.org/doc/draft-geng-rtgwg-cfn-req/00/.

[5] Computing power network - Framework and architecture, 2021, https://www.itu.int/rec/T-REC-Y.2501-202109-I.

[6] Eastern Data and Western Computing: Building New Computing-first Networks, 2022, https://www.huawei.com/en/huaweitech/publication/202202/eastern-data-western-computing-network.

[7] Sky Computing Towards Utility Computing for the Cloud, 2023, https://sky.cs.berkeley.edu/.

[8] Y. Jin, L. Jiao, Z. Qian, S. Zhang, N. Chen, S. Lu, X. Wang, Provisioning edge inference as a service via online learning, in: IEEE SECON, 2020, pp. 1–9, http://dx.doi.org/10.1109/SECON48991.2020.9158425.

[9] H. Sun, X. Chen, Z. Qian, Z. Li, N. Chen, T. Cao, S. Xu, Y. Zhou, BIRP: Batch-aware inference workload redistribution and parallel scheme for edge collaboration, in: Proceedings of the 52nd International Conference on Parallel Processing, ICPP '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 72–81, http://dx.doi.org/10.1145/3605573.3605615.

[10] T. Yongqiang, s. Yanxia, Y. Xinyan, L. Xin, Day-ahead electricity price forecasting employing a novel hybrid frame of deep learning methods: A case study in NSW, Australia, Electr. Power Syst. Res. 220 (2023) 109300, http://dx.doi.org/10.1016/j.epsr.2023.109300, URL https://www.sciencedirect.com/science/article/pii/S037877962300189X.

[11] Y. Li, Z. Han, Q. Zhang, Z. Li, H. Tan, Automating cloud deployment for deep learning inference of real-time online services, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1668–1677, http://dx.doi.org/10.1109/INFOCOM41043.2020.9155267.

[12] Y. Li, Z. Li, Z. Han, Q. Zhang, X. Ma, Automating cloud deployment for real-time online foundation model inference, IEEE/ACM Trans. Netw. 32 (2) (2024) 1509–1523, http://dx.doi.org/10.1109/TNET.2023.3321967.

[13] I. Wohlgenannt, A. Simons, S. Stieglitz, Virtual reality, Bus. Inf. Syst. Eng. 62 (2020) 455–461.

[14] F. Arena, M. Collotta, G. Pau, F. Termine, An overview of augmented reality, Computers 11 (2) (2022) http://dx.doi.org/10.3390/computers11020028, URL https://www.mdpi.com/2073-431X/11/2/28.

[15] X. Yuan, L. Pu, L. Jiao, X. Wang, M. Yang, J. Xu, When computing power network meets distributed machine learning: An efficient federated split learning framework, 2023, arXiv:2305.12979.

[16] K. Peng, X. Wang, T. Qinin, Exploration and practice of collaborative scheduling of computing power network resources, ZTE-Commun. (2023) 1–1.

[17] M. Ji, Y. Jin, Z. Qian, T. Cao, B. Ye, Orchestrating in-network aggregation for distributed machine learning via in-band network telemetry, J. Comput. Sci. Tech..

[18] California ISO, 2023, https://www.caiso.com/.

[19] SOHO NEWS, 2023, https://www.sohu.com/a/724620715_121123908.

[20] R. Desislavov, F. Martínez-Plumed, J. Hernández-Orallo, Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning, Sustain. Comput.: Inform. Syst. 38 (2023) 100857, http://dx.doi.org/10.1016/j.suscom.2023.100857, URL https://www.sciencedirect.com/science/article/pii/S2210537923000124.

[21] X. Tang, C. Cao, Y. Wang, S. Zhang, Y. Liu, M. Li, T. He, Computing power network: The architecture of convergence of computing and networking towards 6G requirement, China Commun. 18 (2) (2021) 175–185, http://dx.doi.org/10.23919/JCC.2021.02.011.

[22] H. Yin, X. Zhang, S. Zhao, Y. Luo, C. Tian, V. Sekar, Tradeoffs between cost and performance for CDN provisioning based on coordinate transformation, IEEE Trans. Multimed. 19 (11) (2017) 2583–2596, http://dx.doi.org/10.1109/TMM.2017.2696309.

[23] V.G. Douros, S.E. Elayoubi, E. Altman, Y. Hayel, Caching games between content providers and internet service providers, Perform. Eval. 113 (2017) 13–25, http://dx.doi.org/10.1016/j.peva.2017.04.006.

[24] Y. Zhao, C. Tian, J. Fan, T. Guan, X. Zhang, C. Qiao, Joint reducer placement and coflow bandwidth scheduling for computing clusters, IEEE/ACM Trans. Netw. 29 (1) (2021) 438–451, http://dx.doi.org/10.1109/TNET.2020.3037064.

[25] X. Zhou, X. Dong, L. Zhao, K. Li, T. Qiu, Learning-driven cloud resource provision policy for content providers with competitor, IEEE Trans. Cloud Comput. 10 (2022) 1913–1924, URL https://api.semanticscholar.org/CorpusID:226422845.

[26] Y. Bai, L. Chen, M. Abdel-Mottaleb, J. Xu, Automated ensemble for deep learning inference on edge computing platforms, IEEE Internet Things J. 9 (6) (2022) 4202–4213, http://dx.doi.org/10.1109/JIOT.2021.3102945.

[27] Y. She, M. Li, Y. Jin, M. Xu, J. Wang, B. Liu, On-demand edge inference scheduling with accuracy and deadline guarantee, in: 2023 IEEE/ACM 31st International Symposium on Quality of Service, IWQoS, 2023, pp. 1–10, http://dx.doi.org/10.1109/IWQoS57198.2023.10188769.

[28] F. Xu, J. Xu, J. Chen, L. Chen, R. Shang, Z. Zhou, F. Liu, iGniter: Interference-aware GPU resource provisioning for predictable DNN inference in the cloud, IEEE Trans. Parallel Distrib. Syst. 34 (3) (2023) 812–827, http://dx.doi.org/10.1109/TPDS.2022.3232715.

[29] E. Song, T. Pan, C. Jia, W. Cao, J. Zhang, T. Huang, Y. Liu, INT-label: Lightweight in-band network-wide telemetry via interval-based distributed labelling, in: IEEE INFOCOM, 2021, pp. 1–10.

[30] J. Liu, Y. Sun, J. Su, Z. Li, X. Zhang, B. Lei, W. Wang, Computing power network: A testbed and applications with edge intelligence, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2022, pp. 1–2, http://dx.doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798112.

[31] X. Yuan, L. Pu, L. Jiao, X. Wang, M. Yang, J. Xu, When computing power network meets distributed machine learning: An efficient federated split learning framework, in: 2023 IEEE/ACM 31st International Symposium on Quality of Service, IWQoS, 2023, pp. 1–10, http://dx.doi.org/10.1109/IWQoS57198.2023.10188789.

[32] W.C. Ng, W.Y.B. Lim, Z. Xiong, D. Niyato, H.V. Poor, X.S. Shen, C. Miao, Stochastic resource optimization for wireless powered hybrid coded edge computing networks, IEEE Trans. Mob. Comput. 23 (3) (2024) 2022–2038, http://dx.doi.org/10.1109/TMC.2023.3246994.

[33] S. Su, Z. Zhou, T. Ouyang, R. Zhou, X. Chen, Learning to be green: Carbon-aware online control for edge intelligence with colocated learning and inference, in: 2023 IEEE 43rd International Conference on Distributed Computing Systems, ICDCS, 2023, pp. 567–578, http://dx.doi.org/10.1109/ICDCS57875.2023.00033.

[34] U. Ugurlu, I. Oksuz, O. Tas, Electricity price forecasting using recurrent neural networks, Energies 11 (5) (2018) http://dx.doi.org/10.3390/en11051255, URL https://www.mdpi.com/1996-1073/11/5/1255.

[35] L. Zhang, Z. Li, C. Wu, S. Ren, Online electricity cost saving algorithms for co-location data centers, IEEE J. Sel. Areas Commun. 33 (12) (2015) 2906–2919, http://dx.doi.org/10.1109/JSAC.2015.2481280.

[36] D. Zheng, L. Liu, G. Tang, Y. Wang, W. Li, Power demand reshaping using energy storage for distributed edge clouds, IEEE Trans. Parallel Distrib. Syst. 35 (2) (2024) 362–376, http://dx.doi.org/10.1109/TPDS.2023.3347774.

[37] L. Jiang, G. Hu, Day-ahead price forecasting for electricity market using long-short term memory recurrent neural network, in: 2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV, 2018, pp. 949–954, http://dx.doi.org/10.1109/ICARCV.2018.8581235.

[38] M. Ji, Z. Qian, B. Ye, When CPN meets AI: Resource provisioning for inference query upon computing power network, in: 2023 IEEE 29th International Conference on Parallel and Distributed Systems, ICPADS, 2023, pp. 2261–2268, http://dx.doi.org/10.1109/ICPADS60453.2023.00304.

[39] L. Tian, M. Yang, S. Wang, An overview of compute first networking, Int. J. Web Grid Serv. 17 (2) (2021) 81–97, http://dx.doi.org/10.1504/ijwgs.2021.114566.

[40] A. Dogan, D. Cidem Dogan, A review on machine learning models in forecasting of virtual power plant uncertainties, Arch. Comput. Methods Eng. 30 (3) (2023) 2081–2103.

[41] Machine instances, 2023, https://www.aliyun.com/.

[42] E. Cho, J. Yoon, D. Baek, D. Lee, D.-H. Bae, DNN model deployment on distributed edges, in: M. Bakaev, I.-Y. Ko, M. Mrissa, C. Pautasso, A. Srivastava (Eds.), ICWE 2021 Workshops, Springer International Publishing, Cham, 2022, pp. 15–26.

[43] L. He, S. Wang, Y. Xu, P. Kuang, J. Cao, Y. Liu, X. Li, S. Peng, Enabling application-aware traffic engineering in IPv6 networks, IEEE Netw. 36 (2) (2022) 42–49, http://dx.doi.org/10.1109/MNET.005.2100440.

[44] Tokyo Shinjuku Live Ch, 2024, https://www.youtube.com/channel/UCpk2ftN35L3xfoV2S5xLN2A/videos.

[45] D.S. Hochba, Approximation algorithms for NP-hard problems, 28, (2) (ISSN: 0163-5700) 1997, pp. 40–52, http://dx.doi.org/10.1145/261342.571216.

[46] Welcome to CVXPY 1.1, 2023, https://www.cvxpy.org.

[47] L2-Norm, 2021, https://mathworld.wolfram.com/L2-Norm.html.

[48] T.S. Breusch, A.R. Pagan, The Lagrange multiplier test and its applications to model specification in econometrics, Rev. Econ. Stud. 47 (1) (1980) 239–253.

[49] H. Everett, III, Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, Oper. Res. 11 (3) (1963) 399–417.

[50] L.D. Nguyen, A. Kortun, Real-time optimisation for industrial internet of things (IIoT): Overview, challenges and opportunities, EAI Endorsed Trans. Ind. Networks Intell. Syst. 7 (2020) e2, URL https://api.semanticscholar.org/CorpusID:230559373.

[51] M. Ji, C. Su, Y. Fan, Y. Jin, Z. Qian, Y. Yan, Y. Chen, T. Cao, S. Zhang, B. Ye, INTaaS: Provisioning in-band network telemetry as a service via online learning, Comput. Netw. (2024) URL https://api.semanticscholar.org/CorpusID:267262562.

[52] T. Chen, Q. Ling, G.B. Giannakis, An online convex optimization approach to proactive network resource allocation, IEEE Trans. Signal Process. 65 (24) (2017) 6350–6364.

[53] Z. Wang, B. Kim, L.P. Kaelbling, Regret bounds for meta bayesian optimization with an unknown gaussian process prior, Adv. Neural Inf. Process. Syst. 31 (2018).

[54] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, X. Wang, Resource-efficient and convergence-preserving online participant selection in federated learning, in: IEEE ICDCS, 2020, pp. 606–616, http://dx.doi.org/10.1109/ICDCS47774.2020.00049.

[55] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: Optimal speed and accuracy of object detection, 2020, arXiv:2004.10934.

[56] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, 2017, arXiv:1706.05587.

[57] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2019, arXiv:1810.04805.

[58] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2015, arXiv:1409.1556.

[59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, 2015, arXiv:1512.00567.

[60] VOC, 2023, https://pjreddie.com/projects/pascal-voc-dataset-mirror.

[61] THUCTC, 2023, http://thuctc.thunlp.org.

[62] cifar, 2023, https://www.cs.toronto.edu/∼kriz/cifar.html.

[63] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0—Survivable network design library, Networks: Int. J. 55 (3) (2010) 276–286.

[64] Networkx, 2023, https://networkx.org/.

[65] Network latency datasets, 2023, https://github.com/uofa-rzhu3/NetLatency-Data.

[66] PlanetLab, 2023, http://www.planet-lab.org/.

[67] Seattle, 2023, https://seattle.poly.edu/.

[68] Y. Yuan, L. Jiao, K. Zhu, L. Zhang, Incentivizing federated learning under long-term energy constraint via online randomized auctions, IEEE Trans. Wireless Commun. 21 (7) (2022) 5129–5144, http://dx.doi.org/10.1109/TWC.2021.3137024.

[69] T. Ouyang, K. Zhao, X. Zhang, Z. Zhou, X. Chen, Dynamic edge-centric resource provisioning for online and offline services co-location, in: IEEE INFOCOM 2023 - IEEE Conference on Computer Communications, 2023, pp. 1–10, http://dx.doi.org/10.1109/INFOCOM53939.2023.10228949.

[70] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, Learning for learning: Predictive online control of federated learning with edge provisioning, in: IEEE INFOCOM, 2021, pp. 1–10, http://dx.doi.org/10.1109/INFOCOM42981.2021.9488733.

[71] Y. Jin, L. Jiao, M. Ji, Z. Qian, S. Zhang, N. Chen, S. Lu, Scheduling in-band network telemetry with convergence-preserving federated learning, IEEE/ACM Trans. Netw. (2023) 1–16, http://dx.doi.org/10.1109/TNET.2023.3253302.

[72] London open data, 2023, https://tfl.gov.uk/info-for/open-data-users/our-open-data.

[73] C.-C. Hung, G. Ananthanarayanan, L. Golubchik, M. Yu, M. Zhang, Wide-area analytics with multiple resources, in: EuroSys, EuroSys '18, Association for Computing Machinery, New York, NY, USA, 2018, http://dx.doi.org/10.1145/3190508.3190528.

[74] Instances prices, 2023, https://ecs-buy.aliyun.com/.

[75] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, F.C. Lau, Moving big data to the cloud: An online cost-minimizing approach, IEEE J. Sel. Areas Commun. 31 (12) (2013) 2710–2721.

Mingtao Ji received the B.E. degree from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics in 2018. He is currently pursuing the Ph.D. degree under the supervision of Professor Zhuzhong Qian in Nanjing University. To date, he has already published over 15 papers, including in journals such as IEEE/ACM TON, Elsevier COMNET, JCST, and in conferences such as IEEE INFOCOM, IEEE SECON, IEEE ISPA, IEEE ICC, IEEE ICPADS, IEEE/ACM IWQOS poster, and BigCom. He won the Best Paper Award in BigCom 2023. His research interests include P4 switch, SDN, Federated Learning, Big Data Analytics and Distributed Machine Learning.

Ji Qi is a general manager of the Platform Product Department at China Mobile (Suzhou) Software Technology Co., Ltd. He received his Ph.D. in Computer Software and Theory from University of Science and Technology of China. Dr. Qi mainly focuses on cloud computing and big data researches and has published more than 20 papers and holds seven authorized patents as the first completer.

Lei Jiao received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is currently with the Department of Computer Science, University of Oregon, USA. Previously he worked at Nokia Bell Labs in Ireland. He is interested in optimization, control, learning, and mechanism design applied to computer and telecommunication systems, networks, and services. He is a recipient of the NSF CAREER award. He often publishes papers in leading journals such as JSAC, TMC, ToN, and TPDS and conferences such as INFOCOM, MOBIHOC, ICNP, ICDCS, and SECON. He also received the Best Paper Awards of IEEE LANMAN 2013 and IEEE CNS 2019, and the 2016 Alcatel-Lucent Bell Labs UK and Ireland Recognition Award, and has been on the program committees of INFOCOM, MOBIHOC, ICDCS, TheWebConf, and IWQoS.

Gangyi Luo is a senior software architect in China Mobile (Suzhou) Software Technology Co., Ltd. He received his Master Degree of computer science in 2014 at Nanjing University. Currently, his research interests include cloud computing, cloud native and edge computing.

Hehan Zhao received the BS degree from the Faculty of Arts & Science, University of Toronto, in 2023. He is currently an M.Eng. candidate in Electrical and Computer Engineering at the University of Toronto and works as a research assistant in the Distributed Computing Laboratory (Dislab) under the supervision of Professor Zhuzhong Qian at Nanjing University. His works about CPN and In-network Aggregation have been accepted by Journal of Computer Science and Technology and Elsevier Computer Networks. Besides, his research interests include edge computing, reinforcement learning, natural language processing and computer vision.

Xin Li received the B.S. and Ph.D. degrees from Nanjing University in 2008 and 2014, respectively. Currently, he is an associate professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His research interests include cloud computing, edge computing, and distributed computing.

Baoliu Ye is a full professor at the School of Computer Science, Nanjing University. He received his Ph.D. in computer science from Nanjing University, China in 2004. He served as a visiting researcher of the University of Aizu, Japan from March 2005 to July 2006, and the Dean of School of Computer and Information, Hohai University since January 2018. His current research interests mainly include distributed systems, cloud computing, wireless networks with over 70 papers published in major conferences and journals. Prof. Ye served as the TPC co-chair of Hot-POST12, Hot-POST11, P2PNet10. He is the regent of CCF, the Secretary-General of CCF Technical Committee of Distributed Computing and Systems, and a member of IEEE.

Zhuzhong Qian is a professor at the School of Computer Science, Nanjing University, P. R. China. He received his PhD. Degree in computer science in 2007. Currently, his research interests include edge intelligence, datacenter networking and distributed machine learning. He has published over 100 research papers including JSAC, ToN, TPDS, INFOCOM, ICDCS, IPDPS and SECON. He received 6 best paper awards and nominations.